aws

Guida per gli sviluppatori

Amazon Braket

Amazon Braket: Guida per gli sviluppatori

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Cos'è Amazon Braket?

> **Tip**
>
> Scopri le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket è un sistema completamente gestito Servizio AWS che aiuta ricercatori, scienziati e sviluppatori a iniziare con l'informatica quantistica. L'informatica quantistica ha il potenziale per risolvere problemi computazionali che sfuggono alla portata dei computer classici perché sfrutta le leggi della meccanica quantistica per elaborare le informazioni in modi nuovi.

L'accesso all'hardware di calcolo quantistico può essere costoso e scomodo. L'accesso limitato rende difficile eseguire algoritmi, ottimizzare i progetti, valutare lo stato attuale della tecnologia e pianificare quando investire le risorse per ottenere il massimo beneficio. Braket ti aiuta a superare queste sfide.

Braket offre un unico punto di accesso a una varietà di tecnologie di calcolo quantistico. Con Braket puoi:

- Esplora e progetta algoritmi quantistici e ibridi.
- Testa algoritmi su diversi simulatori di circuiti quantistici.
- Esegui algoritmi su diversi tipi di computer quantistici.
- Crea applicazioni dimostrative.

La definizione di problemi quantistici e la programmazione di computer quantistici per risolverli richiede una nuova serie di competenze. Per aiutarti ad acquisire queste competenze, Braket offre diversi ambienti per simulare ed eseguire i tuoi algoritmi quantistici. Puoi trovare l'approccio più adatto alle tue esigenze e iniziare rapidamente con una serie di ambienti di esempio chiamati notebook.

Lo sviluppo di Braket prevede tre fasi:

- Compila
- Test

- Esegui

Informazioni sull'informatica quantistica e su Braket

L'informatica quantistica è nella sua fase iniziale di sviluppo. È importante capire che al momento non esiste un computer quantistico universale e tollerante ai guasti. Pertanto, alcuni tipi di hardware quantistico sono più adatti per ogni caso d'uso ed è fondamentale avere accesso a una varietà di hardware informatico. Braket offre una varietà di hardware tramite fornitori terzi.

L'hardware quantistico esistente è limitato a causa del rumore, che introduce errori. Il settore è nell'era di Noisy Intermediate Scale Quantum (). NISQ All'NISQepoca, i dispositivi di calcolo quantistico erano troppo rumorosi per sostenere algoritmi quantistici puri, come l'algoritmo di Shor o l'algoritmo di Grover. Fino a quando non sarà disponibile una migliore correzione degli errori quantistici, l'informatica quantistica più pratica richiede la combinazione di risorse di calcolo classiche (tradizionali) con computer quantistici per creare algoritmi ibridi. Braket ti aiuta a lavorare con algoritmi quantistici ibridi.

Negli algoritmi quantistici ibridi, le unità di elaborazione quantistica (QPUs) vengono utilizzate come coprocessori perCPUs, velocizzando così calcoli specifici in un algoritmo classico. Questi algoritmi utilizzano l'elaborazione iterativa, in cui il calcolo si sposta tra computer classici e quantistici. Ad esempio, le attuali applicazioni dell'informatica quantistica in chimica, ottimizzazione e apprendimento automatico si basano su algoritmi quantistici variazionali, che sono un tipo di algoritmo quantistico ibrido. Negli algoritmi quantistici variazionali, le routine di ottimizzazione classiche regolano i parametri di un circuito quantistico parametrizzato in modo iterativo, più o meno allo stesso modo in cui i pesi di una rete neurale vengono regolati iterativamente in base all'errore in un set di addestramento per l'apprendimento automatico. Braket offre l'accesso alla libreria software PennyLane open source, che ti assiste con algoritmi quantistici variazionali.

L'informatica quantistica sta guadagnando terreno per i calcoli in quattro aree principali:

- Teoria dei numeri, inclusi fattorizzazione e crittografia (ad esempio, l'algoritmo di Shor è un metodo quantistico primario per i calcoli della teoria dei numeri)
- Ottimizzazione, che include la soddisfazione dei vincoli, la risoluzione di sistemi lineari e l'apprendimento automatico
- Informatica oracolare, che include la ricerca, i sottogruppi nascosti e la ricerca degli ordini (ad esempio, l'algoritmo di Grover è un metodo quantistico primario per i calcoli oracolari)
- Simulazione, che include applicazioni di simulazione diretta, invarianti di nodi e algoritmi di ottimizzazione quantistica approssimativa () QAOA

Le applicazioni per queste categorie di calcoli si trovano nei servizi finanziari, nelle biotecnologie, nella produzione e nei prodotti farmaceutici, solo per citarne alcuni. Braket offre funzionalità e quaderni di esempio che possono già essere applicati a molti problemi di proof of concept oltre a determinati problemi pratici.

In questa sezione:

Come funziona Amazon Braket

> **Tip**
>
> Scopri le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

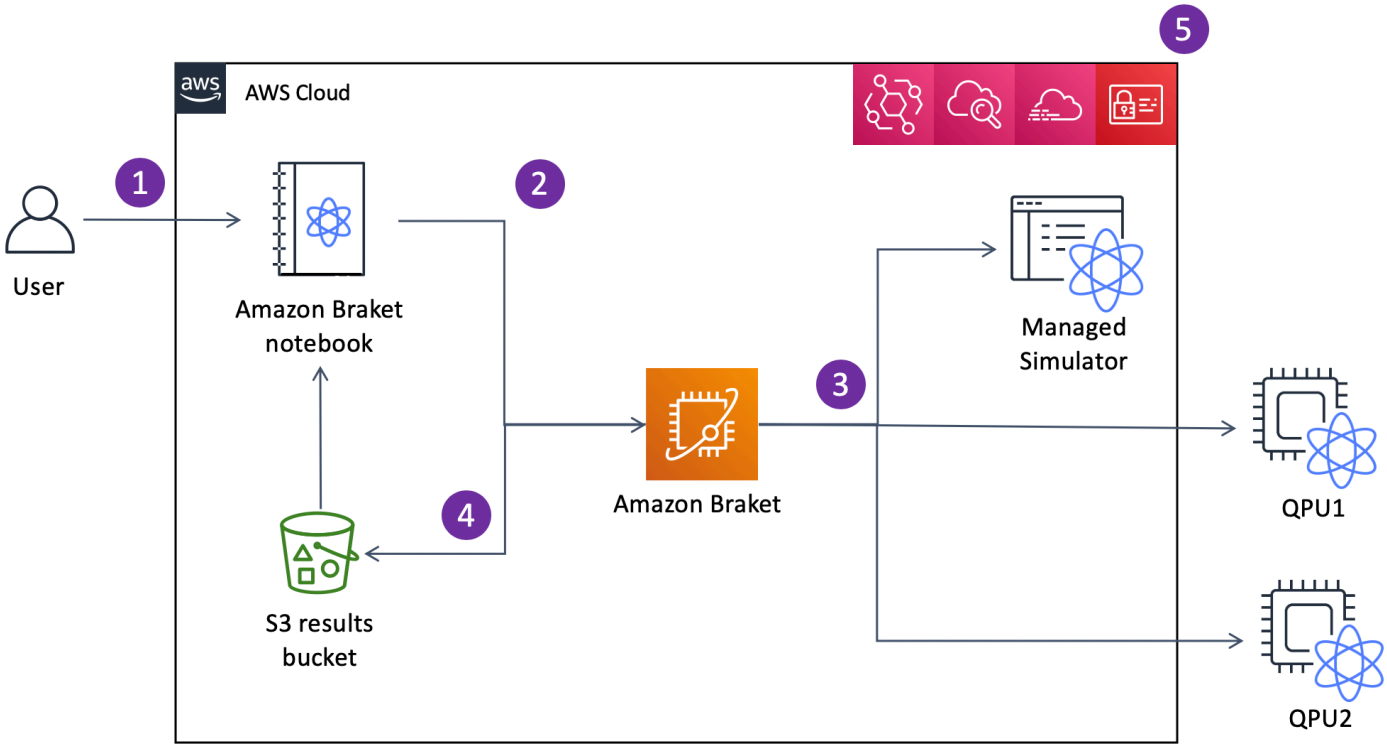
Amazon Braket fornisce l'accesso su richiesta a dispositivi di calcolo quantistico, inclusi simulatori di circuiti on-demand e diversi tipi di. QPUs In Amazon Braket, la richiesta atomica a un dispositivo è un'operazione quantistica. Per i dispositivi QC basati su gate, questa richiesta include il circuito quantistico (comprese le istruzioni di misurazione e il numero di scatti) e altri metadati della richiesta. Per i simulatori hamiltoniani analogici, il task quantistico contiene il layout fisico del registro quantistico e la dipendenza dal tempo e dallo spazio dei campi di manipolazione.

Braket Direct è un programma che amplia il modo in cui è possibile esplorare l'informatica quantistica su AWS, accelerando la ricerca e l'innovazione. È possibile riservare capacità dedicata su vari dispositivi quantistici, interagire direttamente con specialisti di informatica quantistica e avere accesso anticipato alle funzionalità di nuova generazione, incluso il più recente dispositivo a ioni intrappolati di IonQ, Forte: disponibile al pubblico per la prima volta su AWS.

In questa sezione, scopriremo il flusso di alto livello di esecuzione di attività quantistiche su Amazon Braket.

In questa sezione:

Flusso di attività quantistico di Amazon Braket



Con Jupyter con i notebook, puoi definire, inviare e monitorare comodamente le tue attività quantistiche dalla console Amazon Braket o utilizzando Amazon Braket. SDK Puoi costruire i tuoi circuiti quantistici direttamente in. SDK Tuttavia, per Analog Hamiltonian Simulators, è necessario definire il layout del registro e i campi di controllo. Dopo aver definito l'attività quantistica, puoi scegliere un dispositivo su cui eseguirla e inviarla ad Amazon API Braket (2). A seconda del dispositivo scelto, l'operazione quantistica viene messa in coda finché non diventa disponibile e viene inviata al simulatore QPU o per l'implementazione (3). Amazon Braket ti dà accesso a diversi tipi di (QPUsIonQ, IQM, QuEra, Rigetti), tre simulatori su richiesta (SV1, DM1, TN1), due simulatori locali e un simulatore integrato. Per ulteriori informazioni, consulta la pagina Dispositivi supportati da Amazon Braket.

Dopo aver elaborato l'operazione quantistica, Amazon Braket restituisce i risultati in un bucket Amazon S3, dove i dati sono archiviati nel tuo Account AWS (4). Allo stesso tempo, SDK i sondaggi

rilevano i risultati in background e li caricano nel notebook Jupyter al completamento dell'attività quantistica. È inoltre possibile visualizzare e gestire le attività quantistiche nella pagina Attività quantistiche del Amazon Braket console o utilizzando il comando di `GetQuantumTask` Amazon Braket API.

Amazon Braket è integrato con AWS Identity and Access Management (IAM), Amazon CloudWatch, AWS CloudTrail e Amazon EventBridge per la gestione, il monitoraggio e la registrazione degli accessi degli utenti, nonché per l'elaborazione basata sugli eventi (5).

Trattamento dei dati da parte di terzi

Le attività quantistiche inviate a un QPU dispositivo vengono elaborate su computer quantistici situati in strutture gestite da fornitori terzi. Per ulteriori informazioni sulla sicurezza e l'elaborazione di terze parti in Amazon Braket, consulta la sezione Sicurezza dei fornitori di hardware Amazon Braket.

Termini e concetti di Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

In Braket vengono utilizzati i seguenti termini e concetti:

Simulazione hamiltoniana analogica

La simulazione hamiltoniana analogica (AHS) è un paradigma di calcolo quantistico distinto per la simulazione diretta della dinamica quantistica dipendente dal tempo di sistemi a molti corpi. NelAHS, gli utenti specificano direttamente un hamiltoniano dipendente dal tempo e il computer quantistico viene regolato in modo tale da emulare direttamente l'evoluzione temporale continua di questa hamiltoniana. AHSi dispositivi sono in genere dispositivi per scopi speciali e non computer quantistici universali come i dispositivi basati su gate. Sono limitati a una classe di hamiltoniani che possono simulare. Tuttavia, poiché questi hamiltoniani sono naturalmente implementati nel dispositivo, AHS non risentono del sovraccarico necessario per formulare algoritmi sotto forma di circuiti e implementare le operazioni di gate.

Staffa

Abbiamo chiamato il servizio Braket come la notazione bra-ket, una notazione standard della meccanica quantistica. È stato introdotto da Paul Dirac nel 1939 per descrivere lo stato dei sistemi quantistici ed è anche noto come notazione di Dirac.

Staffa Direct

Con Braket Direct, puoi prenotare un accesso dedicato a diversi dispositivi quantistici di tua scelta, entrare in contatto con specialisti di informatica quantistica per ricevere indicazioni sul tuo carico di lavoro e ottenere l'accesso anticipato alle funzionalità di nuova generazione, come i nuovi dispositivi quantistici con disponibilità limitata.

Braket Hybrid Job

Amazon Braket ha una funzione chiamata Amazon Braket Hybrid Jobs che fornisce esecuzioni completamente gestite di algoritmi ibridi. Un lavoro ibrido Braket è composto da tre componenti:

1. La definizione del tuo algoritmo, che può essere fornita come script, modulo Python o contenitore Docker.
2. L'istanza di lavoro ibrida, basata su AmazonEC2, su cui eseguire l'algoritmo. L'impostazione predefinita è un'istanza ml.m5.xlarge.
3. Il dispositivo quantistico su cui eseguire le attività quantistiche che fanno parte dell'algoritmo. Un singolo lavoro ibrido contiene in genere una raccolta di molte attività quantistiche.

Device

In Amazon Braket, un dispositivo è un backend in grado di eseguire attività quantistiche. Un dispositivo può essere un simulatore di circuiti quantistici QPUo un simulatore di circuiti. Per ulteriori informazioni, consulta la pagina Dispositivi supportati da Amazon Braket.

Informatica quantistica basata su Gate

Nel calcolo quantistico basato su gate (QC), chiamato anche QC basato su circuiti, i calcoli sono suddivisi in operazioni elementari (gate). Alcuni set di porte sono universali, il che significa che ogni calcolo può essere espresso come una sequenza finita di tali porte. Le porte sono gli elementi costitutivi dei circuiti quantistici e sono analoghe alle porte logiche dei circuiti digitali classici.

Hamiltoniano

La dinamica quantistica di un sistema fisico è determinata dalla sua formula hamiltoniana, che codifica tutte le informazioni sulle interazioni tra i componenti del sistema e gli effetti delle

L'hamiltoniana di un sistema a N-qubit è comunemente rappresentata come una matrice di numeri complessi 2 N per 2 N su forze motrici esogene.
macchine classiche.

Eseguendo una simulazione hamiltoniana analogica su un dispositivo quantistico, è possibile evitare questi requisiti esponenziali di risorse.

Impulso

Un impulso è un segnale fisico transitorio trasmesso ai qubit. È descritto da una forma d'onda riprodotta in un frame che funge da supporto per il segnale portante ed è collegato al canale o alla porta hardware. I clienti possono progettare i propri impulsi fornendo l'involucro analogico che modula il segnale portante sinusoidale ad alta frequenza. Il frame è descritto in modo univoco da una frequenza e da una fase che spesso vengono scelte in risonanza con la separazione di energia tra i livelli di energia per |0# e |1# del qubit. I gate vengono quindi attivati come impulsi con una forma predeterminata e parametri calibrati come ampiezza, frequenza e durata. I casi d'uso non coperti dalle forme d'onda modello verranno abilitati tramite forme d'onda personalizzate che verranno specificate alla risoluzione del singolo campione fornendo un elenco di valori separati da un tempo di ciclo fisico fisso.

Circuito quantistico

Un circuito quantistico è il set di istruzioni che definisce un calcolo su un computer quantistico basato su gate. Un circuito quantistico è una sequenza di porte quantistiche, che sono trasformazioni reversibili su un qubit registro, insieme alle istruzioni di misurazione.

Simulatore di circuiti quantistici

Un simulatore di circuiti quantistici è un programma per computer che funziona su computer classici e calcola i risultati di misurazione di un circuito quantistico. Per i circuiti generali, il fabbisogno di risorse di una simulazione quantistica cresce esponenzialmente con il numero di qubits simulare. Braket fornisce l'accesso a entrambi i sistemi gestiti (accessibili tramite Braket). API) e locale (parte di Amazon Simulatori di circuiti quantistici SDK (Braket).

Computer quantistico

Un computer quantistico è un dispositivo fisico che utilizza fenomeni quantomeccanici, come la sovrapposizione e l'entanglement, per eseguire calcoli. Esistono diversi paradigmi di calcolo quantistico (QC), come il controllo di qualità basato su gate.

Unità QPU di elaborazione quantistica ()

A QPU è un dispositivo di calcolo quantistico fisico che può essere eseguito su un'attività quantistica. QPUspuò essere basato su diversi paradigmi di controllo qualità, come il controllo di qualità basato su gate. Per ulteriori informazioni, consulta la pagina Dispositivi supportati da Amazon Braket.

QPUporte native

QPUle porte native possono essere mappate direttamente per controllare gli impulsi dal QPU sistema di controllo. Le porte native possono essere eseguite sul QPU dispositivo senza ulteriore compilazione. Sottoinsieme di porte QPUsupportate. Puoi trovare le porte native di un dispositivo nella pagina Dispositivi del Amazon Console Braket e tramite SDK Braket.

QPUcancelli supportati

QPUle porte supportate sono le porte accettate dal QPU dispositivo. Queste porte potrebbero non essere in grado di funzionare direttamente suQPU, il che significa che potrebbe essere necessario scomporle in porte native. Puoi trovare le porte supportate di un dispositivo nella pagina Dispositivi del Amazon Console Braket e tramite Amazon SDKStaffa.

Compito quantistico

In Braket, un compito quantistico è la richiesta atomica a un dispositivo. Per i dispositivi QC basati su gate, ciò include il circuito quantistico (comprese le istruzioni di misurazione e il numero di shots) e altri metadati della richiesta. È possibile creare attività quantistiche tramite il Amazon Staffa SDK o utilizzando il CreateQuantumTask API operazione diretta. Dopo aver creato un task quantistico, questo verrà messo in coda fino a quando il dispositivo richiesto non sarà disponibile. È possibile visualizzare i task quantistici nella pagina Quantum Tasks del Amazon Braket console o utilizzando il GetQuantumTask oppure SearchQuantumTasks API operazioni.

Qubit

L'unità di informazione di base in un computer quantistico è chiamata a qubit (bit quantistico), proprio come un bit nell'informatica classica. A qubit è un sistema quantistico a due livelli che può essere realizzato mediante diverse implementazioni fisiche, come circuiti superconduttori o singoli ioni e atomi. Altro qubit i tipi si basano su fotoni, spin elettronici o nucleari o sistemi quantistici più esotici.

Queue depth

Queue depth si riferisce al numero di attività quantistiche e lavori ibridi in coda per un particolare dispositivo. Le attività quantistiche di un dispositivo e il conteggio delle code di lavoro ibride sono accessibili tramite Braket Software Development Kit (SDK) oppure Amazon Braket Management Console.

1. La profondità della coda delle attività si riferisce al numero totale di attività quantistiche attualmente in attesa di essere eseguite con priorità normale.

2. La profondità della coda delle attività prioritarie si riferisce al numero totale di attività quantistiche inviate in attesa di essere eseguite Amazon Braket Hybrid Jobs. Queste attività hanno la priorità rispetto alle attività autonome una volta avviato un lavoro ibrido.
3. La profondità della coda dei lavori ibridi si riferisce al numero totale di lavori ibridi attualmente in coda su un dispositivo. Quantum tasks i lavori inviati come parte di un lavoro ibrido hanno la priorità e sono aggregati nel Priority Task Queue.

Queue position

Queue position si riferisce alla posizione attuale dell'attività quantistica o del lavoro ibrido all'interno di una rispettiva coda di dispositivi. Può essere ottenuto per attività quantistiche o lavori ibridi tramite Braket Software Development Kit (SDK) oppure Amazon Braket Management Console.

Shots

Poiché l'informatica quantistica è intrinsecamente probabilistica, qualsiasi circuito deve essere valutato più volte per ottenere un risultato accurato. L'esecuzione e la misurazione di un singolo circuito si chiamano colpo. Il numero di scatti (esecuzioni ripetute) per un circuito viene scelto in base alla precisione desiderata per il risultato.

AWS terminologia e suggerimenti per Amazon Braket

IAMpolitiche

Una IAM politica è un documento che consente o nega le autorizzazioni a Servizi AWS e risorse. IAMle politiche consentono di personalizzare i livelli di accesso degli utenti alle risorse. Ad esempio, puoi consentire agli utenti di accedere a tutti i bucket Amazon S3 all'interno del tuo Account AWS o solo un bucket specifico.

- Procedura ottimale: segui il principio di sicurezza del privilegio minimo quando concedi le autorizzazioni. Seguendo questo principio, contribuite a impedire agli utenti o ai ruoli di disporre di più autorizzazioni di quelle necessarie per eseguire le loro attività quantistiche. Ad esempio, se un dipendente deve accedere solo a un bucket specifico, specifica il bucket nella IAM policy invece di concedere al dipendente l'accesso a tutti i bucket del tuo Account AWS.

IAMruoli

Un IAM ruolo è un'identità che puoi assumere per ottenere l'accesso temporaneo alle autorizzazioni. Prima che un utente, un'applicazione o un servizio possa assumere un IAM ruolo,

è necessario concedere loro le autorizzazioni per passare al ruolo. Quando qualcuno assume un IAM ruolo, abbandona tutte le autorizzazioni precedenti che aveva nell'ambito di un ruolo precedente e assume le autorizzazioni del nuovo ruolo.

- Buone pratiche: IAM i ruoli sono ideali per le situazioni in cui l'accesso ai servizi o alle risorse deve essere concesso temporaneamente, anziché a lungo termine.

Bucket Amazon S3

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) è un Servizio AWS che consente di archiviare i dati come oggetti in bucket. I bucket Amazon S3 offrono spazio di archiviazione illimitato. La dimensione massima per un oggetto in un bucket Amazon S3 è di 5 TB. Puoi caricare qualsiasi tipo di file in un bucket Amazon S3, come immagini, video, file di testo, file di backup, file multimediali per un sito Web, documenti archiviati e risultati delle attività quantistiche di Braket.

- Procedura ottimale: puoi impostare le autorizzazioni per controllare l'accesso al tuo bucket S3. Per ulteriori informazioni, consulta le politiche di Bucket e le politiche degli utenti nella documentazione di Amazon S3.

Monitoraggio e risparmio dei costi

> **Tip**
>
> Scopri le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Con Amazon Braket, hai accesso a risorse di calcolo quantistico su richiesta senza impegno iniziale. I prezzi sono calcolati solo in base all'uso effettivo. Per ulteriori informazioni sui prezzi, visita la nostra pagina dei prezzi.

In questa sezione:

- Monitoraggio dei costi quasi in tempo reale
- Le migliori pratiche per il risparmio sui costi

Monitoraggio dei costi quasi in tempo reale

The Braket ti SDK offre la possibilità di aggiungere un monitoraggio dei costi quasi in tempo reale ai tuoi carichi di lavoro quantistici. Ciascuno dei nostri notebook di esempio include un codice di monitoraggio dei costi per fornirti una stima massima dei costi relativi alle unità di elaborazione quantistica () e ai simulatori on-demand di Braket. QPUs Le stime dei costi massimi verranno mostrate USD e non includono crediti o sconti.

> **Note**
>
> I costi indicati sono stime basate sul simulatore Amazon Braket e sull'utilizzo delle attività dell'unità di elaborazione quantistica ()QPU. Le spese stimate mostrate possono differire dalle spese effettive. I costi stimati non tengono conto di sconti o crediti e potrebbero verificarsi costi aggiuntivi in base all'utilizzo di altri servizi come Amazon Elastic Compute Cloud (AmazonEC2).

Monitoraggio dei costi per SV1

Per dimostrare come può essere utilizzata la funzione di tracciamento dei costi, costruiremo un circuito Bell State e lo eseguiremo sul nostro SV1 simulatore. Inizia importando i SDK moduli Braket, definendo un Bell State e aggiungendo la `Tracker()` funzione al nostro circuito:

```
#import any required modules
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.tracking import Tracker

#create our bell circuit
circ = Circuit().h(0).cnot(0,1)
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
with Tracker() as tracker:
    task = device.run(circ, shots=1000).result()

#Your results
print(task.measurement_counts)
```

Quando esegui il tuo Notebook, puoi aspettarti il seguente risultato per la tua simulazione Bell State. La funzione tracker vi mostrerà il numero di scatti inviati, le attività quantistiche completate, la durata

dell'esecuzione, la durata di esecuzione fatturata e il costo massimo in entrata. USD Il tempo di esecuzione può variare per ogni simulazione.

```
tracker.quantum_tasks_statistics()
{'arn:aws:braket:::device/quantum-simulator/amazon/sv1':
 {'shots': 1000,
  'tasks': {'COMPLETED': 1},
  'execution_duration': datetime.timedelta(microseconds=4000),
  'billed_execution_duration': datetime.timedelta(seconds=3)}}

tracker.simulator_tasks_cost()
$0.00375
```

Utilizzo del tracker dei costi per impostare i costi massimi

È possibile utilizzare il tracker dei costi per impostare i costi massimi di un programma. Potresti avere una soglia massima per quanto vuoi spendere per un determinato programma. In questo modo, puoi utilizzare il cost tracker per creare una logica di controllo dei costi nel tuo codice di esecuzione. L'esempio seguente utilizza lo stesso circuito su un Rigetti QPUe limita il costo a 1USD. Il costo per eseguire un'iterazione del circuito nel nostro codice è USD 0,37. Abbiamo impostato la logica per ripetere le iterazioni fino a quando il costo totale supera 1USD; quindi, il frammento di codice verrà eseguito tre volte fino a quando l'iterazione successiva non supererà 1. USD In genere, un programma continua a iterare fino a raggiungere il costo massimo desiderato, in questo caso, tre iterazioni.

```
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3")
with Tracker() as tracker:
    while tracker.qpu_tasks_cost() < 1:
        result = device.run(circ, shots=200).result()
print(tracker.quantum_tasks_statistics())
print(tracker.qpu_tasks_cost(), "USD")
```

```
{'arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3': {'shots': 600, 'tasks':
 {'COMPLETED': 3}}}
1.11 USD
```

> **Note**
>
> Il tracker dei costi non terrà traccia della durata in caso di errore TN1 compiti quantistici. Durante un TN1 simulazione, se la prova è completata, ma la fase di contrazione fallisce, il costo della prova non verrà visualizzato nel tracker dei costi.

Le migliori pratiche per il risparmio sui costi

Prendi in considerazione le seguenti best practice per l'utilizzo di Amazon Braket. Risparmia tempo, minimizza i costi ed evita gli errori più comuni.

Verifica con i simulatori

- Verifica i circuiti utilizzando un simulatore prima di eseguirlo su unQPU, in modo da poterlo ottimizzare senza incorrere in costi di utilizzo. QPU
- Sebbene i risultati dell'esecuzione del circuito su un simulatore possano non essere identici a quelli dell'esecuzione del circuito su un simulatoreQPU, è possibile identificare errori di codifica o problemi di configurazione utilizzando un simulatore.

Limita l'accesso degli utenti a determinati dispositivi

- È possibile impostare restrizioni che impediscano agli utenti non autorizzati di inviare attività quantistiche su determinati dispositivi. Il metodo consigliato per limitare l'accesso è con AWS IAM. Per ulteriori informazioni su come eseguire questa operazione, consulta Limitare l'accesso.
- Ti consigliamo di non utilizzare il tuo account amministratore per concedere o limitare l'accesso degli utenti ai dispositivi Amazon Braket.

Imposta allarmi di fatturazione

- Puoi impostare un allarme di fatturazione per avvisarti quando la fattura raggiunge un limite preimpostato. Il metodo consigliato per impostare un allarme è tramite Budget AWS. Puoi impostare budget personalizzati e ricevere avvisi quando i costi o l'utilizzo possono superare l'importo preventivato. Le informazioni sono disponibili all'indirizzo Budget AWS.

Test TN1 attività quantistiche con un numero di puntate basso

- I simulatori costano meno diQHPs, ma alcuni simulatori possono essere costosi se le attività quantistiche vengono eseguite con un numero elevato di colpi. Ti consigliamo di testare il tuo TN1 attività con un valore basso shot contare. Shot il conteggio non influisce sul costo di SV1 e attività di simulazione locali.

Controlla tutte le regioni per le attività quantistiche

- La console mostra le attività quantistiche solo per le tue operazioni correnti Regione AWS. Quando cerchi attività quantistiche fatturabili che sono state inviate, assicurati di controllare tutte le regioni.
- È possibile visualizzare un elenco dei dispositivi e delle regioni associate nella pagina della documentazione dei dispositivi supportati.

APIriferimenti e repository per Amazon Braket

> **Tip**
>
> Scopri le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket fornisce un'APIsinterfaccia a riga di comando che puoi utilizzare per creare e gestire istanze di notebook e addestrare e distribuire modelli. SDKs

- Amazon Braket SDK Python (consigliato)
- Riferimento Amazon Braket API
- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto)

- AWS SDK for Ruby

Puoi anche ottenere esempi di codice dal repository Amazon Braket GitHub Tutorials.

- Tutorial Braket GitHub

Repository principali

Di seguito viene visualizzato un elenco di repository principali che contengono pacchetti chiave utilizzati per Braket:

- Braket SDK Python: usa Braket SDK Python per configurare il codice su Jupyter quaderni nel linguaggio di programmazione Python. Dopo il tuo Jupyter i notebook sono configurati, puoi eseguire il codice su dispositivi e simulatori Braket
- Braket Schemas - Il contratto tra il servizio Braket e il servizio Braket. SDK
- Braket Default Simulator - Tutti i nostri simulatori quantistici locali per Braket (vettore di stato e matrice di densità).

Plug-in

Poi ci sono i vari plugin che vengono utilizzati insieme a vari dispositivi e strumenti di programmazione. Questi includono i plug-in supportati da Braket e i plug-in supportati da terze parti, come mostrato di seguito.

Amazon Braket supportato:

- Libreria di algoritmi Amazon Braket: un catalogo di algoritmi quantistici predefiniti scritti in Python. Eseguili così come sono o usali come punto di partenza per creare algoritmi più complessi.
- Braket- PennyLane plugin - Usa PennyLane come QML framework su Braket.

Terze parti (il team di Braket monitora e contribuisce):

- Provider Qiskit-Braket: utilizza il Qiskit SDKper accedere alle risorse di Braket.
- Braket-Julia SDK - (EXPERIMENTAL) Una versione nativa di Braket per Julia SDK

Regioni e dispositivi supportati da Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

In Amazon Braket, un dispositivo rappresenta un QPU simulatore OR che puoi chiamare per eseguire attività quantistiche. Amazon Braket fornisce l'accesso ai dispositivi da QPU IonQ, IQM, QuErae Rigetti, tre simulatori on-demand, tre simulatori locali e un simulatore integrato. Per tutti i dispositivi, puoi trovare ulteriori proprietà del dispositivo, come la topologia del dispositivo, i dati di calibrazione e i set di porte nativi, nella scheda Dispositivi della console Amazon Braket o tramite. `GetDevice` API Quando si costruisce un circuito con i simulatori, Amazon Braket attualmente richiede l'utilizzo di qubit o indici contigui. Se lavori con Amazon BraketSDK, hai accesso alle proprietà del dispositivo come mostrato nel seguente esempio di codice.

```
from braket.aws import AwsDevice
from braket.devices import LocalSimulator

device = AwsDevice('arn:aws:braket:::device/quantum-simulator/amazon/sv1')
 #SV1
# device = LocalSimulator()
 #Local State Vector Simulator
# device = LocalSimulator("default")
 #Local State Vector Simulator
# device = LocalSimulator(backend="default")
 #Local State Vector Simulator
# device = LocalSimulator(backend="braket_sv")
 #Local State Vector Simulator
# device = LocalSimulator(backend="braket_dm")
 #Local Density Matrix Simulator
# device = LocalSimulator(backend="braket_ahs")
 #Local Analog Hamiltonian Simulation
# device = AwsDevice('arn:aws:braket:::device/quantum-simulator/amazon/tn1')
 #TN1
# device = AwsDevice('arn:aws:braket:::device/quantum-simulator/amazon/dm1')
 #DM1
```

```
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1')
 #IonQ
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/ionq/Aria-2')
 #IonQ
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/ionq/Forte-1')
 #IonQ
# device = AwsDevice('arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet')
 #IQM Garnet
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/quera/Aquila')
 #QuEra Aquila
# device = AwsDevice('arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3')
 #Rigetti Aspen-M-3
# device = AwsDevice('arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-2')
 #Rigetti Ankaa-2

# get device properties
device.properties
```

Fornitori di hardware quantistico supportati

- IonQ
- IQM
- QuEra Computing
- Rigetti

Simulatori supportati

- Simulatore vettoriale di stato locale (braket_sv) ('Simulatore predefinito')
- Simulatore di `braket_dm` matrici di densità locale ()
- Simulatore locale AHS
- Simulatore vettoriale di stato (SV1)
- Simulatore di matrici di densità (DM1)
- Simulatore di rete Tensor (TN1)
- PennyLaneè Lightning Simulators

Scegli il simulatore migliore per la tua attività quantistica

- Confronta i simulatori

Dispositivi Amazon Braket

Provider	Nome dispositivo	Paradigma	Tipo	Dispositivo ARN	Regione
IonQ	Aria 1	basato su gate	QPU	arn:aws:braket:us-east-1: :device/QPU/ionq/Aria-1	us-east-1
IonQ	Aria 2	basato su gate	QPU	arn:aws:braket:us-east-1: :device/QPU/ionq/Aria-2	us-east-1
IonQ	Forte 1	basato su gate	QPU(solo su prenotazione)	arn:aws:braket:us-east-1: :device/QPU/ionq/forte-1	us-east-1
IQM	Garnet	basato su gate	QPU	arn:aws:braket:eu-north-1: :device/QPU/iQM/Garnet	eu-north-1
QuEra	Aquila	Simulazione hamiltoniana analogica	QPU	arn:aws:braket:us-east-1: :device/QPU/Quera/Aquila	us-east-1
Rigetti	Aspen M-3	basato su gate	QPU	arn:aws:braket:us-west-1: :device/QPU/Rigetti/Aspen-M-3	us-west-1
Rigetti	Ankaa-2	basato su gate	QPU	arn:aws:braket:us-west-1: :device/QPU/Rigetti/ANKAA-2	us-west-1
AWS	braket_sv	basato su gate	simulatore locale	N/A (simulatore locale in Braket) SDK	N/D

Provider	Nome dispositivo	Paradigma	Tipo	Dispositivo ARN	Regione
AWS	braket_dm	basato su gate	simulatore locale	N/A (simulatore locale in Braket) SDK	N/D
AWS	SV1	basato su gate	simulatore su richiesta	arn:aws:braket: ::device/quantum-simulator/amazon/sv1	Tutte le regioni in cui Amazon La staffa è disponibile.
AWS	DM1	basato su cancello	simulatore su richiesta	arn:aws:braket: ::device/quantum-simulator/amazon/dm1	Tutte le regioni in cui Amazon La staffa è disponibile.
AWS	TN1	basato su cancello	simulatore su richiesta	arn:aws:braket: ::device/quantum-simulator/amazon/tn1	us-west-2, us-east-1 e eu-west-2

> **Note**
>
> QPUsÈ possibile accedervi solo utilizzando le prenotazioni tramite Braket Direct, vedi Prenotazioni.

Per visualizzare ulteriori dettagli su che QPUs puoi utilizzare con Amazon Braket, consulta Amazon Braket Hardware Provider.

Regioni ed endpoint di Amazon Braket

Amazon Braket è disponibile nei seguenti paesi Regioni AWS:

Disponibilità regionale di Amazon Braket

Nome della regione	Regione	Braket Endpoint	QPUse simulatori
US East (N. Virginia)	us-east-1	braket.us-east-1.amazonaws.com	IonQ,, QuEra, SV1 DM1 TN1
US West (N. California)	us-west-1	braket.us-west-1.amazonaws.com	Rigetti, SV1 DM1
Stati Uniti Occidentali 2 (Oregon)	us-west-2	braket.us-west-2.amazonaws.com	SV1, DM1, TN1
EU North 1 (Stoccolma)	eu-north-1	braket.eu-north-1.amazonaws.com	IQM
EU West 2 (Londra)	eu-west-2	braket.eu-west-2.amazonaws.com	SV1, DM1, TN1

Le attività quantistiche eseguite su un QPU dispositivo possono essere visualizzate nella console Amazon Braket nella regione di quel dispositivo. Se utilizzi Amazon BraketSDK, puoi inviare attività quantistiche a qualsiasi QPU dispositivo, indipendentemente dalla regione in cui lavori. Crea SDK automaticamente una sessione nella regione per la regione specificata. QPU

Per informazioni generali su come AWS funziona con regioni ed endpoint, vedi Servizio AWS endpoint in AWS Riferimento generale.

Guida introduttiva ad Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Dopo aver seguito le istruzioni in Abilita Amazon Braket, puoi iniziare a usare Amazon Braket.

I passaggi per iniziare includono:

- Abilita Amazon Braket
- Crea un'istanza di notebook Amazon Braket
- Crea un'istanza di notebook Amazon Braket utilizzando AWS CloudFormation

Abilita Amazon Braket

> **Tip**
>
> Scopri le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Puoi abilitare Amazon Braket nel tuo account tramite AWS console.

In questa sezione:

- Prerequisiti
- Passaggi per abilitare Amazon Braket

Prerequisiti

Per abilitare ed eseguire Amazon Braket, devi disporre di un utente o di un ruolo con l'autorizzazione per avviare azioni Amazon Braket. Queste autorizzazioni sono incluse nel AmazonBraketFullAccess IAMpolitica (arn:aws:iam: :aws:policy/). AmazonBraketFullAccess

> **Note**
>
> Se sei un amministratore:
> Per consentire ad altri utenti di accedere ad Amazon Braket, concedi le autorizzazioni agli utenti allegando la AmazonBraketFullAccesspolicy o allegando una policy personalizzata creata da te. Per ulteriori informazioni sulle autorizzazioni necessarie per utilizzare Amazon Braket, consulta Gestire l'accesso ad Amazon Braket.

Passaggi per abilitare Amazon Braket

1. Accedi alla console Amazon Braket con il Account AWS.
2. Apri la console Amazon Braket.
3. Dalla pagina di destinazione di Braket, fai clic su Inizia per accedere alla pagina Service Dashboard. L'avviso nella parte superiore della dashboard del servizio ti guiderà nei tre passaggi seguenti:
   a. Creazione di ruoli collegati ai servizi () SLR
   b. Abilitazione dell'accesso a computer quantistici di terze parti
   c. Creazione di una nuova istanza del notebook Jupyter

Per utilizzare dispositivi quantistici di terze parti, è necessario accettare determinate condizioni relative al trasferimento di dati tra di voi, AWS e quei dispositivi. I termini e le condizioni del presente contratto sono disponibili nella scheda Generale della pagina Autorizzazioni e impostazioni della console Amazon Braket.

> **Note**
>
> I dispositivi quantistici che non coinvolgono terze parti, come i simulatori locali Braket o i simulatori on-demand, possono essere utilizzati senza accettare il contratto Enable third devices.

L'accettazione di questi termini per consentire l'uso di dispositivi di terze parti deve essere effettuata solo una volta per account se si accede a hardware di terze parti.

Crea un'istanza di notebook Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket offre notebook Jupyter completamente gestiti per aiutarti a iniziare. Le istanze di notebook Amazon Braket sono basate su istanze di notebook Amazon SageMaker . Le seguenti istruzioni descrivono i passaggi necessari per creare una nuova istanza di notebook per clienti nuovi ed esistenti.

Nuovi clienti Amazon Braket

1. Apri la console Amazon Braket e accedi alla pagina Dashboard nel riquadro a sinistra.
2. Fai clic su Inizia nella finestra modale Benvenuto in Amazon Braket, situata al centro della pagina del pannello di controllo, per fornire un nome per il notebook. Verrà creato un notebook Jupyter predefinito.
3. La creazione del notebook può richiedere diversi minuti. Il blocco appunti verrà elencato nella pagina Taccuini con lo stato In sospeso. Quando l'istanza del notebook è pronta per l'uso, lo stato cambia in. InService Potrebbe essere necessario aggiornare la pagina per visualizzare lo stato aggiornato del notebook.

Clienti Amazon Braket esistenti

1. Apri la console Amazon Braket, seleziona Notebook nel riquadro a sinistra, scegli Crea istanza notebook. Se non disponi di notebook, seleziona la configurazione Standard per creare un notebook Jupyter predefinito e inserisci il nome dell'istanza Notebook utilizzando solo caratteri alfanumerici e trattini e seleziona la modalità visiva preferita. Quindi, abilita o disabilita il gestore dell'inattività per il tuo notebook.

a. Se abilitato, seleziona la durata di inattività desiderata prima del ripristino del notebook. Quando un notebook viene ripristinato, i costi di elaborazione smetteranno di essere addebitati, ma i costi di archiviazione continueranno.

b. Per visualizzare il tempo di inattività rimanente nell'istanza del notebook, accedi alla barra dei comandi e seleziona la scheda Braket, seguita dalla scheda Inactivity Manager.

> **Note**
>
> Per evitare che il tuo lavoro vada perso, considera l'integrazione dell'istanza del tuo SageMaker notebook con un repository git. In alternativa, lo spostamento del lavoro all'esterno delle `/Braket Examples` cartelle `/Braket Algorithms` and impedirà che il file venga sovrascritto dal riavvio dell'istanza del notebook.

2. (Facoltativo) Con la configurazione avanzata è possibile creare un notebook con autorizzazioni di accesso, configurazioni aggiuntive e impostazioni di accesso alla rete:

   a. Nella configurazione Notebook scegli il tipo di istanza. Per impostazione predefinita, viene scelto il tipo di istanza standard ed economico ml.t3.medium. Per ulteriori informazioni sui prezzi delle istanze, consulta SageMaker i prezzi di Amazon. Se desideri associare un repository Github pubblico all'istanza del tuo notebook, fai clic sul menu a discesa Git repository e seleziona Clona un repository git pubblico dall'url dal menu a discesa Repository. Inserisci il repo nella barra URL di URL testo del repository Git.

   b. In Autorizzazioni, configura eventuali IAM ruoli opzionali, accesso root e chiavi di crittografia.

   c. In Rete, configura le impostazioni di rete e di accesso personalizzate per Jupyter Notebook istanza.

3. Controlla le impostazioni, imposta eventuali tag per identificare l'istanza del tuo notebook e fai clic su Avvia.

> **Note**
>
> Puoi visualizzare e gestire le istanze dei tuoi notebook Amazon Braket nelle console Amazon Braket e Amazon. SageMaker Ulteriori impostazioni del notebook Amazon Braket sono disponibili tramite la SageMaker console.

Se lavori nella console Amazon Braket all'interno AWS Amazon Braket SDK e i plugin sono precaricati nei notebook che hai creato. Se desideri eseguirlo sul tuo computer, puoi installare i

plugin SDK and quando esegui il comando `pip install amazon-braket-sdk` o quando esegui il comando da utilizzare con i plugin. `pip install amazon-braket-pennylane-plugin` PennyLane

Crea un'istanza di notebook Amazon Braket utilizzando AWS CloudFormation

> **Tip**
>
> Impara le basi dell'informatica quantistica con AWS! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

È possibile utilizzare… AWS CloudFormation per gestire le istanze dei tuoi notebook Amazon Braket. Le istanze per notebook Braket sono basate su Amazon. SageMaker Con CloudFormation, puoi fornire a un'istanza notebook un file modello che descrive la configurazione desiderata. Il file modello è scritto in JSON o YAML formato. È possibile creare, aggiornare ed eliminare le istanze in modo ordinato e ripetibile. Potresti trovarlo utile quando gestisci più istanze di Braket Notebook al tuo interno Account AWS.

Dopo aver creato un CloudFormation modello per un taccuino Braket, usi AWS CloudFormation per distribuire la risorsa. Per ulteriori informazioni, consulta Creazione di uno stack su AWS CloudFormation console in AWS CloudFormation guida per l'utente.

Per creare un'istanza del notebook Braket utilizzando CloudFormation, esegui questi tre passaggi:

1. Crea uno script di configurazione del SageMaker ciclo di vita di Amazon.
2. Crea un AWS Identity and Access Management (IAM) ruolo da assumere SageMaker.
3. Crea un'istanza di SageMaker notebook con il prefisso **`amazon-braket-`**

Puoi riutilizzare la configurazione del ciclo di vita per tutti i notebook Braket che crei. Puoi anche riutilizzare il IAM ruolo per i notebook Braket a cui assegni le stesse autorizzazioni di esecuzione.

In questa sezione:

- Fase 1: creare uno script di configurazione del SageMaker ciclo di vita di Amazon

Fase 1: creare uno script di configurazione del SageMaker ciclo di vita di Amazon

Utilizza il seguente modello per creare uno script di configurazione del SageMaker ciclo di vita. Lo script personalizza un'istanza di SageMaker notebook per Braket. Per le opzioni di configurazione per la CloudFormation risorsa del ciclo di vita, vedere AWS:::: SageMaker nella NotebookInstanceLifecycleConfig AWS CloudFormation guida per l'utente.

```
BraketNotebookInstanceLifecycleConfig:
    Type: "AWS::SageMaker::NotebookInstanceLifecycleConfig"
    Properties:
      NotebookInstanceLifecycleConfigName: BraketLifecycleConfig-${AWS::StackName}
      OnStart:
      - Content:
          Fn::Base64: |
            #!/usr/bin/env bash
            sudo -u ec2-user -i #EOS
            curl -o braket-notebook-lcc.zip https://d3ded4lzb1lnme.cloudfront.net/
notebook/braket-notebook-lcc.zip
            unzip braket-notebook-lcc.zip
            ./install.sh
            EOS

            exit 0
```

Fase 2: Creare il IAM ruolo assunto da Amazon SageMaker

Quando utilizzi un'istanza di Braket Notebook, SageMaker esegue operazioni per tuo conto. Ad esempio, supponiamo di utilizzare un notebook Braket utilizzando un circuito su un dispositivo supportato. All'interno dell'istanza del notebook, SageMaker esegue l'operazione su Braket per te. Il ruolo di esecuzione del notebook definisce le operazioni esatte che SageMaker è consentito eseguire per conto dell'utente. Per ulteriori informazioni, consulta SageMaker i ruoli nella Amazon SageMaker Developer Guide.

Usa l'esempio seguente per creare un ruolo di esecuzione del notebook Braket con le autorizzazioni richieste. È possibile modificare le politiche in base alle proprie esigenze.

> **Note**
>
> Assicurati che il ruolo abbia l'autorizzazione per i bucket `s3:ListBucket` e le `s3:GetObject` operazioni su Amazon S3 con il prefisso. `braketnotebookcdk-"` Lo script di configurazione del ciclo di vita richiede queste autorizzazioni per copiare lo script di installazione del notebook Braket.

```
ExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      RoleName: !Sub AmazonBraketNotebookRole-${AWS::StackName}
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        -
          Effect: "Allow"
          Principal:
            Service:
              - "sagemaker.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/service-role/"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonBraketFullAccess
      Policies:
        -
          PolicyName: "AmazonBraketNotebookPolicy"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - s3:GetObject
                  - s3:PutObject
                  - s3:ListBucket
                Resource:
                  - arn:aws:s3:::amazon-braket-*
                  - arn:aws:s3:::braketnotebookcdk-*
              - Effect: "Allow"
                Action:
                  - "logs:CreateLogStream"
```

```
                - "logs:PutLogEvents"
                - "logs:CreateLogGroup"
                - "logs:DescribeLogStreams"
              Resource:
                - !Sub "arn:aws:logs:*:${AWS::AccountId}:log-group:/aws/sagemaker/*"
            - Effect: "Allow"
              Action:
                - braket:*
              Resource: "*"
```

Passaggio 3: creare un'istanza Amazon SageMaker Notebook con il prefisso **amazon-braket-**

Utilizza lo script del SageMaker ciclo di vita e il IAM ruolo creati nei passaggi 1 e 2 per creare un' SageMaker istanza notebook. L'istanza del notebook è personalizzata per Braket ed è accessibile con la console Amazon Braket. Per ulteriori informazioni sulle opzioni di configurazione per questa CloudFormation risorsa, consulta AWS::SageMaker: NotebookInstance in AWS CloudFormation guida per l'utente.

```
BraketNotebook:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t3.medium
      NotebookInstanceName: !Sub amazon-braket-notebook-${AWS::StackName}
      RoleArn: !GetAtt ExecutionRole.Arn
      VolumeSizeInGB: 30
      LifecycleConfigName: !GetAtt
 BraketNotebookInstanceLifecycleConfig.NotebookInstanceLifecycleConfigName
```

Sviluppa le tue attività quantistiche con Amazon Braket

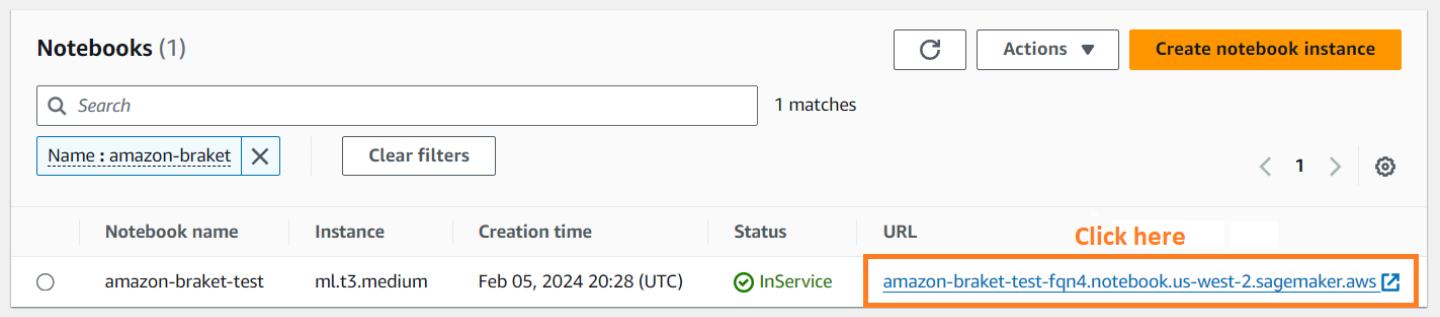
Braket offre ambienti notebook Jupyter completamente gestiti che semplificano l'avvio. I notebook Braket sono preinstallati con algoritmi, risorse e strumenti di sviluppo di esempio, incluso Amazon Braket. SDK Con Amazon BraketSDK, puoi creare algoritmi quantistici e poi testarli ed eseguirli su diversi computer e simulatori quantistici modificando una singola riga di codice.

In questa sezione:

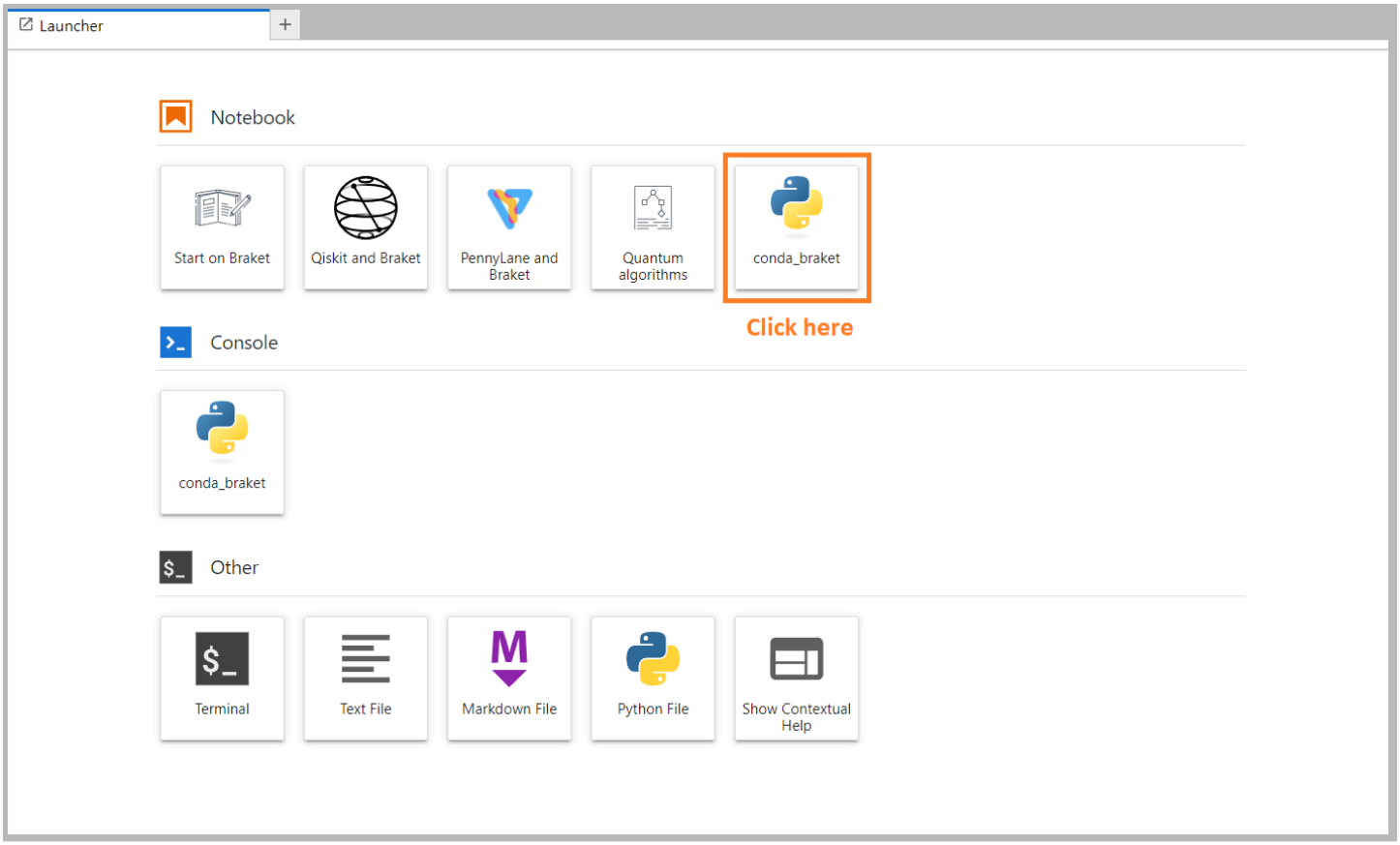
- Costruisci il tuo primo circuito
- Ottenere la consulenza di un esperto
- Guida introduttiva ad Amazon Braket Hybrid Jobs
- Gestisci i tuoi circuiti con Open 3.0 QASM
- Esplora le funzionalità sperimentali
- Controllo a impulsi su Amazon Braket
- Simulazione hamiltoniana analogica
- Lavorare con Boto3

Costruisci il tuo primo circuito

Dopo l'avvio dell'istanza del notebook, apri l'istanza con un'interfaccia Jupyter standard scegliendo il notebook appena creato.



Le istanze di notebook Amazon Braket sono preinstallate con Amazon Braket e tutte le sue dipendenze. SDK Inizia creando un nuovo notebook con kernel. `conda_braket`



Puoi iniziare con un semplice «Hello, world!» esempio. Innanzitutto, costruite un circuito che prepari uno stato di Bell, quindi eseguitelo su dispositivi diversi per ottenere i risultati.

Inizia importando i moduli Amazon SDK Braket e definendo un semplice circuito Bell State.

```
import boto3
from braket.aws import AwsDevice
from braket.devices import LocalSimulator
from braket.circuits import Circuit

# create the circuit
bell = Circuit().h(0).cnot(0, 1)
```

Puoi visualizzare il circuito con questo comando:

```
print(bell)
```

Esegui il tuo circuito sul simulatore locale

Quindi, scegli il dispositivo quantistico su cui eseguire il circuito. Amazon Braket è SDK dotato di un simulatore locale per la prototipazione e i test rapidi. Consigliamo di utilizzare il simulatore locale per circuiti più piccoli, che possono arrivare fino a 25 qubits (a seconda dell'hardware locale).

Ecco come creare un'istanza del simulatore locale:

```
# instantiate the local simulator
local_sim = LocalSimulator()
```

ed esegui il circuito:

```
# run the circuit
result = local_sim.run(bell, shots=1000).result()
counts = result.measurement_counts
print(counts)
```

Dovresti vedere un risultato simile a questo:

```
Counter({'11': 503, '00': 497})
```

Lo stato specifico di Bell che avete preparato è una sovrapposizione uguale di |00> e |11>, e ne troverete uno più o meno uguale (fino a shot distribuzione del rumore) di 00 e 11 come risultati di misurazione, come previsto.

Esegui il tuo circuito su un simulatore on-demand

Amazon Braket fornisce anche l'accesso a un simulatore on-demand ad alte prestazioni, SV1, per far funzionare circuiti più grandi. SV1 è un simulatore vettoriale di stato su richiesta che consente la simulazione di circuiti quantistici fino a 34 qubits. Puoi trovare maggiori informazioni su SV1 nella sezione Dispositivi supportati e nel AWS console. Quando si eseguono attività quantistiche su SV1 (e così via TN1 o qualsiasi altroQPU), i risultati della tua attività quantistica vengono archiviati in un bucket S3 del tuo account. Se non specifichi un bucket, Braket SDK crea automaticamente un bucket predefinito. `amazon-braket-{region}-{accountID}` Per ulteriori informazioni, consulta Gestire l'accesso ad Amazon Braket.

> **Note**
>
> Inserisci il nome effettivo del bucket esistente, come indicato nell'esempio seguente `example-bucket` come nome del bucket. Nomi dei bucket per Amazon La parentesi inizia sempre con `amazon-braket-` seguita da altri caratteri identificativi che aggiungi. Se hai

bisogno di informazioni su come configurare un bucket S3, consulta Guida introduttiva ad Amazon S3.

```
# get the account ID
aws_account_id = boto3.client("sts").get_caller_identity()["Account"]
# the name of the bucket
my_bucket = "example-bucket"
# the name of the folder in the bucket
my_prefix = "simulation-output"
s3_folder = (my_bucket, my_prefix)
```

Per far funzionare un circuito SV1, è necessario fornire la posizione del bucket S3 precedentemente selezionato come argomento posizionale nella chiamata. `.run()`

```
# choose the cloud-based on-demand simulator to run your circuit
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")

# run the circuit
task = device.run(bell, s3_folder, shots=100)
# display the results
print(task.result().measurement_counts)
```

La console Amazon Braket fornisce ulteriori informazioni sulla tua attività quantistica. Vai alla scheda Quantum Tasks nella console e il tuo compito quantistico dovrebbe essere in cima all'elenco. In alternativa, puoi cercare il tuo task quantistico utilizzando l'ID univoco del task quantistico o altri criteri.

> **Note**
>
> Dopo 90 giorni, Amazon Braket rimuove automaticamente tutte le attività quantistiche IDs e gli altri metadati associati alle tue attività quantistiche. Per ulteriori informazioni, consulta Conservazione dei dati.

Esecuzione su un QPU

Con Amazon Braket, puoi eseguire il precedente esempio di circuito quantistico su un computer quantistico fisico semplicemente modificando una singola riga di codice. Amazon Braket fornisce l'accesso a QPU dispositivi da IonQ, IQM, QuErae Rigetti. Puoi trovare informazioni sui diversi

dispositivi e sulle finestre di disponibilità nella sezione Dispositivi supportati e nel AWS console nella scheda Dispositivi. L'esempio seguente mostra come creare un'istanza di IQM dispositivo.

```
# choose the IQM hardware to run your circuit
device = AwsDevice("arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet")
```

Oppure scegli un IonQ dispositivo con questo codice:

```
# choose the Ionq device to run your circuit
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")
```

Dopo aver selezionato un dispositivo e prima di eseguire il carico di lavoro, puoi interrogare la profondità della coda del dispositivo con il codice seguente per determinare il numero di attività quantistiche o lavori ibridi. Inoltre, i clienti possono visualizzare le profondità di coda specifiche del dispositivo nella pagina Dispositivi del Amazon Braket Management Console.

```
# Print your queue depth
print(device.queue_depth().quantum_tasks)
# returns the number of quantum tasks queued on the device
{<QueueType.NORMAL: 'Normal'>: '0', <QueueType.PRIORITY: 'Priority'>: '0'}

print(device.queue_depth().jobs)
'2' # returns the number of hybrid jobs queued on the device
```

Quando esegui un'attività, Amazon Braket verifica SDK il risultato (con un timeout predefinito di 5 giorni). Puoi modificare questa impostazione predefinita modificando il `poll_timeout_seconds` parametro nel `.run()` comando, come mostrato nell'esempio che segue. Tieni presente che se il timeout per il polling è troppo breve, i risultati potrebbero non essere restituiti entro il tempo di polling, ad esempio quando a non QPU è disponibile e viene restituito un errore di timeout locale. Puoi riavviare il polling chiamando la funzione. `task.result()`

```
# define quantum task with 1 day polling timeout
task = device.run(bell, s3_folder, poll_timeout_seconds=24*60*60)
print(task.result().measurement_counts)
```

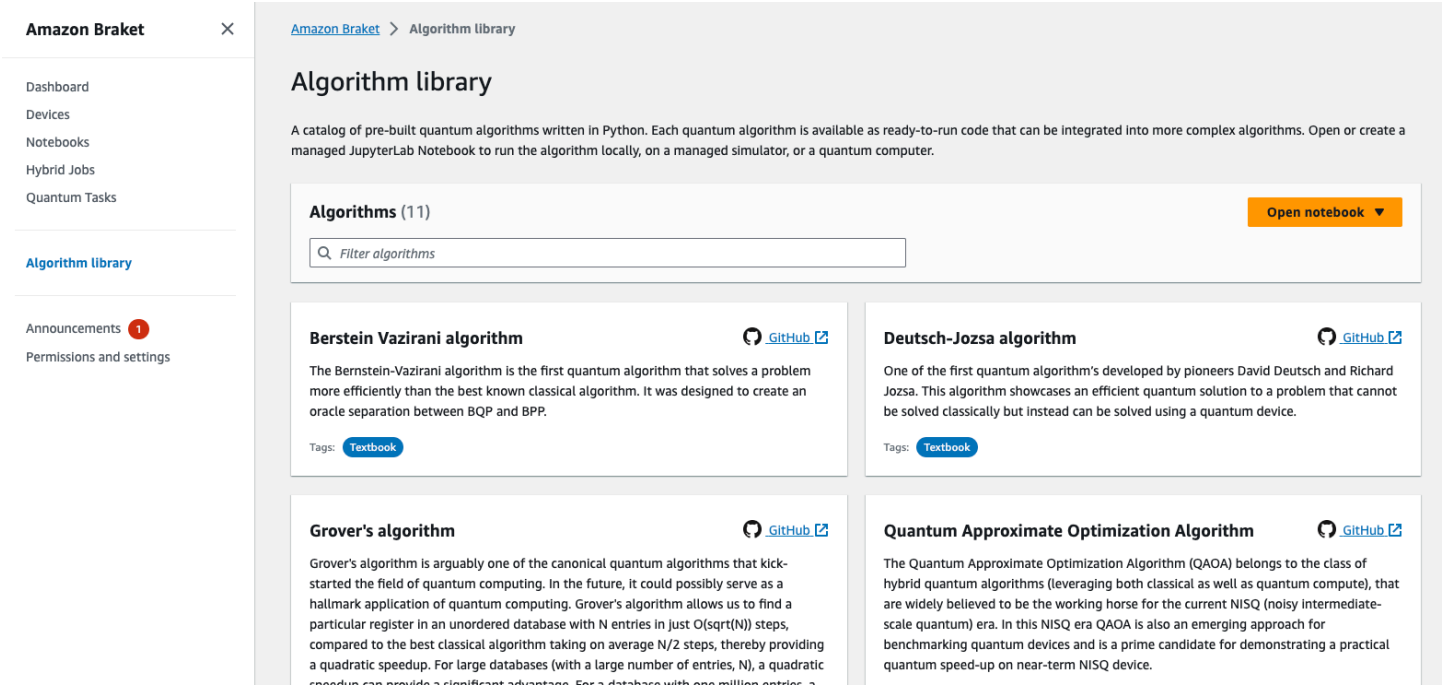
Inoltre, dopo aver inviato l'operazione quantistica o il lavoro ibrido, è possibile richiamare la `queue_position()` funzione per controllare la posizione della coda.

```
print(task.queue_position().queue_position)
# Return the number of quantum tasks queued ahead of you
```

```
'2'
```

Creazione dei primi algoritmi quantistici

La libreria di algoritmi Amazon Braket è un catalogo di algoritmi quantistici predefiniti scritti in Python. Puoi eseguire questi algoritmi così come sono o usarli come punto di partenza per creare algoritmi più complessi. È possibile accedere alla libreria di algoritmi dalla console Braket. Puoi anche accedere alla libreria di algoritmi Braket su Github:. https://github.com/aws-samples/amazon-braket-algorithm-library



La console Braket fornisce una descrizione di ogni algoritmo disponibile nella libreria di algoritmi. Scegliete un GitHub link per visualizzare i dettagli di ogni algoritmo oppure scegliete Apri taccuino per aprire o creare un taccuino che contenga tutti gli algoritmi disponibili. Se scegli l'opzione notebook, puoi trovare la libreria di algoritmi Braket nella cartella principale del tuo notebook.

Costruzione di circuiti in SDK

Questa sezione fornisce esempi di definizione di un circuito, visualizzazione delle porte disponibili, estensione di un circuito e visualizzazione delle porte supportate da ciascun dispositivo. Contiene inoltre istruzioni su come effettuare l'allocazione manuale qubits, ordina al compilatore di eseguire i circuiti esattamente come definiti e crea circuiti rumorosi con un simulatore di rumore.

In Braket puoi anche lavorare a livello di pulsazioni per vari cancelli, tra cui alcuni. QPUs Per ulteriori informazioni, consulta Pulse Control su Amazon Braket.

In questa sezione:

Cancelli e circuiti

Le porte e i circuiti quantistici sono definiti nella `braket.circuits`classe di Amazon Braket Python. SDK DaSDK, puoi creare un'istanza di un nuovo oggetto del circuito chiamando. `Circuit()`

Esempio: definire un circuito

L'esempio inizia definendo un circuito campione di quattro qubits (etichettato`q0`, `q1`q2, `eq3`) costituito da porte Hadamard standard a qubit singolo e porte a due qubit. CNOT È possibile visualizzare questo circuito chiamando la funzione come illustrato nell'esempio seguente. `print`

```
# import the circuit module
from braket.circuits import Circuit

# define circuit with 4 qubits
my_circuit = Circuit().h(range(4)).cnot(control=0, target=2).cnot(control=1, target=3)
print(my_circuit)
```

```
T  : |0| 1 |

q0 : -H-C---
        |
q1 : -H-|-C-
        | |
q2 : -H-X-|-
          |
q3 : -H---X-

T  : |0| 1 |
```

Esempio: definire un circuito parametrizzato

In questo esempio, definiamo un circuito con porte che dipendono da parametri liberi. Possiamo specificare i valori di questi parametri per creare un nuovo circuito o, quando inviamo il circuito, per eseguirlo come attività quantistica su determinati dispositivi.

```
from braket.circuits import Circuit, FreeParameter

#define a FreeParameter to represent the angle of a gate
alpha = FreeParameter("alpha")

#define a circuit with three qubits
my_circuit = Circuit().h(range(3)).cnot(control=0, target=2).rx(0, alpha).rx(1, alpha)
print(my_circuit)
```

È possibile creare un nuovo circuito non parametrizzato a partire da un circuito parametrizzato fornendo un singolo circuito `float` (che è il valore che assumeranno tutti i parametri liberi) o argomenti di parole chiave che specificano il valore di ciascun parametro al circuito come segue.

```
my_fixed_circuit = my_circuit(1.2)
my_fixed_circuit = my_circuit(alpha=1.2)
```

Nota che non `my_circuit` è modificato, quindi puoi usarlo per creare istanziazioni di molti nuovi circuiti con valori di parametro fissi.

Esempio: modifica delle porte in un circuito

L'esempio seguente definisce un circuito con porte che utilizzano modificatori di controllo e potenza. È possibile utilizzare queste modifiche per creare nuove porte, come la porta controllata. `Ry`

```
from braket.circuits import Circuit

# Create a bell circuit with a controlled x gate
my_circuit = Circuit().h(0).x(control=0, target=1)

# Add a multi-controlled Ry gate of angle .13
my_circuit.ry(angle=.13, target=2, control=(0, 1))

# Add a 1/5 root of X gate
my_circuit.x(0, power=1/5)

print(my_circuit)
```

I modificatori di porta sono supportati solo sul simulatore locale.

Esempio: visualizza tutte le porte disponibili

L'esempio seguente mostra come visualizzare tutte le porte disponibili in Amazon Staffa.

```
from braket.circuits import Gate
# print all available gates in Amazon Braket
gate_set = [attr for attr in dir(Gate) if attr[0].isupper()]
print(gate_set)
```

L'output di questo codice elenca tutte le porte.

```
['CCNot', 'CNot', 'CPhaseShift', 'CPhaseShift00', 'CPhaseShift01', 'CPhaseShift10',
 'CSwap', 'CV', 'CY', 'CZ', 'ECR', 'GPi', 'GPi2', 'H', 'I', 'ISwap', 'MS', 'PSwap',
 'PhaseShift', 'PulseGate', 'Rx', 'Ry', 'Rz', 'S', 'Si', 'Swap', 'T', 'Ti', 'Unitary',
 'V', 'Vi', 'X', 'XX', 'XY', 'Y', 'YY', 'Z', 'ZZ']
```

Ognuna di queste porte può essere aggiunta a un circuito chiamando il metodo per quel tipo di circuito. Ad esempio, dovresti chiamare`circ.h(0)`, per aggiungere una porta Hadamard alla prima qubit.

> **Note**
>
> Le porte vengono aggiunte al loro posto e l'esempio che segue aggiunge tutte le porte elencate nell'esempio precedente allo stesso circuito.

```
circ = Circuit()
# toffoli gate with q0, q1 the control qubits and q2 the target.
circ.ccnot(0, 1, 2)
# cnot gate
circ.cnot(0, 1)
# controlled-phase gate that phases the |11> state, cphaseshift(phi) =
 diag((1,1,1,exp(1j*phi))), where phi=0.15 in the examples below
circ.cphaseshift(0, 1, 0.15)
# controlled-phase gate that phases the |00> state, cphaseshift00(phi) =
 diag([exp(1j*phi),1,1,1])
circ.cphaseshift00(0, 1, 0.15)
# controlled-phase gate that phases the |01> state, cphaseshift01(phi) =
 diag([1,exp(1j*phi),1,1])
```

```
circ.cphaseshift01(0, 1, 0.15)
# controlled-phase gate that phases the |10> state, cphaseshift10(phi) =
 diag([1,1,exp(1j*phi),1])
circ.cphaseshift10(0, 1, 0.15)
# controlled swap gate
circ.cswap(0, 1, 2)
# swap gate
circ.swap(0,1)
# phaseshift(phi)= diag([1,exp(1j*phi)])
circ.phaseshift(0,0.15)
# controlled Y gate
circ.cy(0, 1)
# controlled phase gate
circ.cz(0, 1)
# Echoed cross-resonance gate applied to q0, q1
circ = Circuit().ecr(0,1)
# X rotation with angle 0.15
circ.rx(0, 0.15)
# Y rotation with angle 0.15
circ.ry(0, 0.15)
# Z rotation with angle 0.15
circ.rz(0, 0.15)
# Hadamard gates applied to q0, q1, q2
circ.h(range(3))
# identity gates applied to q0, q1, q2
circ.i([0, 1, 2])
# iswap gate, iswap = [[1,0,0,0],[0,0,1j,0],[0,1j,0,0],[0,0,0,1]]
circ.iswap(0, 1)
# pswap gate, PSWAP(phi) = [[1,0,0,0],[0,0,exp(1j*phi),0],[0,exp(1j*phi),0,0],
[0,0,0,1]]
circ.pswap(0, 1, 0.15)
# X gate applied to q1, q2
circ.x([1, 2])
# Y gate applied to q1, q2
circ.y([1, 2])
# Z gate applied to q1, q2
circ.z([1, 2])
# S gate applied to q0, q1, q2
circ.s([0, 1, 2])
# conjugate transpose of S gate applied to q0, q1
circ.si([0, 1])
# T gate applied to q0, q1
circ.t([0, 1])
# conjugate transpose of T gate applied to q0, q1
```

```
circ.ti([0, 1])
# square root of not gate applied to q0, q1, q2
circ.v([0, 1, 2])
# conjugate transpose of square root of not gate applied to q0, q1, q2
circ.vi([0, 1, 2])
# exp(-iXX theta/2)
circ.xx(0, 1, 0.15)
# exp(i(XX+YY) theta/4), where theta=0.15 in the examples below
circ.xy(0, 1, 0.15)
# exp(-iYY theta/2)
circ.yy(0, 1, 0.15)
# exp(-iZZ theta/2)
circ.zz(0, 1, 0.15)
# IonQ native gate GPi with angle 0.15 applied to q0
circ.gpi(0, 0.15)
# IonQ native gate GPi2 with angle 0.15 applied to q0
circ.gpi2(0, 0.15)
# IonQ native gate MS with angles 0.15, 0.15, 0.15 applied to q0, q1
circ.ms(0, 1, 0.15, 0.15, 0.15)
```

Oltre al set di porte predefinito, è possibile applicare al circuito anche porte unitarie autodefinite. Queste possono essere porte a qubit singolo (come mostrato nel codice sorgente seguente) o porte a più qubit applicate a qubits definito dal parametro. `targets`

```
import numpy as np
# apply a general unitary
my_unitary = np.array([[0, 1],[1, 0]])
circ.unitary(matrix=my_unitary, targets=[0])
```

Esempio: estendere i circuiti esistenti

È possibile estendere i circuiti esistenti aggiungendo istruzioni. An `Instruction` è una direttiva quantistica che descrive il compito quantistico da eseguire su un dispositivo quantistico. `Instruction`gli operatori includono solo oggetti di tipo. `Gate`

```
# import the Gate and Instruction modules
from braket.circuits import Gate, Instruction

# add instructions directly.
circ = Circuit([Instruction(Gate.H(), 4), Instruction(Gate.CNot(), [4, 5])])

# or with add_instruction/add functions
```

```
instr = Instruction(Gate.CNot(), [0, 1])
circ.add_instruction(instr)
circ.add(instr)

# specify where the circuit is appended
circ.add_instruction(instr, target=[3, 4])
circ.add_instruction(instr, target_mapping={0: 3, 1: 4})

# print the instructions
print(circ.instructions)
# if there are multiple instructions, you can print them in a for loop
for instr in circ.instructions:
     print(instr)

# instructions can be copied
new_instr = instr.copy()
# appoint the instruction to target
new_instr = instr.copy(target=[5])
new_instr = instr.copy(target_mapping={0: 5})
```

Esempio: visualizza le porte supportate da ogni dispositivo

I simulatori supportano tutti i gate del BraketSDK, ma QPU i dispositivi supportano un sottoinsieme più piccolo. È possibile trovare le porte supportate di un dispositivo nelle proprietà del dispositivo. Di seguito viene mostrato un esempio con un dispositivo IonQ:

```
# import the device module
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")

# get device name
device_name = device.name
# show supportedQuantumOperations (supported gates for a device)
device_operations = device.properties.dict()['action']['braket.ir.openqasm.program']
['supportedOperations']
print('Quantum Gates supported by {}:\n {}'.format(device_name, device_operations))
```

```
Quantum Gates supported by the Aria-1 device:
 ['x', 'y', 'z', 'rx', 'ry', 'rz', 'h', 'cnot', 's', 'si', 't', 'ti', 'v', 'vi', 'xx',
 'yy', 'zz', 'swap']
```

Potrebbe essere necessario compilare le porte supportate in porte native prima di poter essere eseguite su hardware quantistico. Quando invii un circuito, Amazon Braket esegue questa compilazione automaticamente.

Esempio: recupera a livello di codice la fedeltà delle porte native supportate da un dispositivo

È possibile visualizzare le informazioni sulla fedeltà nella pagina Dispositivi della console Braket. A volte è utile accedere alle stesse informazioni a livello di programmazione. Il codice seguente mostra come estrarre le due qubit gate fedelity tra due porte di a. QPU

```
# import the device module
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3")

#specify the qubits
a=10
b=113
print(f"Fidelity of the XY gate between qubits {a} and {b}: ",
 device.properties.provider.specs["2Q"][f"{a}-{b}"]["fXY"])
```

Misurazione parziale

Seguendo gli esempi precedenti, abbiamo misurato tutti i qubit del circuito quantistico. Tuttavia, è possibile misurare singoli qubit o un sottoinsieme di qubit.

Esempio: misura un sottoinsieme di qubit

In questo esempio, dimostriamo una misurazione parziale aggiungendo un'`measure`istruzione con i qubit target alla fine del circuito.

```
# Use the local state vector simulator
device = LocalSimulator()

# Define an example bell circuit and measure qubit 0
circuit = Circuit().h(0).cnot(0, 1).measure(0)

# Run the circuit
task = device.run(circuit, shots=10)

# Get the results
result = task.result()
```

```
# Print the circuit and measured qubits
print(circuit)
print()
print("Measured qubits: ", result.measured_qubits)
```

Manuale qubit assegnazione

Quando si esegue un circuito quantistico su computer quantistici da Rigetti, puoi opzionalmente usare il manuale qubit allocazione per controllare quali qubits vengono utilizzati per il tuo algoritmo. Amazon Braket Console e Amazon SDK Braket ti aiutano a ispezionare i dati di calibrazione più recenti del dispositivo di elaborazione quantistica QPU () selezionato, in modo da poter scegliere il migliore qubits per il tuo esperimento.

Manuale qubit l'allocazione consente di far funzionare i circuiti con maggiore precisione e di analizzarne i singoli qubit proprietà. Ricercatori e utenti esperti ottimizzano la progettazione dei circuiti sulla base dei più recenti dati di calibrazione dei dispositivi e possono ottenere risultati più accurati.

L'esempio seguente mostra come allocare qubits esplicitamente.

```
circ = Circuit().h(0).cnot(0, 7)  # Indices of actual qubits in the QPU
my_task = device.run(circ, s3_location, shots=100, disable_qubit_rewiring=True)
```

Per ulteriori informazioni, consulta gli esempi di Amazon Braket su GitHub, o più specificamente, su questo taccuino: Allocazione di Qubit sui dispositivi. QPU

Compilazione Verbatim

Quando si esegue un circuito quantistico su computer quantistici basati su gate, è possibile indirizzare il compilatore a eseguire i circuiti esattamente come definito senza alcuna modifica. Utilizzando la compilazione letterale, è possibile specificare che un intero circuito venga preservato esattamente come specificato o che vengano conservate solo parti specifiche di esso (supportato da Rigetti solo). Quando si sviluppano algoritmi per il benchmarking dell'hardware o i protocolli di mitigazione degli errori, è necessario avere la possibilità di specificare esattamente le porte e i layout dei circuiti in esecuzione sull'hardware. La compilazione Verbatim ti dà il controllo diretto sul processo di compilazione disattivando alcune fasi di ottimizzazione, garantendo così che i circuiti funzionino esattamente come previsto.

La compilazione Verbatim è attualmente supportata su Rigetti, IonQe IQM dispositivi e richiede l'uso di porte native. Quando si utilizza la compilazione letterale, è consigliabile controllare la topologia del

dispositivo per assicurarsi che le porte vengano richiamate e collegate qubits e che il circuito utilizzi le porte native supportate dall'hardware. L'esempio seguente mostra come accedere a livello di codice all'elenco delle porte native supportate da un dispositivo.

```
device.properties.paradigm.nativeGateSet
```

In Rigetti, qubit il ricablaggio deve essere disattivato impostando l'utilizzo con `disableQubitRewiring=True` la compilazione letterale. Se `disableQubitRewiring=False` è impostato quando si usano le caselle letterali in una compilazione, il circuito quantistico fallisce la convalida e non viene eseguito.

Se la compilazione letterale è abilitata per un circuito ed è eseguita su un circuito QPU che non la supporta, viene generato un errore che indica che un'operazione non supportata ha causato il fallimento dell'operazione. Poiché sempre più hardware quantistico supporta nativamente le funzioni del compilatore, questa funzionalità verrà ampliata per includere questi dispositivi. I dispositivi che supportano la compilazione letterale la includono come operazione supportata quando viene richiesta con il codice seguente.

```
from braket.aws import AwsDevice
from braket.device_schema.device_action_properties import DeviceActionType
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3")
device.properties.action[DeviceActionType.OPENQASM].supportedPragmas
```

Non sono previsti costi aggiuntivi associati all'utilizzo della compilazione letterale. Continueranno a essere addebitati i costi per le attività quantistiche eseguite su QPU dispositivi Braket, istanze notebook e simulatori on-demand in base alle tariffe correnti, come specificato nella pagina dei prezzi di Amazon Braket. Per ulteriori informazioni, consulta il taccuino di esempio della compilazione Verbatim.

> **Note**
>
> Se state usando Open QASM per scrivere i circuiti per IonQ dispositivo e desideri mappare il tuo circuito direttamente sui qubit fisici, devi usare il `#pragma braket verbatim` dato che il `disableQubitRewiring` flag viene completamente ignorato da Open. QASM

Simulazione del rumore

Per creare un'istanza del simulatore di rumore locale è possibile modificare il backend come segue.

```
device = LocalSimulator(backend="braket_dm")
```

È possibile creare circuiti rumorosi in due modi:

1. Costruisci il circuito rumoroso dal basso verso l'alto.
2. Prendi un circuito esistente e privo di rumore e inietta rumore dappertutto.

L'esempio seguente mostra gli approcci che utilizzano un circuito semplice con rumore depolarizzante e un canale Kraus personalizzato.

```
# Bottom up approach
# apply depolarizing noise to qubit 0 with probability of 0.1
circ = Circuit().x(0).x(1).depolarizing(0, probability=0.1)

# create an arbitrary 2-qubit Kraus channel
E0 = scipy.stats.unitary_group.rvs(4) * np.sqrt(0.8)
E1 = scipy.stats.unitary_group.rvs(4) * np.sqrt(0.2)
K = [E0, E1]

# apply a two-qubit Kraus channel to qubits 0 and 2
circ = circ.kraus([0,2], K)
```

```
# Inject noise approach
# define phase damping noise
noise = Noise.PhaseDamping(gamma=0.1)
# the noise channel is applied to all the X gates in the circuit
circ = Circuit().x(0).y(1).cnot(0,2).x(1).z(2)
circ_noise = circ.copy()
circ_noise.apply_gate_noise(noise, target_gates = Gate.X)
```

La gestione di un circuito è la stessa esperienza utente di prima, come illustrato nei due esempi seguenti.

Esempio 1

```
task = device.run(circ, s3_location)
```

Or

Esempio 2

```
task = device.run(circ_noise, s3_location)
```

Per altri esempi, vedi [l'esempio introduttivo del simulatore di rumore Braket]

Ispezione del circuito

Circuiti quantistici in Amazon Braket ha un concetto di pseudo tempo chiamato. `Moments` Ciascuno qubit può provare un solo gate per`Moment`. Lo scopo `Moments` è rendere i circuiti e le loro porte più facili da indirizzare e fornire una struttura temporale.

> **Note**
>
> I momenti generalmente non corrispondono al momento reale in cui le porte vengono eseguite su a. QPU

La profondità di un circuito è data dal numero totale di Momenti in quel circuito. È possibile visualizzare la profondità del circuito chiamando il metodo `circuit.depth` come mostrato nell'esempio seguente.

```
# define a circuit with parametrized gates
circ = Circuit().rx(0, 0.15).ry(1, 0.2).cnot(0,2).zz(1, 3, 0.15).x(0)
print(circ)
print('Total circuit depth:', circ.depth)
```

```
T  : |   0    |    1     |2|

q0 : -Rx(0.15)-C----------X-
                |
q1 : -Ry(0.2)--|-ZZ(0.15)---
                | |
q2 : ----------X-|----------
                 |
q3 : ------------ZZ(0.15)---

T  : |   0    |    1     |2|
Total circuit depth: 3
```

La profondità totale del circuito precedente è 3 (mostrata come momenti `0`1, e2). Puoi controllare il funzionamento del cancello per ogni momento.

`Moments`funziona come un dizionario di coppie chiave-valore.

- La chiave è`MomentsKey()`, che contiene pseudo-tempo e qubit informazioni.
- Il valore viene assegnato nel tipo di`Instructions()`.

```
moments = circ.moments
for key, value in moments.items():
    print(key)
    print(value, "\n")
```

```
MomentsKey(time=0, qubits=QubitSet([Qubit(0)]))
Instruction('operator': Rx('angle': 0.15, 'qubit_count': 1), 'target':
 QubitSet([Qubit(0)]))

MomentsKey(time=0, qubits=QubitSet([Qubit(1)]))
Instruction('operator': Ry('angle': 0.2, 'qubit_count': 1), 'target':
 QubitSet([Qubit(1)]))

MomentsKey(time=1, qubits=QubitSet([Qubit(0), Qubit(2)]))
Instruction('operator': CNot('qubit_count': 2), 'target': QubitSet([Qubit(0),
 Qubit(2)]))

MomentsKey(time=1, qubits=QubitSet([Qubit(1), Qubit(3)]))
Instruction('operator': ZZ('angle': 0.15, 'qubit_count': 2), 'target':
 QubitSet([Qubit(1), Qubit(3)]))

MomentsKey(time=2, qubits=QubitSet([Qubit(0)]))
Instruction('operator': X('qubit_count': 1), 'target': QubitSet([Qubit(0)]))
```

È inoltre possibile aggiungere porte a un circuito tramite`Moments`.

```
new_circ = Circuit()
instructions = [Instruction(Gate.S(), 0),
                Instruction(Gate.CZ(), [1,0]),
                Instruction(Gate.H(), 1)
]
new_circ.moments.add(instructions)
print(new_circ)
```

```
T  : |0|1|2|
```

```
q0 : -S-Z---
        |
q1 : ---C-H-

T  : |0|1|2|
```

Tipi di risultati

Amazon Braket può restituire diversi tipi di risultati quando un circuito viene misurato utilizzando. `ResultType` Un circuito può restituire i seguenti tipi di risultati.

- `AdjointGradient`restituisce il gradiente (derivata vettoriale) del valore atteso di un osservabile fornito. Questo osservabile agisce su un obiettivo fornito rispetto a parametri specificati utilizzando il metodo di differenziazione aggiuntiva. Puoi usare questo metodo solo quando shots=0.
- `Amplitude`restituisce l'ampiezza degli stati quantistici specificati nella funzione d'onda di uscita. È disponibile su SV1 e solo simulatori locali.
- `Expectation`restituisce il valore di aspettativa di un dato osservabile, che può essere specificato con la `Observable` classe introdotta più avanti in questo capitolo. L'obiettivo qubits utilizzato per misurare l'osservabile deve essere specificato e il numero di obiettivi specificati deve essere uguale al numero di qubits su cui agisce l'osservabile. Se non viene specificato alcun obiettivo, l'osservabile deve operare solo su 1 qubit e si applica a tutti qubits in parallelo.
- `Probability`restituisce le probabilità di misurazione degli stati di base computazionali. Se non viene specificato alcun obiettivo, `Probability` restituisce la probabilità di misurare tutti gli stati di base. Se vengono specificati obiettivi, solo le probabilità marginali dei vettori di base sul valore specificato qubits vengono restituiti.
- `Reduced density matrix`restituisce una matrice di densità per un sottosistema dell'obiettivo specificato qubits da un sistema di qubits. Per limitare la dimensione di questo tipo di risultato, Braket limita il numero di obiettivi qubits a un massimo di 8.
- `StateVector`restituisce il vettore di stato completo. È disponibile sul simulatore locale.
- `Sample`restituisce i conteggi delle misurazioni di un obiettivo specificato qubit impostato e osservabile. Se non viene specificato alcun obiettivo, l'osservabile deve operare solo su 1 qubit e si applica a tutti qubits in parallelo. Se vengono specificati obiettivi, il numero di obiettivi specificati deve essere uguale al numero di qubits su cui agisce l'osservabile.
- `Variance`restituisce la varianza (`mean([x-mean(x)]`^{2}`)`) del target specificato qubit impostato e osservabile come tipo di risultato richiesto. Se non viene specificato alcun obiettivo, l'osservabile

deve funzionare solo su 1 qubit e si applica a tutti qubits in parallelo. Altrimenti, il numero di obiettivi specificato deve essere uguale al numero di qubits a cui può essere applicato l'osservabile.

I tipi di risultati supportati per diversi dispositivi:

	Sim locale	SV1	DM1	TN1	Rigetti	IonQ	IQM
Aggiungi Gradiente	N	Y	N	N	N	N	N
Amplitude	Y	Y	N	N	N	N	N
Aspettativa	Y	Y	Y	Y	Y	Y	Y
Probability	Y	Y	Y	N	Y*	Y	Y
Matrice a densità ridotta	Y	N	Y	N	N	N	N
Vettore di stato	Y	N	N	N	N	N	N
Project N.E.M.O.	Y	Y	Y	Y	Y	Y	Y
Varianza	Y	Y	Y	Y	Y	Y	Y

> **Note**
>
> * Rigetti supporta solo tipi di risultati di probabilità fino a 40 qubits.

È possibile verificare i tipi di risultati supportati esaminando le proprietà del dispositivo, come illustrato nell'esempio seguente.

```
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3")

# print the result types supported by this device
for iter in device.properties.action['braket.ir.jaqcd.program'].supportedResultTypes:
    print(iter)
```

```
name='Sample' observables=['x', 'y', 'z', 'h', 'i'] minShots=10 maxShots=100000
name='Expectation' observables=['x', 'y', 'z', 'h', 'i'] minShots=10 maxShots=100000
name='Variance' observables=['x', 'y', 'z', 'h', 'i'] minShots=10 maxShots=100000
name='Probability' observables=None minShots=10 maxShots=100000
```

Per chiamare a`ResultType`, aggiungetelo a un circuito, come mostrato nell'esempio seguente.

```
from braket.circuits import Observable

circ = Circuit().h(0).cnot(0, 1).amplitude(state=["01", "10"])
circ.probability(target=[0, 1])
circ.probability(target=0)
circ.expectation(observable=Observable.Z(), target=0)
circ.sample(observable=Observable.X(), target=0)
circ.state_vector()
circ.variance(observable=Observable.Z(), target=0)

# print one of the result types assigned to the circuit
print(circ.result_types[0])
```

> **Note**
>
> Alcuni dispositivi forniscono misurazioni (ad esempio Rigetti) come risultati e altri forniscono probabilità come risultati (ad esempio IonQ). SDKFornisce una proprietà di misurazione sui risultati, ma per i dispositivi che restituiscono probabilità, viene post-calcolata. Pertanto, dispositivi come quelli forniti da IonQ hanno risultati di misurazione determinati in base alla probabilità poiché le misurazioni per colpo non vengono restituite. È possibile verificare se un risultato è stato post-calcolato visualizzando l'`measurements_copied_from_device`oggetto del risultato, come mostrato in questo file.

Osservabili

Amazon Braket include una `Observable` classe, che può essere utilizzata per specificare un osservabile da misurare.

È possibile applicare al massimo un osservabile univoco non identitario a ciascuno qubit. Se si specificano due o più osservabili non identici diversi allo stesso qubit, viene visualizzato un errore. A tal fine, ogni fattore di un prodotto tensoriale conta come un singolo osservabile, quindi è consentito che più prodotti tensoriali agiscano sullo stesso qubit, a condizione che il fattore che agisce su quello qubit è lo stesso.

Puoi anche ridimensionare un osservabile e aggiungere osservabili (ridimensionati o meno). Questo crea un valore `Sum` che può essere utilizzato nel tipo di risultato. `AdjointGradient`

La `Observable` classe include i seguenti osservabili.

```
Observable.I()
Observable.H()
Observable.X()
Observable.Y()
Observable.Z()

# get the eigenvalues of the observable
print("Eigenvalue:", Observable.H().eigenvalues)
# or whether to rotate the basis to be computational basis
print("Basis rotation gates:",Observable.H().basis_rotation_gates)

# get the tensor product of observable for the multi-qubit case
tensor_product = Observable.Y() @ Observable.Z()
# view the matrix form of an observable by using
print("The matrix form of the observable:\n",Observable.Z().to_matrix())
print("The matrix form of the tensor product:\n",tensor_product.to_matrix())

# also factorize an observable in the tensor form
print("Factorize an observable:",tensor_product.factors)

# self-define observables given it is a Hermitian
print("Self-defined Hermitian:",Observable.Hermitian(matrix=np.array([[0, 1],[1, 0]])))

print("Sum of other (scaled) observables:", 2.0 * Observable.X() @ Observable.X() + 4.0
 * Observable.Z() @ Observable.Z())
```

```
Eigenvalue: [ 1 -1]
Basis rotation gates: (Ry('angle': -0.7853981633974483, 'qubit_count': 1),)
The matrix form of the observable:
 [[ 1.+0.j  0.+0.j]
 [ 0.+0.j -1.+0.j]]
The matrix form of the tensor product:
 [[ 0.+0.j  0.+0.j  0.-1.j  0.-0.j]
 [ 0.+0.j -0.+0.j  0.-0.j  0.+1.j]
 [ 0.+1.j  0.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j -0.-1.j  0.+0.j -0.+0.j]]
Factorize an observable: (Y('qubit_count': 1), Z('qubit_count': 1))
Self-defined Hermitian: Hermitian('qubit_count': 1, 'matrix': [[0.+0.j 1.+0.j], [1.+0.j
 0.+0.j]])
Sum of other (scaled) observables: Sum(TensorProduct(X('qubit_count': 1),
 X('qubit_count': 1)), TensorProduct(Z('qubit_count': 1), Z('qubit_count': 1)))
```

Parametri

I circuiti possono includere parametri liberi, che possono essere utilizzati in modo «costruisci una volta ed esegui più volte» e per calcolare i gradienti. I parametri liberi hanno un nome codificato in una stringa che potete usare per specificarne i valori o per determinare se differenziarli rispetto ad essi.

```
from braket.circuits import Circuit, FreeParameter, Observable
theta = FreeParameter("theta")
phi = FreeParameter("phi")
circ = Circuit().h(0).rx(0, phi).ry(0, phi).cnot(0, 1).xx(0, 1, theta)
circ.adjoint_gradient(observable=Observable.Z() @ Observable.Z(), target=[0, 1],
 parameters = ["phi", theta]
```

Per i parametri da differenziare, specificateli utilizzando il loro nome (come stringa) o tramite riferimento diretto. Nota che il calcolo del gradiente utilizzando il tipo di `AdjointGradient` risultato viene eseguito rispetto al valore atteso dell'osservabile.

Nota: se avete fissato i valori dei parametri liberi passandoli come argomenti al circuito parametrizzato, l'esecuzione di un circuito con `AdjointGradient` come risultato il tipo e i parametri specificati produrrà un errore. Questo perché i parametri che utilizziamo per differenziare non sono più presenti. Guarda l'esempio seguente.

```
device.run(circ(0.2), shots=0) # will error, as no free parameters will be present
```

```
device.run(circ, shots=0, inputs={'phi'=0.2, 'theta'=0.2) # will succeed
```

Ottenere la consulenza di un esperto

Connect con esperti di quantum computing direttamente nella console di gestione Braket per ottenere ulteriori indicazioni sui carichi di lavoro.

Per esplorare le opzioni di consulenza degli esperti tramite Braket Direct, apri la console Braket, scegli Braket Direct nel riquadro di sinistra e vai alla sezione Consigli degli esperti. Sono disponibili le seguenti opzioni di consulenza degli esperti:

- Orari di ufficio Braket: gli orari di ufficio Braket prevedono sessioni individuali, assegnate in base all'ordine di arrivo e si svolgono ogni mese. Ogni fascia oraria d'ufficio disponibile dura 30 minuti ed è gratuita. Parlare con gli esperti di Braket può aiutarti a passare più rapidamente dall'ideazione all'esecuzione esplorando l' use-case-to-device adattamento, identificando le opzioni per sfruttare al meglio Braket per il tuo algoritmo e ricevendo consigli su come utilizzare determinate funzionalità di Braket come Amazon Braket Hybrid Jobs, Braket Pulse o Analog Hamiltonian Simulation.
  - Per iscriverti agli orari d'ufficio di Braket, seleziona Iscriviti e inserisci le informazioni di contatto, i dettagli del carico di lavoro e gli argomenti di discussione desiderati.
  - Riceverai un invito sul calendario per il prossimo slot disponibile tramite e-mail.

  > **Note**
  >
  > Per problemi emergenti o domande rapide sulla risoluzione dei problemi, ti consigliamo di contattare AWS Support. Per domande non urgenti, puoi anche usare il AWS Il forum re:post o il Quantum Computing Stack Exchange, dove puoi sfogliare le domande a cui hai risposto in precedenza e porne di nuove.

- Offerte dei fornitori di hardware quantistico: IonQ, QuErae Rigetti ciascuno fornisce offerte di servizi professionali tramite Marketplace AWS.
  - Per esplorare le loro offerte, seleziona Connect e sfoglia le loro inserzioni.
  - Per saperne di più sulle offerte di servizi professionali, visita il Marketplace AWS, vedi Prodotti per servizi professionali.
- Amazon Quantum Solutions Lab (QSL): QSL è un team collaborativo di ricerca e servizi professionali composto da esperti di informatica quantistica che possono aiutarvi a esplorare in modo efficace l'informatica quantistica e a valutare le prestazioni attuali di questa tecnologia.

- Per contattarliQSL, seleziona Connect e inserisci le informazioni di contatto e i dettagli del caso d'uso.
- Il QSL team ti contatterà tramite e-mail per indicarti i passaggi successivi.

Guida introduttiva ad Amazon Braket Hybrid Jobs

Questa sezione fornisce informazioni sui componenti e istruzioni su come configurare i lavori ibridi in Amazon Braket.

Puoi accedere ai lavori ibridi in Braket utilizzando:

- Amazon Braket Python. SDK
- La console Amazon Braket.
- Amazon Braket API.

In questa sezione:

Cos'è un Hybrid Job?

Amazon Braket Hybrid Jobs ti offre un modo per eseguire algoritmi ibridi quantistici classici che richiedono entrambi i metodi classici AWS risorse e unità di elaborazione quantistica (). QPUs Hybrid Jobs è progettato per attivare le risorse classiche richieste, eseguire l'algoritmo e rilasciare le istanze dopo il completamento, in modo da pagare solo per ciò che si utilizza.

Hybrid Jobs è ideale per algoritmi iterativi di lunga durata che coinvolgono risorse sia classiche che quantistiche. Si invia l'algoritmo per l'esecuzione, Braket lo esegue in un ambiente containerizzato scalabile e si recuperano i risultati quando l'algoritmo è completo.

Inoltre, le attività quantistiche create da un lavoro ibrido traggono vantaggio dall'accodamento con priorità più elevata verso un obiettivo. QPU Ciò garantisce che le attività quantistiche vengano elaborate ed eseguite prima degli altri in coda. Ciò è particolarmente utile per gli algoritmi ibridi

iterativi in cui le attività successive dipendono dai risultati delle attività quantistiche precedenti. Esempi di tali algoritmi includono il Quantum Approximate Optimization Algorithm (QAOA), l'autosolver quantistico variazionale o l'apprendimento automatico quantistico. È inoltre possibile monitorare i progressi dell'algoritmo quasi in tempo reale, in modo da tenere traccia dei costi, del budget o di parametri personalizzati come la perdita di allenamento o i valori di aspettativa.

Quando usare Amazon Braket Hybrid Jobs

Amazon Braket Hybrid Jobs ti consente di eseguire algoritmi ibridi quantistici classici, come Variational Quantum Eigensolver (VQE) e Quantum Approximate Optimization Algorithm (QAOA), che combinano risorse di calcolo classiche con dispositivi di calcolo quantistico per ottimizzare le prestazioni dei sistemi quantistici odierni. Amazon Braket Hybrid Jobs offre tre vantaggi principali:

1. Prestazioni: Amazon Braket Hybrid Jobs offre prestazioni migliori rispetto all'esecuzione di algoritmi ibridi dal tuo ambiente. Mentre il processo è in esecuzione, ha accesso prioritario alla destinazione selezionata. QPU Le attività del tuo job vengono eseguite prima delle altre attività in coda sul dispositivo. Ciò si traduce in tempi di esecuzione più brevi e prevedibili per gli algoritmi ibridi. Amazon Braket Hybrid Jobs supporta anche la compilazione parametrica. Puoi inviare un circuito utilizzando parametri liberi e Braket lo compila una sola volta, senza la necessità di ricompilarlo per i successivi aggiornamenti dei parametri sullo stesso circuito, con tempi di esecuzione ancora più rapidi.
2. Convenienza: Amazon Braket Hybrid Jobs semplifica la configurazione e la gestione dell'ambiente di calcolo e ne semplifica l'esecuzione durante l'esecuzione dell'algoritmo ibrido. Devi solo fornire lo script dell'algoritmo e selezionare un dispositivo quantistico (un'unità di elaborazione quantistica o un simulatore) su cui eseguire. Amazon Braket attende che il dispositivo di destinazione diventi disponibile, attiva le risorse classiche, esegue il carico di lavoro in ambienti container predefiniti, restituisce i risultati ad Amazon Simple Storage Service (Amazon S3) e rilascia le risorse di elaborazione.
3. Metriche: Amazon Braket Hybrid Jobs on-the-fly fornisce approfondimenti sugli algoritmi in esecuzione e fornisce metriche degli algoritmi personalizzabili quasi in tempo reale ad Amazon e alla console Amazon Braket in modo da poter monitorare CloudWatch l'avanzamento dei tuoi algoritmi.

Ingressi, output, variabili ambientali e funzioni di supporto

Oltre al file o ai file che compongono lo script completo dell'algoritmo, il lavoro ibrido può avere input e output aggiuntivi. Quando inizia il processo ibrido, Amazon Braket copia gli input forniti come parte

della creazione del lavoro ibrido nel contenitore che esegue lo script dell'algoritmo. Al termine del processo ibrido, tutti gli output definiti durante l'algoritmo vengono copiati nella posizione Amazon S3 specificata.

> **Note**
>
> Le metriche dell'algoritmo vengono riportate in tempo reale e non seguono questa procedura di output.

Amazon Braket fornisce anche diverse variabili di ambiente e funzioni di supporto per semplificare le interazioni con gli input e gli output dei container.

Questa sezione spiega i concetti chiave della `AwsQuantumJob.create` funzione fornita da Amazon Braket Python SDK e la loro mappatura alla struttura del file contenitore.

In questa sezione:

Input

Dati di input: i dati di input possono essere forniti all'algoritmo ibrido specificando il file di dati di input, che è impostato come dizionario, con l'`input_data`argomento. L'utente definisce l'`input_data`argomento all'interno della `AwsQuantumJob.create` funzione in. SDK Questo copia i dati di input nel file system del contenitore nella posizione indicata dalla variabile di ambiente`"AMZN_BRAKET_INPUT_DIR"`. Per un paio di esempi di come i dati di input vengono utilizzati in un algoritmo ibrido, consulta Amazon Braket Hybrid Jobs PennyLane e Quantum machine learning nei notebook Amazon Braket Hybrid Jobs Jobs Jupyter. QAOA

> **Note**
>
> Quando i dati di input sono di grandi dimensioni (> 1 GB), ci sarà un lungo tempo di attesa prima che il lavoro ibrido venga inviato. Ciò è dovuto al fatto che i dati di input locali verranno

prima caricati su un bucket S3, quindi il percorso S3 verrà aggiunto alla richiesta di lavoro ibrida e, infine, la richiesta di lavoro ibrida verrà inviata al servizio Braket.

Iperparametri: se si passa`hyperparameters`, sono disponibili nella variabile di ambiente. `"AMZN_BRAKET_HP_FILE"`

> **Note**
>
> Per ulteriori informazioni su come creare iperparametri e dati di input e quindi passare queste informazioni allo script di lavoro ibrido, consulta la sezione Use hyperparameters e questa pagina github.

Punti di controllo: per specificare `job-arn` il checkpoint di cui desideri utilizzare in un nuovo lavoro ibrido, usa il comando. `copy_checkpoints_from_job` Questo comando copia i dati del checkpoint nel nuovo processo ibrido, rendendoli disponibili nel percorso indicato dalla variabile di ambiente `AMZN_BRAKET_CHECKPOINT_DIR` durante l'esecuzione del lavoro. `checkpoint_configs3Uri` L'impostazione predefinita è `None` che i dati del checkpoint di un altro lavoro ibrido non verranno utilizzati nel nuovo lavoro ibrido.

Output

Attività quantistiche: i risultati delle attività quantistiche vengono archiviati nella posizione S3. `s3://amazon-braket-<region>-<accountID>/jobs/<job-name>/tasks`

Risultati del lavoro: tutto ciò che lo script dell'algoritmo salva nella directory fornita dalla variabile di ambiente `"AMZN_BRAKET_JOB_RESULTS_DIR"` viene copiato nella posizione S3 specificata in. `output_data_config` Se non specifichi questo valore, il valore predefinito è. `s3://amazon-braket-<region>-<accountID>/jobs/<job-name>/<timestamp>/data` Forniamo la funzione di SDK supporto **`save_job_result`**, che puoi utilizzare per memorizzare comodamente i risultati sotto forma di dizionario quando richiami dallo script dell'algoritmo.

Punti di controllo: se si desidera utilizzare i checkpoint, è possibile salvarli nella directory fornita dalla variabile di ambiente. `"AMZN_BRAKET_CHECKPOINT_DIR"` Al suo posto puoi anche usare la funzione SDK helper. `save_job_checkpoint`

Metriche dell'algoritmo: puoi definire le metriche dell'algoritmo come parte dello script dell'algoritmo che vengono emesse su Amazon CloudWatch e visualizzate in tempo reale nel Amazon Blocca la

console mentre il processo ibrido è in esecuzione. Per un esempio di come utilizzare le metriche degli algoritmi, consulta Use Amazon Braket Hybrid Jobs per eseguire un QAOA algoritmo.

Variabili di ambiente

Amazon Braket fornisce diverse variabili di ambiente per semplificare le interazioni con gli input e gli output dei container. Il codice seguente elenca le variabili ambientali utilizzate da Braket.

```
# the input data directory opt/braket/input/data
os.environ["AMZN_BRAKET_INPUT_DIR"]
# the output directory opt/braket/model to write job results to
os.environ["AMZN_BRAKET_JOB_RESULTS_DIR"]
# the name of the job
os.environ["AMZN_BRAKET_JOB_NAME"]
# the checkpoint directory
os.environ["AMZN_BRAKET_CHECKPOINT_DIR"]
# the file containing the hyperparameters
os.environ["AMZN_BRAKET_HP_FILE"]
# the device ARN (AWS Resource Name)
os.environ["AMZN_BRAKET_DEVICE_ARN"]
# the output S3 bucket, as specified in the CreateJob request’s OutputDataConfig
os.environ["AMZN_BRAKET_OUT_S3_BUCKET"]
# the entry point as specified in the CreateJob request’s ScriptModeConfig
os.environ["AMZN_BRAKET_SCRIPT_ENTRY_POINT"]
# the compression type as specified in the CreateJob request’s ScriptModeConfig
os.environ["AMZN_BRAKET_SCRIPT_COMPRESSION_TYPE"]
# the S3 location of the user’s script as specified in the CreateJob request’s
 ScriptModeConfig
os.environ["AMZN_BRAKET_SCRIPT_S3_URI"]
# the S3 location where the SDK would store the quantum task results by default for the
 job
os.environ["AMZN_BRAKET_TASK_RESULTS_S3_URI"]
# the S3 location where the job results would be stored, as specified in CreateJob
 request’s OutputDataConfig
os.environ["AMZN_BRAKET_JOB_RESULTS_S3_PATH"]
# the string that should be passed to CreateQuantumTask’s jobToken parameter for
 quantum tasks created in the job container
os.environ["AMZN_BRAKET_JOB_TOKEN"]
```

Funzioni di supporto

Amazon Braket offre diverse funzioni di supporto per semplificare le interazioni con gli input e gli output dei container. Queste funzioni di supporto verrebbero richiamate dall'interno dello script dell'algoritmo utilizzato per eseguire Hybrid Job. L'esempio seguente mostra come utilizzarle.

```
get_checkpoint_dir() # get the checkpoint directory
get_hyperparameters() # get the hyperparameters as strings
get_input_data_dir() # get the input data directory
get_job_device_arn() # get the device specified by the hybrid job
get_job_name() # get the name of the hybrid job.
get_results_dir() # get the path to a results directory
save_job_result() # save hybrid job results
save_job_checkpoint() # save a checkpoint
load_job_checkpoint() # load a previously saved checkpoint
```

Definite l'ambiente per lo script dell'algoritmo

Amazon Braket supporta tre ambienti definiti da contenitori per lo script dell'algoritmo:

- Un contenitore di base (predefinito, se non `image_uri` è specificato)
- Un contenitore con Tensorflow e PennyLane
- Un contenitore con e PyTorch PennyLane

La tabella seguente fornisce dettagli sui contenitori e sulle librerie che includono.

Contenitori Amazon Braket

Tipo	PennyLane con TensorFlow	PennyLane con PyTorch	Pennylane
Base	292282985366.dkr.ecr.us-east-1.amazonaws.com /:più recente amazon-braket-tensorflow-jobs	292282985366.dkr.ecr.us-west-2.amazonaws.com /:più recente amazon-braket-pytorch-jobs	292282985366.dkr.ecr.us-west-2.amazonaws.com /:più recente amazon-braket-base-jobs
Librerie ereditate	• awscli • numpy	• awscli • numpy	

Tipo	PennyLane con TensorFlow	PennyLane con PyTorch	Pennylane
	• pandas • scipy	• pandas • scipy	
Librerie aggiuntive	• amazon-braket-default-simulator • amazon-braket-pennylane-plugin • amazon-braket-schemas • amazon-braket-sdk • ipykernel • keras • matplotlib • reti x • babele aperte • PennyLane • protobuf • psi 4 • RSA • PennyLane-GPU Lightning • cuQuantum	• amazon-braket-default-simulator • amazon-braket-pennylane-plugin • amazon-braket-schemas • amazon-braket-sdk • kernel ipy • keras • matplotlib • reti x • babele aperte • PennyLane • protobuf • psi 4 • RSA • PennyLane-GPU Lightning • cuQuantum	• amazon-braket-default-simulator • amazon-braket-pennylane-plugin • amazon-braket-schemas • amazon-braket-sdk • awscli • boto3 • kernel ipy • matplotlib • reti x • numpy • babele aperte • pandas • PennyLane • protobuf • psi 4 • RSA • scipy

Puoi visualizzare e accedere alle definizioni dei contenitori open source su aws/. amazon-braket-containers Scegli il contenitore più adatto al tuo caso d'uso. Il contenitore deve essere nel Regione AWS da cui richiami il tuo lavoro ibrido. Si specifica l'immagine del contenitore quando si crea un lavoro ibrido aggiungendo uno dei tre argomenti seguenti alla `create(…)` chiamata nello script del lavoro ibrido. È possibile installare dipendenze aggiuntive nel contenitore scelto in fase di esecuzione (al costo dell'avvio o del runtime) perché Amazon I contenitori Braket dispongono di connettività Internet. L'esempio seguente si riferisce alla regione us-west-2.

- Immagine di base image_uri="292282985366.dkr. ecr.us-west-2.amazonaws.com /:1.0-cpu-py39-ubuntu22.04" amazon-braket-base-jobs
- Immagine Tensorflow image_uri="292282985366.dkr. ecr.us-east-1.amazonaws.com /:2.11.0-gpu-py39-cu112-ubuntu20.04" amazon-braket-tensorflow-jobs
- PyTorch image image_uri="292282985366.dkr. ecr.us-west-2.amazonaws.com /:1.13.1-gpu-py39-cu117-ubuntu20.04" amazon-braket-pytorch-jobs

`image-uris`Possono essere recuperati anche utilizzando la funzione in `retrieve_image()` Amazon Staffa SDK L'esempio seguente mostra come recuperarli da us-west-2 Regione AWS.

```
from braket.jobs.image_uris import retrieve_image, Framework

image_uri_base = retrieve_image(Framework.BASE, "us-west-2")
image_uri_tf = retrieve_image(Framework.PL_TENSORFLOW, "us-west-2")
image_uri_pytorch = retrieve_image(Framework.PL_PYTORCH, "us-west-2")
```

Utilizzo degli iperparametri

È possibile definire gli iperparametri necessari all'algoritmo, come il tasso di apprendimento o la dimensione dei passi, quando si crea un lavoro ibrido. I valori degli iperparametri vengono in genere utilizzati per controllare vari aspetti dell'algoritmo e spesso possono essere regolati per ottimizzare le prestazioni dell'algoritmo. Per utilizzare gli iperparametri in un processo ibrido Braket, è necessario specificarne i nomi e i valori in modo esplicito come dizionario. Nota che i valori devono essere del tipo di dati stringa. Specificate i valori degli iperparametri che desiderate testare durante la ricerca del set di valori ottimale. Il primo passaggio per utilizzare gli iperparametri consiste nell'impostare e definire gli iperparametri come dizionario, come illustrato nel codice seguente:

```
#defining the number of qubits used
n_qubits = 8
#defining the number of layers used
n_layers = 10
#defining the number of iterations used for your optimization algorithm
n_iterations = 10

hyperparams = {
    "n_qubits": n_qubits,
    "n_layers": n_layers,
    "n_iterations": n_iterations
```

```
}
```

Dovreste quindi passare gli iperparametri definiti nel frammento di codice sopra riportato, da utilizzare nell'algoritmo di vostra scelta, con qualcosa di simile al seguente:

```
import time
from braket.aws import AwsQuantumJob

#Name your job so that it can be later identified
job_name = f"qcbm-gaussian-training-{n_qubits}-{n_layers}-" + str(int(time.time()))

job = AwsQuantumJob.create(
    #Run this hybrid job on the SV1 simulator
    device="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
    #The directory or single file containing the code to run.
    source_module="qcbm",
    #The main script or function the job will run.
    entry_point="qcbm.qcbm_job:main",
    #Set the job_name
    job_name=job_name,
    #Set the hyperparameters
    hyperparameters=hyperparams,
    #Define the file that contains the input data
    input_data="data.npy", # or input_data=s3_path
    # wait_until_complete=False,
)
```

> **Note**
>
> Per saperne di più sui dati di input, consulta la sezione Input.

Gli iperparametri verrebbero quindi caricati nello script di lavoro ibrido utilizzando il seguente codice:

```
import json
import os

#Load the Hybrid Job hyperparameters
hp_file = os.environ["AMZN_BRAKET_HP_FILE"]
with open(hp_file, "r") as f:
    hyperparams = json.load(f)
```

> **Note**
>
> Per ulteriori informazioni su come passare informazioni come i dati di input e l'arn del dispositivo allo script di lavoro ibrido, consulta questa pagina github.

Un paio di guide molto utili per imparare a usare gli iperparametri sono fornite dai tutorial QAOAcon Amazon Braket Hybrid Jobs PennyLane e Quantum machine learning in Amazon Braket Hybrid Jobs.

Gestisci i tuoi circuiti con Open 3.0 QASM

Amazon Braket ora supporta Open QASM 3.0 per dispositivi e simulatori quantistici basati su gate. Questa guida per l'utente fornisce informazioni sul sottoinsieme di Open 3.0 supportato da Braket. QASM I clienti Braket ora possono scegliere di inviare circuiti Braket con SDKo fornendo direttamente stringhe Open QASM 3.0 a tutti i dispositivi basati su gate con Amazon Braket e Amazon Braket API Python. SDK

Gli argomenti di questa guida illustrano vari esempi di come completare le seguenti attività quantistiche.

- Crea e invia attività Open QASM quantum su diversi dispositivi Braket
- Accedi alle operazioni e ai tipi di risultati supportati
- Simula il rumore con Open QASM
- Usa la compilazione letterale con Open QASM
- Risolvi i problemi relativi a Open QASM

Questa guida fornisce anche un'introduzione ad alcune funzionalità specifiche dell'hardware che possono essere implementate con Open QASM 3.0 su Braket e collegamenti ad altre risorse.

In questa sezione:

- Che cos'è Open 3.0? QASM
- Quando usare Open 3.0 QASM
- Come funziona Open QASM 3.0
- Prerequisiti
- Quali QASM funzionalità di Open supporta Braket?

- Crea e invia un esempio di task quantistico Open QASM 3.0
- Support per Open QASM su diversi dispositivi Braket
- Simula il rumore con Open 3.0 QASM
- Qubit ricablaggio con Open 3.0 QASM
- Compilazione Verbatim con Open 3.0 QASM
- La console Braket
- Altre risorse
- Calcolo dei gradienti con Open 3.0 QASM
- Misurazione di qubit specifici con Open 3.0 QASM

Che cos'è Open 3.0? QASM

L'Open Quantum Assembly Language (OpenQASM) è una rappresentazione intermedia per le istruzioni quantistiche. Open QASM è un framework open source ed è ampiamente utilizzato per la specifica di programmi quantistici per dispositivi basati su gate. Con OpenQASM, gli utenti possono programmare le porte quantistiche e le operazioni di misurazione che costituiscono gli elementi costitutivi del calcolo quantistico. La versione precedente di Open QASM (2.0) veniva utilizzata da diverse librerie di programmazione quantistica per descrivere programmi semplici.

La nuova versione di Open QASM (3.0) estende la versione precedente per includere ulteriori funzionalità, come il controllo a livello di impulsi, il gate timing e il flusso di controllo classico per colmare il divario tra l'interfaccia utente finale e il linguaggio di descrizione dell'hardware. Dettagli e specifiche sulla versione corrente 3.0 sono disponibili nella Open 3.x Live Specification GitHub . QASM Lo sviluppo futuro QASM di Open è governato dal Technical Steering Committee di Open QASM 3.0, di cui AWS è membro insieme a IBM Microsoft e all'Università di Innsbruck.

Quando usare Open 3.0 QASM

Open QASM fornisce un framework espressivo per specificare programmi quantistici tramite controlli di basso livello che non sono specifici dell'architettura, il che lo rende adatto come rappresentazione su più dispositivi basati su gate. Il supporto di Braket per Open ne QASM promuove l'adozione come approccio coerente allo sviluppo di algoritmi quantistici basati su gate, riducendo la necessità per gli utenti di apprendere e gestire librerie in più framework.

Se disponi di librerie di programmi esistenti in Open QASM 3.0, puoi adattarle per l'uso con Braket anziché riscrivere completamente questi circuiti. I ricercatori e gli sviluppatori dovrebbero inoltre trarre

vantaggio da un numero crescente di librerie di terze parti disponibili con supporto per lo sviluppo di algoritmi in Open. QASM

Come funziona Open QASM 3.0

Il supporto per Open QASM 3.0 di Braket fornisce la parità di funzionalità con l'attuale Intermediate Representation. Ciò significa che tutto ciò che puoi fare oggi su dispositivi hardware e simulatori on-demand con Braket, puoi farlo con Open usando Braket QASM API. È possibile eseguire programmi Open QASM 3.0 fornendo direttamente QASM stringhe Open a tutti i dispositivi basati su gate in un modo simile a come i circuiti vengono attualmente forniti ai dispositivi su Braket. Gli utenti di Braket possono anche integrare librerie di terze parti che supportano Open 3.0. QASM Il resto di questa guida descrive in dettaglio come sviluppare QASM rappresentazioni Open da utilizzare con Braket.

Prerequisiti

Per utilizzare Open QASM 3.0 su Amazon Braket, devi avere la versione v1.8.0 di Amazon Braket Python Schemas e la versione 1.17.0 o successiva di Amazon Braket Python. SDK

Se sei un utente di Amazon Braket per la prima volta, devi abilitare Amazon Staffa. Per istruzioni, consulta Abilita Amazon Braket.

Quali QASM funzionalità di Open supporta Braket?

La sezione seguente elenca i tipi di dati, le istruzioni e le istruzioni pragma Open QASM 3.0 supportate da Braket.

In questa sezione:

- Tipi di dati Open supportati QASM
- Dichiarazioni Open supportate QASM
- Pragmi Braket Open QASM
- Supporto di funzionalità avanzate per Open QASM on the Local Simulator
- Operazioni e grammatica supportate con OpenPulse

Tipi di dati Open supportati QASM

I seguenti tipi di QASM dati aperti sono supportati da Amazon Braket.

- I numeri interi non negativi vengono utilizzati per gli indici di qubit (virtuali e fisici):

  - `cnot q[0], q[1];`
  - `h $0;`
- È possibile utilizzare numeri o costanti a virgola mobile per gli angoli di rotazione del cancello:
  - `rx(-0.314) $0;`
  - `rx(pi/4) $0;`

> **Note**
>
> pi è una costante incorporata in Open QASM e non può essere utilizzata come nome di parametro.

- Le matrici di numeri complessi (con la QASM `im` notazione Open per la parte immaginaria) sono consentite nei pragmi di tipo di risultato per la definizione di osservabili hermitiane generali e nei pragmi unitari:
  - `#pragma braket unitary [[0, -1im], [1im, 0]] q[0]`
  - `#pragma braket result expectation hermitian([[0, -1im], [1im, 0]]) q[0]`

Dichiarazioni Open supportate QASM

Le seguenti QASM dichiarazioni aperte sono supportate da Amazon Braket.

- `Header: OPENQASM 3;`
- Dichiarazioni di bit classiche:
  - `bit b1;`(equivalentemente,) `creg b1;`
  - `bit[10] b2;`(equivalentemente,) `creg b2[10];`
- Dichiarazioni Qubit:
  - `qubit b1;`(equivalentemente,) `qreg b1;`
  - `qubit[10] b2;`(equivalentemente,) `qreg b2[10];`
- Indicizzazione all'interno di array: `q[0]`
- Ingresso: `input float alpha;`
- specificazione fisica qubits: `$0`
- Porte e operazioni supportate su un dispositivo:

- `h $0;`
- `iswap q[0], q[1];`

> **Note**
>
> Le porte supportate da un dispositivo sono disponibili nelle proprietà del dispositivo per QASM le azioni Apri; non sono necessarie definizioni di porte per utilizzare queste porte.

- Dichiarazioni letterali. Al momento, non supportiamo la notazione della durata delle caselle. Porte native e fisiche qubits sono obbligatori nelle caselle letterali.

```
#pragma braket verbatim
box{
    rx(0.314) $0;
}
```

- Misurazione e assegnazione delle misurazioni su qubits o un tutto qubit registrare.
  - `measure $0;`
  - `measure q;`
  - `measure q[0];`
  - `b = measure q;`
  - `measure q # b;`

> **Note**
>
> pi è una costante incorporata in Open QASM e non può essere utilizzata come nome di parametro.

Pragmi Braket Open QASM

Le seguenti istruzioni di Open QASM pragma sono supportate da Amazon Braket.

- Pragmi del rumore

  - `#pragma braket noise bit_flip(0.2) q[0]`
  - `#pragma braket noise phase_flip(0.1) q[0]`
  - `#pragma braket noise pauli_channel`
- Pragmi letterali
  - `#pragma braket verbatim`
- Tipo di risultato: pragmi
  - Tipi di risultati invarianti di base:
    - Vettore di stato: `#pragma braket result state_vector`
    - Matrice di densità: `#pragma braket result density_matrix`
  - Pragmi di calcolo del gradiente:
    - Gradiente aggiunto: `#pragma braket result adjoint_gradient expectation(2.2 * x[0] @ x[1]) all`
  - Tipi di risultati di base Z:
    - Ampiezza: `#pragma braket result amplitude "01"`
    - Probabilità: `#pragma braket result probability q[0], q[1]`
  - Tipi di risultati ruotati di base
    - Aspettativa: `#pragma braket result expectation x(q[0]) @ y([q1])`
    - Varianza: `#pragma braket result variance hermitian([[0, -1im], [1im, 0]]) $0`
    - Esempio: `#pragma braket result sample h($1)`

> **Note**
>
> Open QASM 3.0 è retrocompatibile con Open QASM 2.0, quindi i programmi scritti utilizzando 2.0 possono essere eseguiti su Braket. Tuttavia, le funzionalità di Open QASM 3.0 supportate da Braket presentano alcune differenze di sintassi minori, come `qreg` vs e vs. `creg` `qubit` `bit` Esistono anche differenze nella sintassi di misurazione e queste devono essere supportate con la sintassi corretta.

Supporto di funzionalità avanzate per Open QASM on the Local Simulator

`LocalSimulator`Supporta QASM funzionalità Open avanzate che non sono offerte come parte dei simulatori di Braket o on-demand. QPU Il seguente elenco di funzionalità è supportato solo in: `LocalSimulator`

- Modificatori di gate
- Cancelli integrati aperti QASM
- Variabili classiche
- Operazioni classiche
- Cancelli personalizzati
- Controllo classico
- QASMfile
- Sottoroutine

Per esempi di ogni funzionalità avanzata, consultate questo taccuino di esempio. Per le QASM specifiche complete di Open, consultate il QASMsito Web Open.

Operazioni e grammatica supportate con OpenPulse

Tipi di OpenPulse dati supportati

Blocchi Cal:

```
cal {
    ...
}
```

Blocchi Defcal:

```
// 1 qubit
defcal x $0 {
...
}

// 1 qubit w. input parameters as constants
defcal my_rx(pi) $0 {
...
```

```
}

// 1 qubit w. input parameters as free parameters
defcal my_rz(angle theta) $0 {
...
}

// 2 qubit (above gate args are also valid)
defcal cz $1, $0 {
...
}
```

Cornici:

```
frame my_frame = newframe(port_0, 4.5e9, 0.0);
```

Forme d'onda:

```
// prebuilt
waveform my_waveform_1 = constant(1e-6, 1.0);

//arbitrary
waveform my_waveform_2 = {0.1 + 0.1im, 0.1 + 0.1im, 0.1, 0.1};
```

Esempio di calibrazione Custom Gate:

```
cal {
    waveform wf1 = constant(1e-6, 0.25);
}

defcal my_x $0 {
   play(wf1, q0_rf_frame);
}

defcal my_cz $1, $0 {
    barrier q0_q1_cz_frame, q0_rf_frame;
    play(q0_q1_cz_frame, wf1);
    delay[300ns] q0_rf_frame
    shift_phase(q0_rf_frame, 4.366186381749424);
    delay[300ns] q0_rf_frame;
    shift_phase(q0_rf_frame.phase, 5.916747563126659);
    barrier q0_q1_cz_frame, q0_rf_frame;
```

```
    shift_phase(q0_q1_cz_frame, 2.183093190874712);
}

bit[2] ro;
my_x $0;
my_cz $1,$0;
c[0] = measure $0;
```

Esempio di impulso arbitrario:

```
bit[2] ro;
cal {
    waveform wf1 = {0.1 + 0.1im, 0.1 + 0.1im, 0.1, 0.1};
    barrier q0_drive, q0_q1_cross_resonance;
    play(q0_q1_cross_resonance, wf1);
    delay[300ns] q0_drive;
    shift_phase(q0_drive, 4.366186381749424);
    delay[300dt] q0_drive;
   barrier q0_drive, q0_q1_cross_resonance;
   play(q0_q1_cross_resonance, wf1);
    ro[0] = capture_v0(r0_measure);
    ro[1] = capture_v0(r1_measure);
}
```

Crea e invia un esempio di task quantistico Open QASM 3.0

Puoi usare Amazon Braket PythonSDK, Boto3 o AWS CLI per inviare attività quantistiche Open QASM 3.0 a un dispositivo Amazon Braket.

In questa sezione:

- Un esempio di programma Open 3.0 QASM
- Usa Python SDK per creare attività quantistiche Open QASM 3.0
- Usa Boto3 per creare attività quantistiche Open 3.0 QASM
- Usa il AWS CLI per creare attività Open QASM 3.0

Un esempio di programma Open 3.0 QASM

Per creare un'attività Open QASM 3.0, è possibile iniziare con un semplice programma Open QASM 3.0 (ghz.qasm) che prepara uno stato come illustrato nell'esempio seguente. GHZ

```
// ghz.qasm
// Prepare a GHZ state
OPENQASM 3;

qubit[3] q;
bit[3] c;

h q[0];
cnot q[0], q[1];
cnot q[1], q[2];

c = measure q;
```

Usa Python SDK per creare attività quantistiche Open QASM 3.0

Puoi utilizzare Amazon Braket Python SDK per inviare questo programma a un dispositivo Amazon Braket con il codice seguente.

```
with open("ghz.qasm", "r") as ghz:
    ghz_qasm_string = ghz.read()

# import the device module
from braket.aws import AwsDevice
# choose the Rigetti device
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3")
from braket.ir.openqasm import Program

program = Program(source=ghz_qasm_string)
my_task = device.run(program)

# You can also specify an optional s3 bucket location and number of shots,
 # if you so choose, when running the program
s3_location = ("amazon-braket-my-bucket", "openqasm-tasks")
my_task = device.run(
    program,
    s3_location,
    shots=100,
)
```

Usa Boto3 per creare attività quantistiche Open 3.0 QASM

Puoi anche usare AWS Python SDK for Braket (Boto3) per creare le attività quantistiche utilizzando stringhe Open QASM 3.0, come mostrato nell'esempio seguente. Il seguente frammento di codice fa riferimento a ghz.qasm che prepara uno stato come mostrato sopra. GHZ

```
import boto3
import json

my_bucket = "amazon-braket-my-bucket"
s3_prefix = "openqasm-tasks"

with open("ghz.qasm") as f:
    source = f.read()

action = {
    "braketSchemaHeader": {
        "name": "braket.ir.openqasm.program",
        "version": "1"
    },
    "source": source
}
device_parameters = {}
device_arn = "arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3"
shots = 100

braket_client = boto3.client('braket', region_name='us-west-1')
rsp = braket_client.create_quantum_task(
    action=json.dumps(
        action
    ),
    deviceParameters=json.dumps(
        device_parameters
    ),
    deviceArn=device_arn,
    shots=shots,
    outputS3Bucket=my_bucket,
    outputS3KeyPrefix=s3_prefix,
)
```

Usa il AWS CLI per creare attività Open QASM 3.0

Il AWS Command Line Interface (CLI) può essere utilizzato anche per inviare programmi Open QASM 3.0, come illustrato nell'esempio seguente.

```
aws braket create-quantum-task \
    --region "us-west-1" \
    --device-arn "arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3" \
    --shots 100 \
    --output-s3-bucket "amazon-braket-my-bucket" \
    --output-s3-key-prefix "openqasm-tasks" \
    --action '{
        "braketSchemaHeader": {
            "name": "braket.ir.openqasm.program",
            "version": "1"
        },
        "source": $(cat ghz.qasm)
    }'
```

Support per Open QASM su diversi dispositivi Braket

Per i dispositivi che supportano Open QASM 3.0, il `action` campo supporta una nuova azione tramite la `GetDevice` risposta, come illustrato nell'esempio seguente per Rigetti e IonQ dispositivi.

```
//OpenQASM as available with the Rigetti device capabilities
{
    "braketSchemaHeader": {
        "name": "braket.device_schema.rigetti.rigetti_device_capabilities",
        "version": "1"
    },
    "service": {...},
    "action": {
        "braket.ir.jaqcd.program": {...},
        "braket.ir.openqasm.program": {
            "actionType": "braket.ir.openqasm.program",
            "version": [
                "1"
            ],
            ….
        }
    }
}
```

```
//OpenQASM as available with the IonQ device capabilities
{
    "braketSchemaHeader": {
        "name": "braket.device_schema.ionq.ionq_device_capabilities",
        "version": "1"
    },
    "service": {...},
    "action": {
        "braket.ir.jaqcd.program": {...},
        "braket.ir.openqasm.program": {
            "actionType": "braket.ir.openqasm.program",
            "version": [
                "1"
            ],
            ….
        }
    }
}
```

Per i dispositivi che supportano il controllo a impulsi, il `pulse` campo viene visualizzato nella `GetDevice` risposta. L'esempio seguente mostra questo `pulse` campo per Rigetti dispositivo.

```
// Rigetti
{
  "pulse": {
    "braketSchemaHeader": {
      "name": "braket.device_schema.pulse.pulse_device_action_properties",
      "version": "1"
    },
    "supportedQhpTemplateWaveforms": {
      "constant": {
        "functionName": "constant",
        "arguments": [
          {
            "name": "length",
            "type": "float",
            "optional": false
          },
          {
            "name": "iq",
            "type": "complex",
            "optional": false
```

```
        }
      ]
    },
    ...
  },
  "ports": {
    "q0_ff": {
      "portId": "q0_ff",
      "direction": "tx",
      "portType": "ff",
      "dt": 1e-9,
      "centerFrequencies": [
        375000000
      ]
    },
    ...
  },
  "supportedFunctions": {
    "shift_phase": {
      "functionName": "shift_phase",
      "arguments": [
        {
          "name": "frame",
          "type": "frame",
          "optional": false
        },
        {
          "name": "phase",
          "type": "float",
          "optional": false
        }
      ]
    },
   ...
  },
  "frames": {
    "q0_q1_cphase_frame": {
      "frameId": "q0_q1_cphase_frame",
      "portId": "q0_ff",
      "frequency": 462475694.24460185,
      "centerFrequency": 375000000,
      "phase": 0,
      "associatedGate": "cphase",
      "qubitMappings": [
```

```
          0,
          1
        ]
      },
      ...
    },
    "supportsLocalPulseElements": false,
    "supportsDynamicFrames": false,
    "supportsNonNativeGatesWithPulses": false,
    "validationParameters": {
      "MAX_SCALE": 4,
      "MAX_AMPLITUDE": 1,
      "PERMITTED_FREQUENCY_DIFFERENCE": 400000000
    }
  }
}
```

I campi precedenti descrivono in dettaglio quanto segue:

Porte:

Descrive le porte predefinite del dispositivo external (`extern`) dichiarate su oltre alle proprietà associate della porta specificata. QPU Tutte le porte elencate in questa struttura sono pre-dichiarate come identificatori validi all'interno del `OpenQASM 3.0` programma inviato dall'utente. Le proprietà aggiuntive di una porta includono:

- ID porta (portId)
  - Il nome della porta dichiarato come identificatore in Open QASM 3.0.
- Direzione (direzione)
  - La direzione del porto. Le porte di azionamento trasmettono gli impulsi (direzione «tx»), mentre le porte di misurazione ricevono gli impulsi (direzione «rx»).
- Tipo di porta () portType
  - Il tipo di azione di cui è responsabile questa porta (ad esempio, drive, capture o ff - fast-flux).
- Dt (dt)
  - Il tempo in secondi che rappresenta una singola fase temporale di campionamento sulla porta specificata.
- Mappature Qubit () qubitMappings
  - I qubit associati alla porta specificata.

- Frequenze centrali () centerFrequencies
  - Un elenco delle frequenze centrali associate per tutti i frame predichiarati o definiti dall'utente sulla porta. Per ulteriori informazioni, fate riferimento a Frames.
- QHPProprietà specifiche (qhpSpecificProperties)
  - Una mappa opzionale che descrive in dettaglio le proprietà esistenti sulla porta specifica diQHP.

Cornici:

Descrive i frame esterni predefiniti dichiarati sui QPU frame e le proprietà associate relative ai frame. Tutti i frame elencati in questa struttura sono pre-dichiarati come identificatori validi all'interno del `OpenQASM 3.0` programma inviato dall'utente. Le proprietà aggiuntive di un frame includono:

- ID del frame (frameId)
  - Il nome del frame dichiarato come identificatore in Open QASM 3.0.
- ID porta () portId
  - La porta hardware associata per il frame.
- Frequenza (frequenza)
  - La frequenza iniziale predefinita del frame.
- Frequenza centrale (centerFrequency)
  - Il centro della larghezza di banda della frequenza per il frame. In genere, i frame possono essere regolati solo su una determinata larghezza di banda attorno alla frequenza centrale. Di conseguenza, le regolazioni della frequenza devono rimanere entro un determinato delta della frequenza centrale. Puoi trovare il valore della larghezza di banda nei parametri di convalida.
- Fase (fase)
  - La fase iniziale predefinita del frame.
- Porta associata (associatedGate)
  - Le porte associate al frame specificato.
- Mappature Qubit () qubitMappings
  - I qubit associati al frame specificato.
- QHPProprietà specifiche () qhpSpecificProperties
  - Una mappa opzionale che descrive in dettaglio le proprietà esistenti relative al frame specifico diQHP.

SupportsDynamicFrames:

Descrive se un frame può essere dichiarato o meno in uno `cal` o più `defcal` blocchi tramite OpenPulse `newframe`funzione. Se questo è falso, all'interno del programma possono essere utilizzati solo i frame elencati nella struttura dei frame.

SupportedFunctions:

Descrive il OpenPulse funzioni supportate per il dispositivo oltre agli argomenti, ai tipi di argomento e ai tipi restituiti associati per le funzioni specificate. Per vedere esempi di utilizzo di OpenPulse funzioni, vedere le OpenPulsespecifiche. Al momento, Braket supporta:

- shift_phase
  - Sposta la fase di un frame in base a un valore specificato
- set_phase
  - Imposta la fase del frame sul valore specificato
- swap_phases
  - Scambia le fasi tra due frame.
- shift_frequency
  - Sposta la frequenza di un fotogramma in base a un valore specificato
- set_frequency
  - Imposta la frequenza del frame sul valore specificato
- giocare
  - Pianifica una forma d'onda
- capture_v0
  - Restituisce il valore di un frame di acquisizione in un registro di bit

SupportedQhpTemplateWaveforms:

Descrive le funzioni di forma d'onda predefinite disponibili sul dispositivo e gli argomenti e i tipi associati. Per impostazione predefinita, Braket Pulse offre routine di forme d'onda predefinite su tutti i dispositivi, che sono:

Costante

$$Constant(t, \tau, iq) = iq$$

`τ`è la lunghezza della forma d'onda ed è un numero complesso. `iq`

```
def constant(length, iq)
```

gaussiana

$$Gaussian(t, \tau, \sigma, A = 1, ZaE = 0) =$$
$$\frac{A}{1 - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)} \left[\exp\left(-\frac{1}{2}\left(\frac{t - \frac{\tau}{2}}{\sigma}\right)^2\right) - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)\right]$$

`τ`è la lunghezza della forma d'onda, è la larghezza della gaussiana e l'ampiezza. `σ` `A` Se è impostata `ZaE` su`True`, la gaussiana viene sfalsata e ridimensionata in modo che sia uguale a zero all'inizio e alla fine della forma d'onda e raggiunga il massimo. `A`

```
def gaussian(length, sigma, amplitude=1, zero_at_edges=False)
```

DRAGgaussiana

$$DRAG\_Gaussian(t, \tau, \sigma, \beta, A = 1, ZaE = 0) =$$
$$\frac{A}{1 - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)} \left(1 - i\beta\frac{t - \frac{\tau}{2}}{\sigma^2}\right) \left[\exp\left(-\frac{1}{2}\left(\frac{t - \frac{\tau}{2}}{\sigma}\right)^2\right) - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)\right]$$

`τ`è la lunghezza della forma d'onda, è la larghezza della gaussiana, σ è un parametro libero ed è l'ampiezza. `β` `A` Se è impostata `ZaE` su`True`, la Derivative Removal by Adiabatic Gate (DRAG) Gaussian viene sfalsata e ridimensionata in modo da essere uguale a zero all'inizio e alla fine della forma d'onda e la parte reale raggiunge il massimo. `A` Per ulteriori informazioni sulla DRAG forma d'onda, vedere il paper Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits.

```
def drag_gaussian(length, sigma, beta, amplitude=1, zero_at_edges=False)
```

Piazza Erf

$$\text{Erf\_Square}(t, L, W, \sigma, A = 1, ZaE = 0) =$$

$$A \times \frac{\text{erf}((t - t_1)/\sigma) + \text{erf}(-(t - t_2)/\sigma)}{2 \times \text{erf}(W/2\sigma))}$$

Dov'è la lunghezza, `L` `W` è la larghezza della forma d'onda, σ definisce la velocità con cui gli spigoli si alzano e si abbassano e, è l'ampiezza. t_1`=(L-W)/2` t_2`2=(L+W)/2` `A` Se è impostata `ZaE` su`True`, la gaussiana viene sfalsata e ridimensionata in modo che sia uguale a zero all'inizio e alla fine della forma d'onda e raggiunga il massimo. `A` L'equazione seguente è la versione ridimensionata della forma d'onda.

$$\text{Erf\_Square}(..., ZaE = 1) = (a \times \text{Erf\_Square}(..., ZaE = 0) - bA)/(a - b)$$

`b=erf(-`t_1`/σ)/2+erf(t` $_2$`/σ)/2` Dove e. `a=erf(W/2σ)`

```
def erf_square(length, width, sigma, amplitude=1, zero_at_edges=False)
```

SupportsLocalPulseElements:

Descrive se gli elementi a impulsi, come porte, frame e forme d'onda, possono essere definiti localmente in `defcal` blocchi. Se il valore è`false`, gli elementi devono essere definiti in `cal` blocchi.

SupportsNonNativeGatesWithPulses:

Descrive se è possibile o meno utilizzare porte non native in combinazione con programmi a impulsi. Ad esempio, non possiamo usare un gate non nativo come un `H` gate in un programma senza prima definire il gate through `defcal` per il qubit usato. Puoi trovare l'elenco delle `nativeGateSet` chiavi dei gate nativi sotto le funzionalità del dispositivo.

ValidationParameters:

Descrive i limiti di convalida degli elementi Pulse, tra cui:

- Valori di scala massima/ampiezza massima per le forme d'onda (arbitrari e predefiniti)
- Larghezza di banda di frequenza massima rispetto alla frequenza centrale fornita in Hz
- Lunghezza/durata minima dell'impulso in secondi
- Lunghezza/durata massima dell'impulso in secondi

Operazioni, risultati e tipi di risultati supportati con Open QASM

Per scoprire quali funzionalità Open QASM 3.0 sono supportate da ciascun dispositivo, potete fare riferimento alla `braket.ir.openqasm.program` chiave presente nel `action` campo sull'output

delle funzionalità del dispositivo. Ad esempio, le seguenti sono le operazioni supportate e i tipi di risultati disponibili per il simulatore Braket State Vector SV1.

```
...
  "action": {
    "braket.ir.jaqcd.program": {
      ...
    },
 "braket.ir.openqasm.program": {
      "version": [
        "1.0"
      ],
      "actionType": "braket.ir.openqasm.program",
      "supportedOperations": [
        "ccnot",
        "cnot",
        "cphaseshift",
        "cphaseshift00",
        "cphaseshift01",
        "cphaseshift10",
        "cswap",
        "cy",
        "cz",
        "h",
        "i",
        "iswap",
        "pswap",
        "phaseshift",
        "rx",
        "ry",
        "rz",
        "s",
        "si",
        "swap",
        "t",
        "ti",
        "v",
        "vi",
        "x",
        "xx",
        "xy",
        "y",
        "yy",
```

```
        "z",
        "zz"
      ],
      "supportedPragmas": [
        "braket_unitary_matrix"
      ],
      "forbiddenPragmas": [],
      "maximumQubitArrays": 1,
      "maximumClassicalArrays": 1,
      "forbiddenArrayOperations": [
        "concatenation",
        "negativeIndex",
        "range",
        "rangeWithStep",
        "slicing",
        "selection"
      ],
      "requiresAllQubitsMeasurement": true,
      "supportsPhysicalQubits": false,
      "requiresContiguousQubitIndices": true,
      "disabledQubitRewiringSupported": false,
      "supportedResultTypes": [
        {
          "name": "Sample",
          "observables": [
            "x",
            "y",
            "z",
            "h",
            "i",
            "hermitian"
          ],
          "minShots": 1,
          "maxShots": 100000
        },
        {
          "name": "Expectation",
          "observables": [
            "x",
            "y",
            "z",
            "h",
            "i",
            "hermitian"
```

```
          ],
          "minShots": 0,
          "maxShots": 100000
        },
        {
          "name": "Variance",
          "observables": [
            "x",
            "y",
            "z",
            "h",
            "i",
            "hermitian"
          ],
          "minShots": 0,
          "maxShots": 100000
        },
        {
          "name": "Probability",
          "minShots": 1,
          "maxShots": 100000
        },
        {
          "name": "Amplitude",
          "minShots": 0,
          "maxShots": 0
        }
        {
          "name": "AdjointGradient",
          "minShots": 0,
          "maxShots": 0
        }
      ]
    }
  },
...
```

Simula il rumore con Open 3.0 QASM

Per simulare il rumore con OpenQASM3, utilizzate le istruzioni pragma per aggiungere operatori di rumore. Ad esempio, per simulare la versione rumorosa del [GHZprogramma fornita in precedenza, è possibile inviare il seguente programma](#) Open. QASM

```
// ghz.qasm
// Prepare a GHZ state
OPENQASM 3;

qubit[3] q;
bit[3] c;

h q[0];
#pragma braket noise depolarizing(0.75) q[0] cnot q[0], q[1];
#pragma braket noise depolarizing(0.75) q[0]
#pragma braket noise depolarizing(0.75) q[1] cnot q[1], q[2];
#pragma braket noise depolarizing(0.75) q[0]
#pragma braket noise depolarizing(0.75) q[1]

c = measure q;
```

Le specifiche per tutti gli operatori Pragma Noise supportati sono fornite nell'elenco seguente.

```
#pragma braket noise bit_flip(<float in [0,1/2]>) <qubit>
#pragma braket noise phase_flip(<float in [0,1/2]>) <qubit>
#pragma braket noise pauli_channel(<float>, <float>, <float>)  <qubit>
#pragma braket noise depolarizing(<float in [0,3/4]>) <qubit>
#pragma braket noise two_qubit_depolarizing(<float in [0,15/16]>) <qubit>, <qubit>
#pragma braket noise two_qubit_dephasing(<float in [0,3/4]>) <qubit>, <qubit>
#pragma braket noise amplitude_damping(<float in [0,1]>) <qubit>
#pragma braket noise generalized_amplitude_damping(<float in [0,1]> <float in [0,1]>)
 <qubit>
#pragma braket noise phase_damping(<float in [0,1]>) <qubit>
#pragma braket noise kraus([[<complex m0_00>, ], ...], [[<complex m1_00>, ], ...], ...)
 <qubit>[, <qubit>]     // maximum of 2 qubits and maximum of 4 matrices for 1 qubit,
 16 for 2
```

Operatore Kraus

Per generare un operatore Kraus, puoi scorrere un elenco di matrici, stampando ogni elemento della matrice come espressione complessa.

Quando usi gli operatori Kraus, ricorda quanto segue:

- Il numero di qubits non deve essere superiore a 2. La definizione corrente negli schemi stabilisce questo limite.

- La lunghezza dell'elenco degli argomenti deve essere un multiplo di 8. Ciò significa che deve essere composto solo da matrici 2x2.
- La lunghezza totale non supera 2 matrici 2*num_qubits. Ciò significa 4 matrici per 1 qubit e 16 per 2 qubits.
- Tutte le matrici fornite sono completamente a conservazione positiva delle tracce (). CPTP
- Il prodotto degli operatori Kraus con i loro coniugati di trasposizione deve sommarsi a una matrice di identità.

Qubit ricablaggio con Open 3.0 QASM

Amazon Braket supporta il sistema fisico qubit notazione all'interno di Open on QASM Rigetti dispositivi (per saperne di più consulta questa pagina). Quando si utilizza un dispositivo fisico qubits con l'ingenua strategia di ricablaggio, assicurati che qubits sono collegati sul dispositivo selezionato. In alternativa, se qubit vengono invece utilizzati i registri, la strategia di PARTIAL ricablaggio è abilitata di default su Rigetti dispositivi.

```
// ghz.qasm
// Prepare a GHZ state
OPENQASM 3;

h $0;
cnot $0, $1;
cnot $1, $2;

measure $0;
measure $1;
measure $2;
```

Compilazione Verbatim con Open 3.0 QASM

Quando si esegue un circuito quantistico su computer quantistici da Rigettie IonQ, è possibile fare in modo che il compilatore esegua i circuiti esattamente come definito, senza alcuna modifica. Questa funzionalità è nota come compilazione letterale. Con i dispositivi Rigetti, puoi specificare con precisione cosa deve essere preservato: un intero circuito o solo parti specifiche di esso. Per preservare solo parti specifiche di un circuito, è necessario utilizzare porte native all'interno delle regioni protette. Attualmente IonQ supporta solo la compilazione letterale per l'intero circuito, quindi ogni istruzione del circuito deve essere racchiusa in una casella letterale.

Con OpenQASM, è possibile specificare un pragma letterale attorno a una casella di codice che non viene modificata e non ottimizzata dalla routine di compilazione di basso livello dell'hardware. Il seguente esempio di codice mostra come utilizzare. `#pragma braket verbatim`

```
OPENQASM 3;

bit[2] c;

#pragma braket verbatim
box{
    rx(0.314159) $0;
    rz(0.628318) $0, $1;
    cz $0, $1;
}

c[0] = measure $0;
c[1] = measure $1;
```

Per ulteriori informazioni sulla compilazione letterale, consultate il taccuino di esempio della compilazione Verbatim.

La console Braket

Le attività Open QASM 3.0 sono disponibili e possono essere gestite all'interno della console Amazon Braket. Sulla console, hai la stessa esperienza nell'invio di attività quantistiche in Open QASM 3.0 come avevi nell'invio di attività quantistiche esistenti.

Altre risorse

Open QASM è disponibile in tutte le regioni Amazon Braket.

Per un esempio di notebook per iniziare a usare Open QASM su Amazon Braket, consulta Braket Tutorials. GitHub

Calcolo dei gradienti con Open 3.0 QASM

Amazon Braket supporta il calcolo dei gradienti su simulatori on-demand e locali in modalità `shots=0` (esatta) utilizzando il metodo di differenziazione aggiuntiva. Puoi fornire il pragma appropriato per specificare il gradiente che desideri calcolare, come mostrato nell'esempio seguente.

```
OPENQASM 3.0;
```

```
input float alpha;

bit[2] b;
qubit[2] q;

h q[0];
h q[1];
rx(alpha) q[0];
rx(alpha) q[1];
b[0] = measure q[0];
b[1] = measure q[1];

#pragma braket result adjoint_gradient h(q[0]) @ i(q[1]) alpha
```

Invece di elencare tutti i parametri singolarmente, potete anche specificarli nel pragma. `all` Questo calcola il gradiente rispetto a tutti i parametri elencati. `input` Questo può essere utile quando il numero di parametri è molto elevato. In questo caso, il pragma sarà simile al seguente esempio.

```
#pragma braket result adjoint_gradient h(q[0]) @ i(q[1]) all
```

Sono supportati tutti i tipi di osservabili, inclusi i singoli operatori, i prodotti tensoriali, gli osservabili hermitiani e. `Sum` L'operatore che si desidera utilizzare per calcolare il gradiente deve essere racchiuso nell'`expectation()`incapsulatore e devono essere specificati i qubit su cui agisce ciascun termine.

Misurazione di qubit specifici con Open 3.0 QASM

Il simulatore vettoriale dello stato locale e il simulatore di matrice a densità locale supportano l'invio OpenQASM programmi in cui è possibile misurare un sottoinsieme dei qubit del circuito. Questa operazione viene spesso denominata misurazione parziale. Ad esempio, nel codice seguente è possibile creare un circuito a due qubit e misurare solo il primo qubit.

```
partial_measure_qasm = """
OPENQASM 3.0;
bit[1] b;
qubit[2] q;
h q[0];
cnot q[0], q[1];
b[0] = measure q[0];
"""
```

Ci sono due qubit, `q[0]` `q[1]` ma qui stiamo misurando solo il qubit 0:. `b[0] = measure q[0]` Ora, esegui quanto segue sul simulatore vettoriale dello stato locale.

```
from braket.devices import LocalSimulator

local_sim = LocalSimulator()
partial_measure_local_sim_task =
 local_sim.run(OpenQASMProgram(source=partial_measure_qasm), shots = 10)
partial_measure_local_sim_result = partial_measure_local_sim_task.result()
print(partial_measure_local_sim_result.measurement_counts)
print("Measured qubits: ", partial_measure_local_sim_result.measured_qubits)
```

È possibile verificare se un dispositivo supporta la misurazione parziale ispezionando il `requiresAllQubitsMeasurement` campo nelle sue proprietà di azione; in tal caso`False`, è supportata la misurazione parziale.

```
AwsDevice(Devices.Rigetti.AspenM3).properties.action['braket.ir.openqasm.program'].requiresAllQ
```

Qui `requiresAllQubitsMeasurement` è`False`, il che indica che non tutti i qubit devono essere misurati.

Esplora le funzionalità sperimentali

Per potenziare i carichi di lavoro di ricerca, è importante accedere rapidamente a nuove funzionalità innovative. Con Braket Direct, puoi richiedere l'accesso alle funzionalità sperimentali disponibili, come i nuovi dispositivi quantistici con disponibilità limitata, direttamente nella console Braket.

Alcune funzionalità sperimentali funzionano al di fuori delle specifiche standard dei dispositivi e richiedono una guida pratica personalizzata in base al caso d'uso. Per garantire la corretta configurazione dei carichi di lavoro, l'accesso è disponibile su richiesta tramite Braket Direct.

In questa sezione:

- Accesso solo su prenotazione a IonQ Forte
- Accesso al detuning locale sull'Aquila QuEra
- Accesso alle geometrie alte dell'Aquila QuEra
- Accesso a geometrie strette su Aquila QuEra

Accesso solo su prenotazione a IonQ Forte

Con Braket Direct, ottieni l'accesso solo su prenotazione a IonQ Forte. QPU A causa della sua disponibilità limitata, questo dispositivo è disponibile solo tramite Braket Direct.

Per saperne di più e richiedere l'accesso a IonQ Forte, segui questi passaggi:

1. Apri la console Amazon Braket.
2. Seleziona Braket Direct nel menu a sinistra, quindi in Experimental Capabilities vai a IonQ Forte. Scegli Visualizza dispositivo.
3. Nella pagina dei dettagli del dispositivo Forte, in Riepilogo scegli Prenota dispositivo.
4. Fornisci le tue informazioni di contatto, tra cui nome ed email. Fornisci un indirizzo email valido che controlli regolarmente.
5. Nella sezione Comunicaci il tuo carico di lavoro, fornisci dettagli sul carico di lavoro da eseguire utilizzando la tua prenotazione, come la durata della prenotazione desiderata, i vincoli pertinenti o la pianificazione desiderata.
6. (Facoltativo) Se sei interessato a contattare un esperto di Braket per una sessione di preparazione alla prenotazione dopo la conferma della prenotazione, seleziona Sono interessato a una sessione di preparazione.

Una volta inviato il modulo, il team di Braket ti contatterà per indicarti i passaggi successivi.

> **Note**
>
> A causa della disponibilità limitata dei dispositivi, l'accesso a Forte è limitato. Contattaci per saperne di più.

Accesso al detuning locale sull'Aquila QuEra

Con Braket Direct, puoi richiedere l'accesso per controllare la desintonizzazione locale durante la programmazione su QuEra Aquila QPU. Con questa funzionalità, puoi regolare l'impatto del campo di guida su ogni qubit specifico.

Per saperne di più e richiedere l'accesso a questa funzionalità, segui questi passaggi:

1. Apri la console Amazon Braket.

2. Seleziona Braket Direct nel menu a sinistra, quindi in Experimental Capabilities vai a QuEra Aquila: detoning locale. Scegli Ottieni accesso.
3. Fornisci le tue informazioni di contatto, inclusi nome ed e-mail. Fornisci un indirizzo email valido che controlli regolarmente.
4. In Comunicaci il tuo carico di lavoro, fornisci dettagli sul carico di lavoro e su dove intendi utilizzare questa funzionalità.

Accesso alle geometrie alte dell'Aquila QuEra

Con Braket Direct, puoi richiedere l'accesso a geometrie espanse durante la programmazione su QuEra Aquila QPU. Con questa funzionalità, è possibile sperimentare oltre le funzionalità standard dei dispositivi e specificare geometrie con una maggiore altezza del reticolo.

Per saperne di più e richiedere l'accesso a questa funzionalità, segui questi passaggi:

1. Apri la console Amazon Braket.
2. Seleziona Braket Direct nel menu a sinistra, quindi in Experimental Capabilities vai a QuEra Aquila - tall geometries. Scegli Ottieni l'accesso.
3. Fornisci le tue informazioni di contatto, inclusi nome ed e-mail. Fornisci un indirizzo email valido che controlli regolarmente.
4. In Comunicaci il tuo carico di lavoro, fornisci dettagli sul carico di lavoro e su dove intendi utilizzare questa funzionalità.

Accesso a geometrie strette su Aquila QuEra

Con Braket Direct, puoi richiedere l'accesso a geometrie espanse durante la programmazione su QuEra Aquila QPU. Con questa funzionalità, è possibile sperimentare oltre le funzionalità standard dei dispositivi e disporre file reticolari con spaziature verticali più strette.

Per saperne di più e richiedere l'accesso a questa funzionalità, segui questi passaggi:

1. Apri la console Amazon Braket.
2. Seleziona Braket Direct nel menu a sinistra, quindi in Experimental Capabilities vai a QuEra Aquila - tall geometries. Scegli Ottieni l'accesso.
3. Fornisci le tue informazioni di contatto, inclusi nome ed e-mail. Fornisci un indirizzo email valido che controlli regolarmente.

4. In Comunicaci il tuo carico di lavoro, fornisci dettagli sul carico di lavoro e su dove intendi utilizzare questa funzionalità.

Controllo a impulsi su Amazon Braket

Gli impulsi sono i segnali analogici che controllano i qubit in un computer quantistico. Con alcuni dispositivi su Amazon Braket, puoi accedere alla funzione di controllo degli impulsi per inviare circuiti utilizzando impulsi. Puoi accedere al controllo degli impulsi tramite BraketSDK, utilizzando Open QASM 3.0 o direttamente tramite Braket. APIs Innanzitutto, introduciamo alcuni concetti chiave per il controllo degli impulsi in Braket.

In questa sezione:

- Frames (Fotogrammi)
- Porte
- Forme d'onda
- Ruoli dei frame e delle porte
- Ciao Pulse
- Accesso ai gate nativi tramite impulsi

Frames (Fotogrammi)

Un frame è un'astrazione software che funge sia da orologio all'interno del programma quantistico che da fase. L'ora dell'orologio viene incrementata a ogni utilizzo e un segnale portante statico definito da una frequenza. Quando si trasmettono segnali al qubit, un frame determina la frequenza portante del qubit, l'offset di fase e l'ora in cui viene emesso l'inviluppo della forma d'onda. In Braket Pulse, la costruzione dei frame dipende dal dispositivo, dalla frequenza e dalla fase. A seconda del dispositivo, potete scegliere un frame predefinito o creare un'istanza di nuovi frame fornendo una porta.

```
from braket.pulse import Frame
# predefined frame from a device
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-2")
drive_frame = device.frames["Transmon_5_charge_tx_frame"]

# create a custom frame
readout_frame = Frame(name="r0_measure", port=port0, frequency=5e9, phase=0)
```

Porte

Una porta è un'astrazione software che rappresenta qualsiasi componente hardware di input/output che controlla i qubit. Aiuta i fornitori di hardware a fornire un'interfaccia con cui gli utenti possono interagire per manipolare e osservare i qubit. Le porte sono caratterizzate da una singola stringa che rappresenta il nome del connettore. Questa stringa mostra anche un incremento di tempo minimo che specifica con quanta precisione possiamo definire le forme d'onda.

```
from braket.pulse import Port
Port0 = Port("channel_0", dt=1e-9)
```

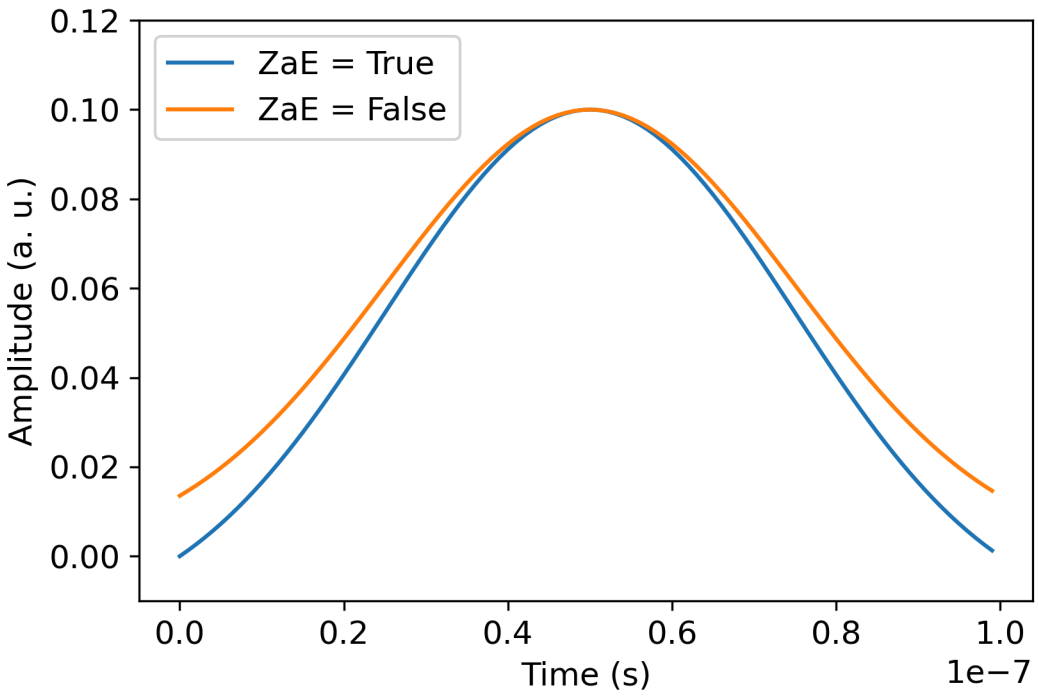
Forme d'onda

Una forma d'onda è un involucro dipendente dal tempo che possiamo usare per emettere segnali su una porta di uscita o acquisire segnali attraverso una porta di ingresso. È possibile specificare le forme d'onda direttamente tramite un elenco di numeri complessi o utilizzando un modello di forma d'onda per generare un elenco fornito dal fornitore di hardware.

```
from braket.pulse import ArbitraryWaveform, ConstantWaveform
cst_wfm = ConstantWaveform(length=1e-7, iq=0.1)
arb_wf = ArbitraryWaveform(amplitudes=np.linspace(0, 100))
```

Braket Pulse fornisce una libreria standard di forme d'onda, tra cui una forma d'onda costante, una forma d'onda gaussiana e una forma d'onda Derivative Removal by Adiabatic Gate (). DRAG È possibile recuperare i dati della forma d'onda tramite la funzione per disegnare la forma della forma d'onda, come mostrato nell'esempio seguente. `sample`

```
gaussian_waveform = GaussianWaveform(1e-7, 25e-9, 0.1)
x = np.arange(0, gaussian_waveform.length, drive_frame.port.dt)
plt.plot(x, gaussian_waveform.sample(drive_frame.port.dt))
```



L'immagine precedente mostra le forme d'onda gaussiane create da. `GaussianWaveform` Abbiamo scelto una lunghezza dell'impulso di 100 ns, una larghezza di 25 ns e un'ampiezza di 0,1 (unità arbitrarie). Le forme d'onda sono centrate nella finestra degli impulsi. `GaussianWaveform`accetta un argomento booleano `zero_at_edges` (zAe nella legenda). Se impostato su`True`, questo argomento compensa la forma d'onda gaussiana in modo tale che i punti in t=0 e t= `length` siano a zero e ne ridimensiona l'ampiezza in modo che il valore massimo corrisponda all'argomento. `amplitude`

Ora che abbiamo trattato i concetti di base per l'accesso a livello di impulsi, vedremo ora come costruire un circuito usando porte e impulsi.

Ruoli dei frame e delle porte

Questa sezione descrive i frame e le porte predefiniti disponibili per ogni dispositivo. Discuteremo anche brevemente i meccanismi coinvolti nella riproduzione degli impulsi su determinati frame.

Cornici Rigetti

Rigetti i dispositivi supportano frame predefiniti la cui frequenza e fase sono calibrate per essere in risonanza con il qubit associato. La convenzione di denominazione è `q{i}[_q{j}]_{role}_frame` dove `{i}` si riferisce al primo numero di qubit, al secondo numero di

qubit nel caso in cui il frame serva ad attivare un'interazione a due qubit e `{j}` si riferisce al ruolo del frame. `{role}` I ruoli sono i seguenti:

- `rf`è il frame per guidare la transizione 0-1 del qubit. Gli impulsi vengono trasmessi come segnali transitori a microonde di frequenza e fase precedentemente forniti tramite le funzioni and. `set shift` L'ampiezza del segnale dipendente dal tempo è data dalla forma d'onda riprodotta sul frame. Il frame collega un'interazione a singolo qubit, fuori diagonale. Per ulteriori informazioni, vedere Krantz et al. e Rahamim et al. .
- `rf_f12`è simile a `rf` e i suoi parametri sono mirati alla transizione 1-2.
- `ro_rx`viene utilizzato per ottenere una lettura dispersiva del qubit attraverso una guida d'onda complanare accoppiata. La frequenza, la fase e il set completo di parametri per la forma d'onda di lettura sono precalibrati. Attualmente viene utilizzato tramite`capture_v0`, che non richiede alcun argomento oltre all'identificatore del frame.
- `ro_tx`serve per trasmettere segnali dal risonatore. Attualmente è inutilizzato.
- `cz`è un frame calibrato per abilitare il gate a due `cz` qubit. Come tutti i frame associati a una `ff` porta, attiva un'interazione interconnessa attraverso la linea di flusso modulando il qubit sintonizzabile della coppia in base alla risonanza con la porta adiacente. Per ulteriori informazioni sul meccanismo di entangling, vedere Reagor et al. , Caldwell e altri , e Didier et al. .
- `cphase`è un frame calibrato per abilitare il `cphaseshift` gate a due qubit ed è collegato a una porta. `ff` Per ulteriori informazioni sul meccanismo di aggrovigliamento, consultate la descrizione del frame. `cz`
- `xy`è un frame calibrato per abilitare le porte XY (θ) a due qubit ed è collegato a una porta. `ff` Per ulteriori informazioni sul meccanismo di entangling e su come realizzare le porte XY, vedere la descrizione del `cz` frame e Abrams et al. .

Man mano che i frame basati sulla `ff` porta spostano la frequenza del qubit sintonizzabile, tutti gli altri frame di pilotaggio relativi al qubit verranno defasati di un valore correlato all'ampiezza e alla durata dello spostamento di frequenza. Di conseguenza, è necessario compensare questo effetto aggiungendo uno sfasamento corrispondente ai frame dei qubit adiacenti.

Porte

Il Rigetti i dispositivi forniscono un elenco di porte che è possibile ispezionare tramite le funzionalità del dispositivo. I nomi delle porte seguono la convenzione `q{i}_{type}` in cui `{i}` si riferisce al numero di qubit e `{type}` si riferisce al tipo di porta. Nota che non tutti i qubit hanno un set completo di porte. I tipi di porte sono i seguenti:

- `rf`rappresenta l'interfaccia principale per guidare la transizione a qubit singolo. È associato ai frame `rf` `erf_f12`. È accoppiato in modo capacitivo al qubit, permettendo la guida a microonde nell'intervallo dei gigahertz.
- `ro_tx`serve a trasmettere segnali al risonatore di lettura accoppiato capacitivamente al qubit. La trasmissione del segnale di lettura è moltiplicata otto volte per ottagono.
- `ro_rx`serve a ricevere segnali dal risonatore di lettura accoppiato al qubit.
- `ff`rappresenta la linea a flusso rapido accoppiata induttivamente al qubit. Possiamo usarla per regolare la frequenza del transmon. Solo i qubit progettati per essere altamente sintonizzabili hanno una porta. `ff` Questa porta serve ad attivare l'interazione qubit-qubit in quanto esiste un accoppiamento capacitivo statico tra ogni coppia di transmon adiacenti.

Per ulteriori informazioni sull'architettura, vedere Valery et al. .

Ciao Pulse

In questa sezione, imparerai come caratterizzare e costruire una singola porta qubit direttamente usando pulse su un Rigetti dispositivo. L'applicazione di un campo elettromagnetico a un qubit porta all'oscillazione Rabi, che commuta i qubit tra lo stato 0 e lo stato 1. Con la lunghezza e la fase dell'impulso calibrate, l'oscillazione Rabi può calcolare una singola porta qubit. Qui determineremo la lunghezza ottimale dell'impulso per misurare un impulso pi/2, un blocco elementare utilizzato per costruire sequenze di impulsi più complesse.

Innanzitutto, per creare una sequenza di impulsi, importate la classe. `PulseSequence`

```
from braket.aws import AwsDevice
from braket.circuits import FreeParameter
from braket.devices import Devices
from braket.pulse import PulseSequence, GaussianWaveform

import numpy as np
```

Quindi, crea un'istanza di un nuovo dispositivo Braket utilizzando il Amazon Resource Name (ARN) del. QPU Il seguente blocco di codice utilizza Rigetti Ankaa-2.

```
device = AwsDevice(Devices.Rigetti.Ankaa2)
```

La seguente sequenza di impulsi include due componenti: riproduzione di una forma d'onda e misurazione di un qubit. La sequenza di impulsi può in genere essere applicata ai fotogrammi. Con

alcune eccezioni come la barriera e il ritardo, che possono essere applicati ai qubit. Prima di costruire la sequenza di impulsi è necessario recuperare i frame disponibili. Il frame di azionamento viene utilizzato per applicare l'impulso per l'oscillazione Rabi e il frame di lettura serve per misurare lo stato del qubit. Questo esempio utilizza i frame del qubit 25. Per ulteriori informazioni sui frame, vedere Ruoli dei frame e delle porte.

```
drive_frame = device.frames["Transmon_25_charge_tx"]
readout_frame = device.frames["Transmon_25_readout_rx"]
```

Ora, create la forma d'onda che verrà riprodotta nel frame del drive. L'obiettivo è caratterizzare il comportamento dei qubit per diverse lunghezze di impulso. Suonerai ogni volta una forma d'onda con lunghezze diverse. Invece di istanziare ogni volta una nuova forma d'onda, utilizzate il parametro libero supportato da Braket nella sequenza di impulsi. È possibile creare la forma d'onda e la sequenza di impulsi una volta con parametri liberi, quindi eseguire la stessa sequenza di impulsi con valori di input diversi.

```
waveform = GaussianWaveform(FreeParameter("length"), FreeParameter("length") * 0.25,
 0.2, False)
```

Infine, mettili insieme come sequenza di impulsi. Nella sequenza di impulsi, `play` riproduce la forma d'onda specificata sul frame di azionamento e quindi `capture_v0` misura lo stato a partire dal riquadro di lettura.

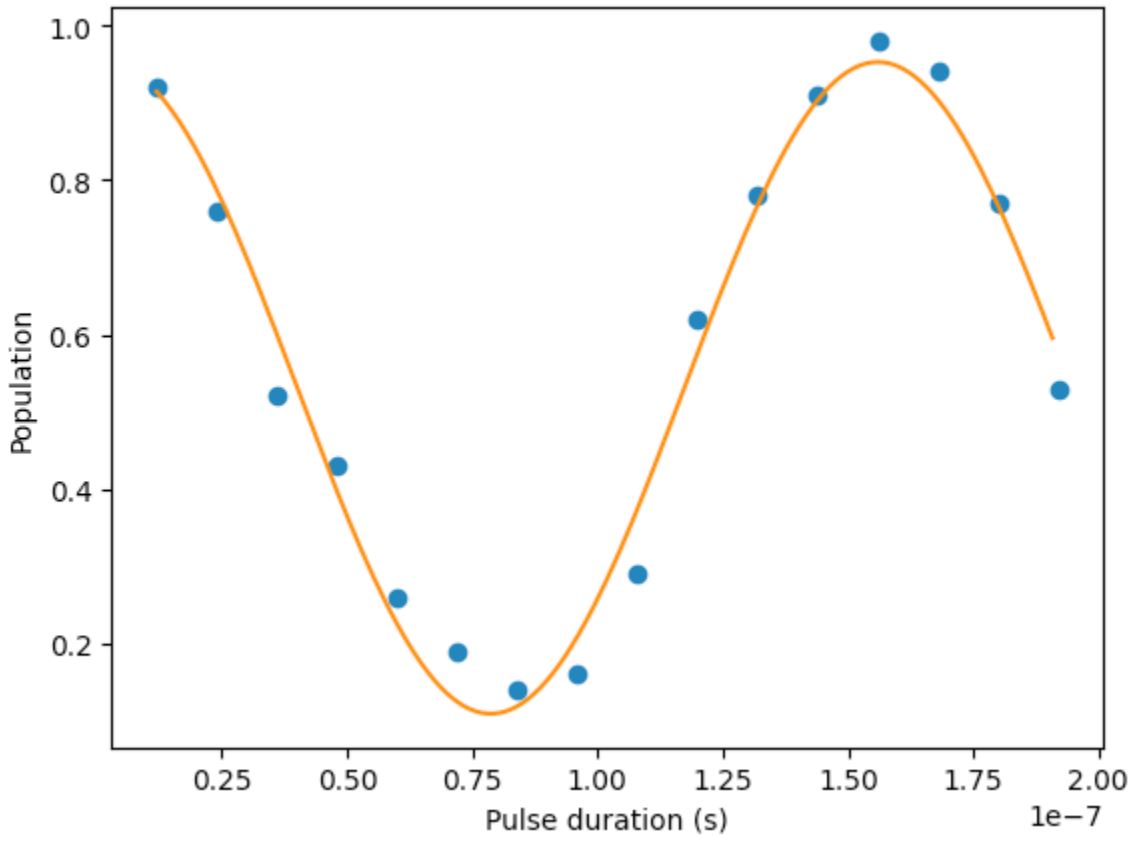
```
pulse_sequence = (
    PulseSequence()
    .play(drive_frame, waveform)
    .capture_v0(readout_frame)
)
```

Esegue la scansione su un intervallo di lunghezza degli impulsi e li invia a. QPU

```
start_length=12e-9
end_length=2e-7
lengths = np.arange(start_length, end_length, 12e-9)

tasks = [
    device.run(pulse_sequence, shots=100, inputs={"length": length})
    for length in lengths
]
```

```
probability_of_zero = [
    task.result().measurement_counts['0']/N_shots
    for task in tasks
]
```

Le statistiche della misurazione del qubit mostrano la dinamica oscillatoria del qubit che oscilla tra lo stato 0 e lo stato 1. Dai dati di misurazione, è possibile estrarre la frequenza Rabi e regolare con precisione la lunghezza dell'impulso per implementare una particolare porta da 1 qubit. Ad esempio, dai dati riportati nella figura seguente, la periodicità è di circa 154 ns. Quindi una porta di rotazione pi/2 corrisponderebbe alla sequenza di impulsi con lunghezza = 38,5 ns.



Hello Pulse sta usando OpenPulse

OpenPulseè un linguaggio per specificare il controllo a livello di impulsi di un dispositivo quantistico generale e fa parte delle specifiche Open 3.0. QASM Supporti Amazon Braket OpenPulse per programmare direttamente gli impulsi utilizzando la rappresentazione Open QASM 3.0.

Usi della staffa OpenPulse come rappresentazione intermedia sottostante per esprimere gli impulsi nelle istruzioni native. OpenPulse supporta l'aggiunta di calibrazioni delle istruzioni sotto forma di dichiarazioni `defcal` (abbreviazione di «define calibration»). Con queste dichiarazioni, è possibile

specificare un'implementazione di un'istruzione gate all'interno di una grammatica di controllo di livello inferiore.

È possibile visualizzare il OpenPulse programma di un Braket `PulseSequence` utilizzando il seguente comando.

```
print(pulse_sequence.to_ir())
```

Puoi anche costruire direttamente un OpenPulse programma.

```
from braket.ir.openqasm import Program

openpulse_script = """
OPENQASM 3.0;
cal {
    bit[1] psb;
    waveform my_waveform = gaussian(12.0ns, 3.0ns, 0.2, false);
    play(Transmon_25_charge_tx, my_waveform);
    psb[0] = capture_v0(Transmon_25_readout_rx);
}
"""
```

Crea un `Program` oggetto con il tuo script. Quindi, invia il programma a unQPU.

```
from braket.aws import AwsDevice
from braket.devices import Devices
from braket.ir.openqasm import Program

program = Program(source=qasm_script)

device = AwsDevice(Devices.Rigetti.Ankaa2)
task = device.run(program, shots=100)
```

Accesso ai gate nativi tramite impulsi

I ricercatori spesso hanno bisogno di sapere esattamente come le porte native supportate da un particolare dispositivo vengono implementate come impulsi. QPU Le sequenze di impulsi vengono calibrate con cura dai fornitori di hardware, ma accedervi offre ai ricercatori l'opportunità di progettare gate migliori o di esplorare protocolli per la mitigazione degli errori, come l'estrapolazione a zero rumore mediante l'allungamento degli impulsi di porte specifiche.

Amazon Braket supporta l'accesso programmatico ai gate nativi di Rigetti.

```
import math
from braket.aws import AwsDevice
from braket.circuits import Circuit, GateCalibrations, QubitSet
from braket.circuits.gates import Rx

device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-2")

calibrations = device.gate_calibrations
print(f"Downloaded {len(calibrations)} calibrations.")
```

> **Note**
>
> I fornitori di hardware li calibrano periodicamenteQPU, spesso più di una volta al giorno. Il Braket SDK consente di ottenere le calibrazioni più recenti dei cancelli.

```
device.refresh_gate_calibrations()
```

Per recuperare un determinato gate nativo, come il gate RX o XY, è necessario passare l'`Gate`oggetto e i qubit di interesse. Ad esempio, è possibile ispezionare l'implementazione a impulsi del RX (π /2) applicato su qubit 0.

```
rx_pi_2_q0 = (Rx(math.pi/2), QubitSet(0))

pulse_sequence_rx_pi_2_q0 = calibrations.pulse_sequences[rx_pi_2_q0]
```

È possibile creare un set filtrato di calibrazioni utilizzando la funzione. `filter` Si passa un elenco di porte o un elenco di. `QubitSet` Il codice seguente crea due set che contengono tutte le calibrazioni per RX (π /2) e per qubit 0.

```
rx_calibrations = calibrations.filter(gates=[Rx(math.pi/2)])
q0_calibrations = calibrations.filter(qubits=QubitSet([0])
```

Ora puoi fornire o modificare l'azione delle porte native collegando un set di calibrazione personalizzato. Ad esempio, si consideri il seguente circuito.

```
bell_circuit = (
```

```
Circuit()
.rx(0,math.pi/2)
.rx(1,math.pi/2)
.iswap(0,1)
.rx(1,-math.pi/2)
)
```

Puoi eseguirlo con una calibrazione di gate personalizzata per il `rx` gate on `qubit 0` passando un dizionario di `PulseSequence` oggetti all'argomento della `gate_definitions` parola chiave. È possibile costruire un dizionario dall'attributo `pulse_sequences` dell'`GateCalibrations`oggetto. Tutte le porte non specificate vengono sostituite con la calibrazione a impulsi del fornitore di hardware quantistico.

```
nb_shots = 50
custom_calibration = GateCalibrations({rx_pi_2_q0: pulse_sequence_rx_pi_2_q0})
task=device.run(bell_circuit, gate_definitions=custom_calibration.pulse_sequences,
 shots=nb_shots)
```

Simulazione hamiltoniana analogica

La simulazione hamiltoniana analogica (AHS) è un paradigma dell'informatica quantistica diverso dai circuiti quantistici: invece di una sequenza di porte, ciascuna delle quali agisce solo su un paio di qubit alla volta, viene definito un AHS programma dai parametri dell'hamiltoniano in questione che dipendono dal tempo e dallo spazio. L'hamiltoniano di un sistema codifica i suoi livelli di energia e gli effetti delle forze esterne, che insieme governano l'evoluzione temporale dei suoi stati. Per un sistema a N-qubit, l'hamiltoniano può essere rappresentato da una matrice quadrata di numeri complessi di 2 N X2 N.

I dispositivi quantistici in grado di funzionare AHS regoleranno i propri parametri (ad esempio l'ampiezza e la desintonizzazione di un campo determinante coerente) per approssimare da vicino l'evoluzione temporale del sistema quantistico secondo l'Hamiltoniano personalizzato. Il AHS paradigma è adatto per simulare le proprietà statiche e dinamiche dei sistemi quantistici di molte particelle interagenti. Costruito appositamenteQPUs, come il dispositivo Aquila di QuEra può simulare l'evoluzione temporale di sistemi con dimensioni altrimenti irrealizzabili sull'hardware classico.

In questa sezione:

- CiaoAHS: esegui la tua prima simulazione hamiltoniana analogica
- Invia un programma analogico usando Aquila QuEra

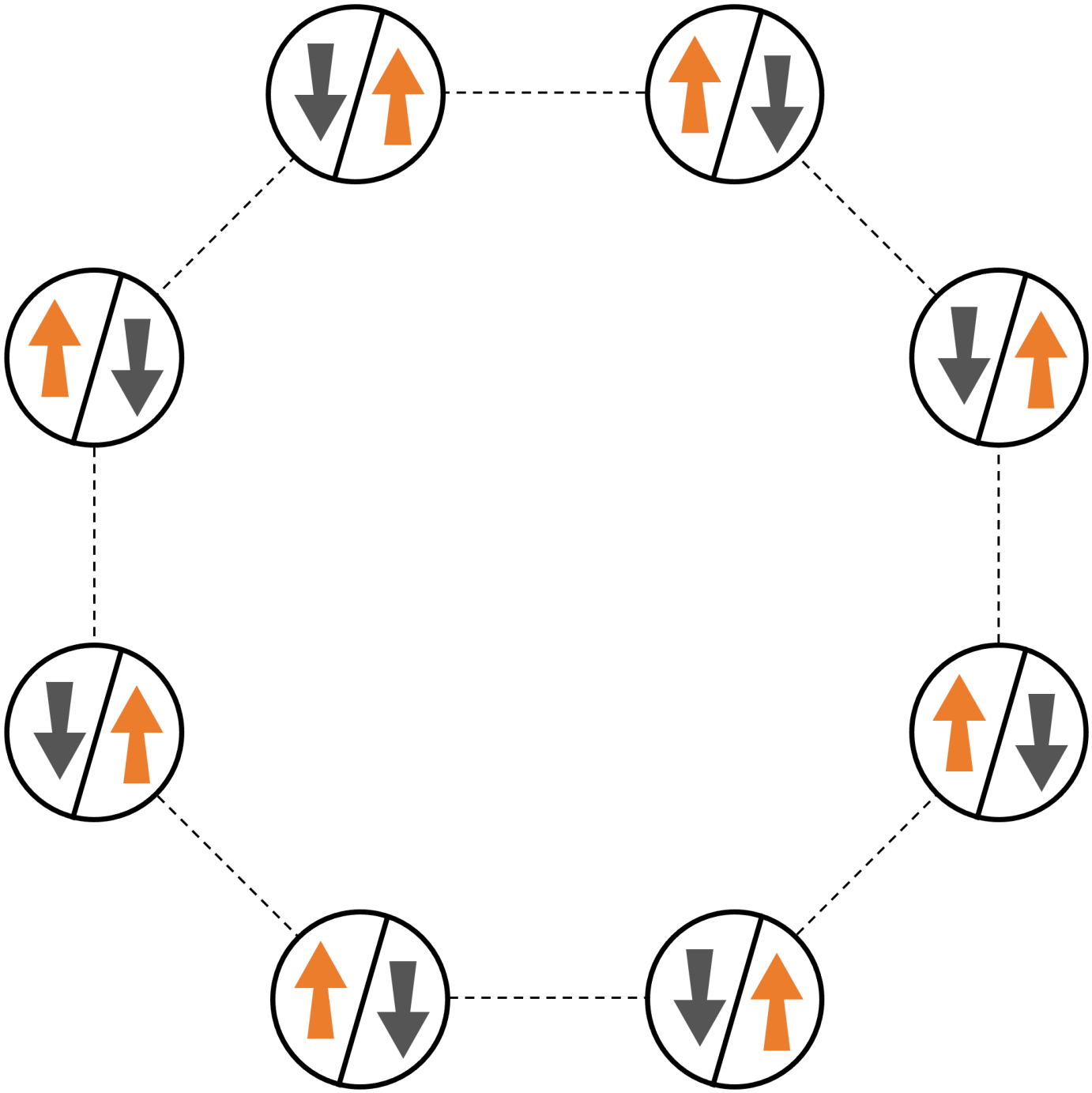
CiaoAHS: esegui la tua prima simulazione hamiltoniana analogica

Questa sezione fornisce informazioni sull'esecuzione della prima simulazione hamiltoniana analogica.

In questa sezione:

Catena di rotazione interagente

Per un esempio canonico di un sistema di molte particelle interagenti, consideriamo un anello di otto spin (ognuno dei quali può trovarsi negli stati «su» ↑# e «giù» ↓ #). Anche se piccolo, questo sistema modello mostra già una manciata di interessanti fenomeni legati ai materiali magnetici presenti in natura. In questo esempio, mostreremo come preparare un cosiddetto ordine antiferromagnetico, in cui gli spin consecutivi puntano in direzioni opposte.



Disposizione

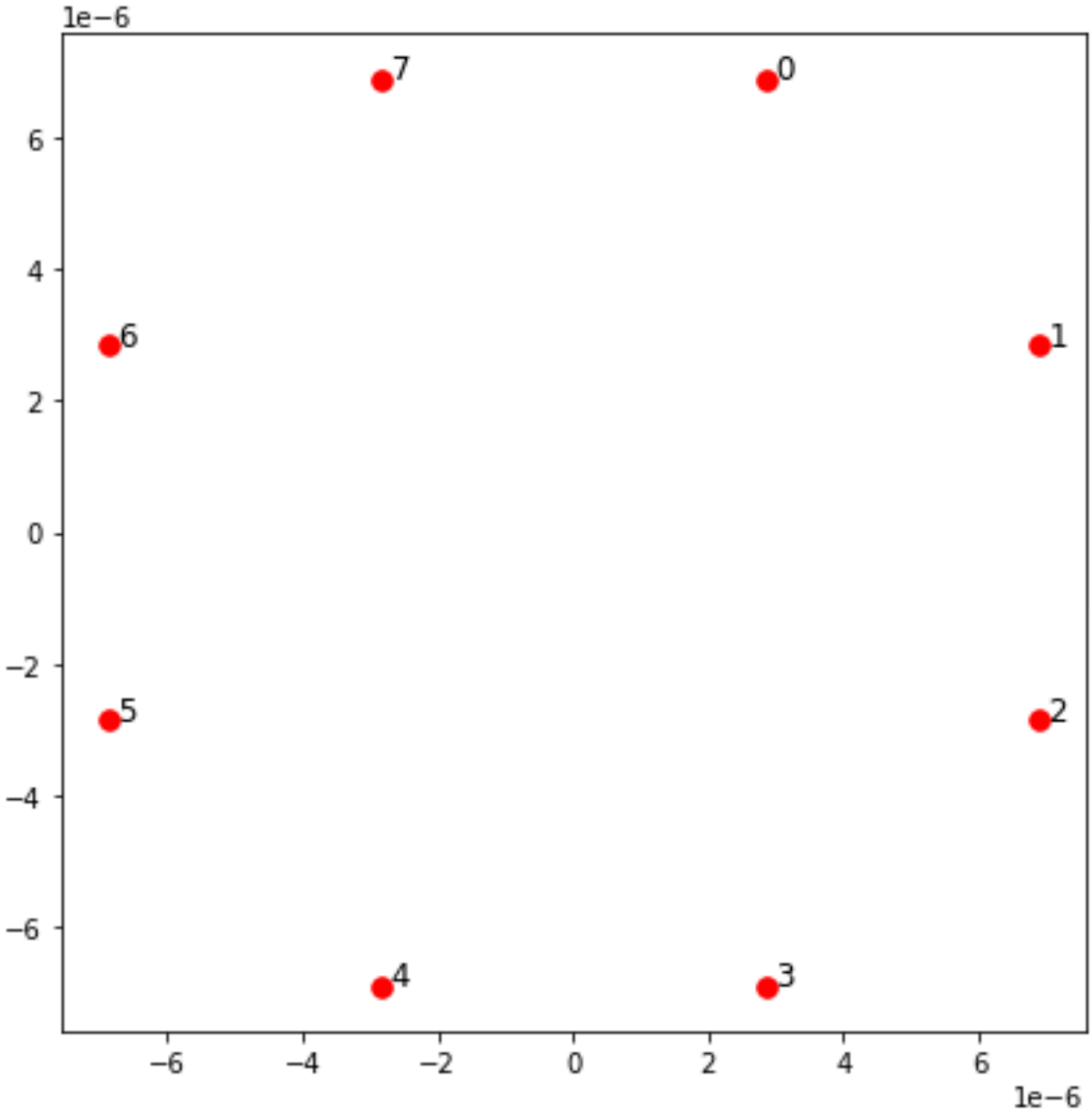
Useremo un atomo neutro per rappresentare ogni spin, e gli stati di spin «su» e «giù» saranno codificati rispettivamente nello stato eccitato di Rydberg e nello stato fondamentale degli atomi. Per prima cosa, creiamo la disposizione bidimensionale. Possiamo programmare il suddetto anello di giri con il seguente codice.

Prerequisiti: è necessario installare pip Braket. SDK (Se si utilizza un'istanza di notebook ospitata da Braket, questa SDK viene preinstallata con i notebook.) Per riprodurre i grafici, è inoltre necessario installare separatamente matplotlib con il comando shell. `pip install matplotlib`

```
import numpy as np
   import matplotlib.pyplot as plt  # required for plotting

   from braket.ahs.atom_arrangement import AtomArrangement

   a = 5.7e-6  # nearest-neighbor separation (in meters)

   register = AtomArrangement()
   register.add(np.array([0.5, 0.5 + 1/np.sqrt(2)]) * a)
   register.add(np.array([0.5 + 1/np.sqrt(2), 0.5]) * a)
   register.add(np.array([0.5 + 1/np.sqrt(2), - 0.5]) * a)
   register.add(np.array([0.5, - 0.5 - 1/np.sqrt(2)]) * a)
   register.add(np.array([-0.5, - 0.5 - 1/np.sqrt(2)]) * a)
   register.add(np.array([-0.5 - 1/np.sqrt(2), - 0.5]) * a)
   register.add(np.array([-0.5 - 1/np.sqrt(2), 0.5]) * a)
   register.add(np.array([-0.5, 0.5 + 1/np.sqrt(2)]) * a)
```

con cui possiamo anche tracciare

```
fig, ax = plt.subplots(1, 1, figsize=(7,7))
   xs, ys = [register.coordinate_list(dim) for dim in (0, 1)]
   ax.plot(xs, ys, 'r.', ms=15)
   for idx, (x, y) in enumerate(zip(xs, ys)):
       ax.text(x, y, f" {idx}", fontsize=12)
   plt.show()  # this will show the plot below in an ipython or jupyter session
```



Interazione

Per preparare la fase antiferromagnetica, dobbiamo indurre interazioni tra spin adiacenti. A tale scopo utilizziamo l'interazione di van der Waals, che viene implementata nativamente da dispositivi atomici neutri (come il Aquila dispositivo da QuEra). Utilizzando la rappresentazione dello spin, il termine hamiltoniano per questa interazione può essere espresso come somma di tutte le coppie di spin (j, k).

$$H_{\text{interaction}} = \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} V_{j,k} n_j n_k$$

In questo caso, nj=^ ↑ è un operatore che assume $_j$ il valore di 1 solo se lo spin j è nello stato $_j$ «alto», e 0 in caso contrario. La forza è V $_{j,k}$ =C_6/($d_{j,k}$) ^{6}, dove C $_6$ è il coefficiente fisso e d è la distanza euclidea tra gli spin j e $_{j,k}$ k. L'effetto immediato di questo termine di interazione è che ogni stato in cui sia lo spin j che lo spin k sono «verso l'alto» ha un'energia elevata (della quantità V). $_{j,k}$ Progettando attentamente il resto del AHS programma, questa interazione eviterà che entrambi gli spin adiacenti si trovino nello stato «attivo», un effetto comunemente noto come «blocco di Rydberg».

Campo di guida

All'inizio del AHS programma, tutti gli spin (per impostazione predefinita) iniziano nel loro stato «inattivo», si trovano in una cosiddetta fase ferromagnetica. Tenendo d'occhio il nostro obiettivo di preparare la fase antiferromagnetica, specifichiamo un campo guida coerente dipendente dal tempo che trasferisce agevolmente gli spin da questo stato a uno stato a molti corpi in cui sono preferiti gli stati «alti». L'hamiltoniano corrispondente può essere scritto come

$$H_{\text{drive}}(t) = \sum_{k=1}^{N} \frac{1}{2}\Omega(t)\left[e^{i\phi(t)} S_{-,k} + e^{-i\phi(t)} S_{+,k}\right] - \sum_{k=1}^{N} \Delta(t) n_k$$

dove Ω (t), (t), Δ (t) sono l'ampiezza globale (nota anche come frequenza Rabi), la fase e la desintonizzazione del campo di pilotaggio dipendenti dal tempo che influiscono uniformemente su tutti gli spin. Qui S $_{-,k}$ =↓ $_k$ †↑ $_k$ e S $_{+,k}$ =($S_{-,k}$) † =↑ ↑ $_k$ ↓ $_k$ sono rispettivamente gli operatori di abbassamento e innalzamento dello spin k, e $_k$ n $_k$ =↑ ↑ ≤ $_k$ è lo stesso operatore di prima. La parte Ω del campo di pilotaggio accoppia in modo coerente gli stati «giù» e «su» di tutti gli spin contemporaneamente, mentre la parte Δ controlla la ricompensa energetica per gli stati «su».

Per programmare una transizione graduale dalla fase ferromagnetica alla fase antiferromagnetica, specifichiamo il campo di pilotaggio con il seguente codice.

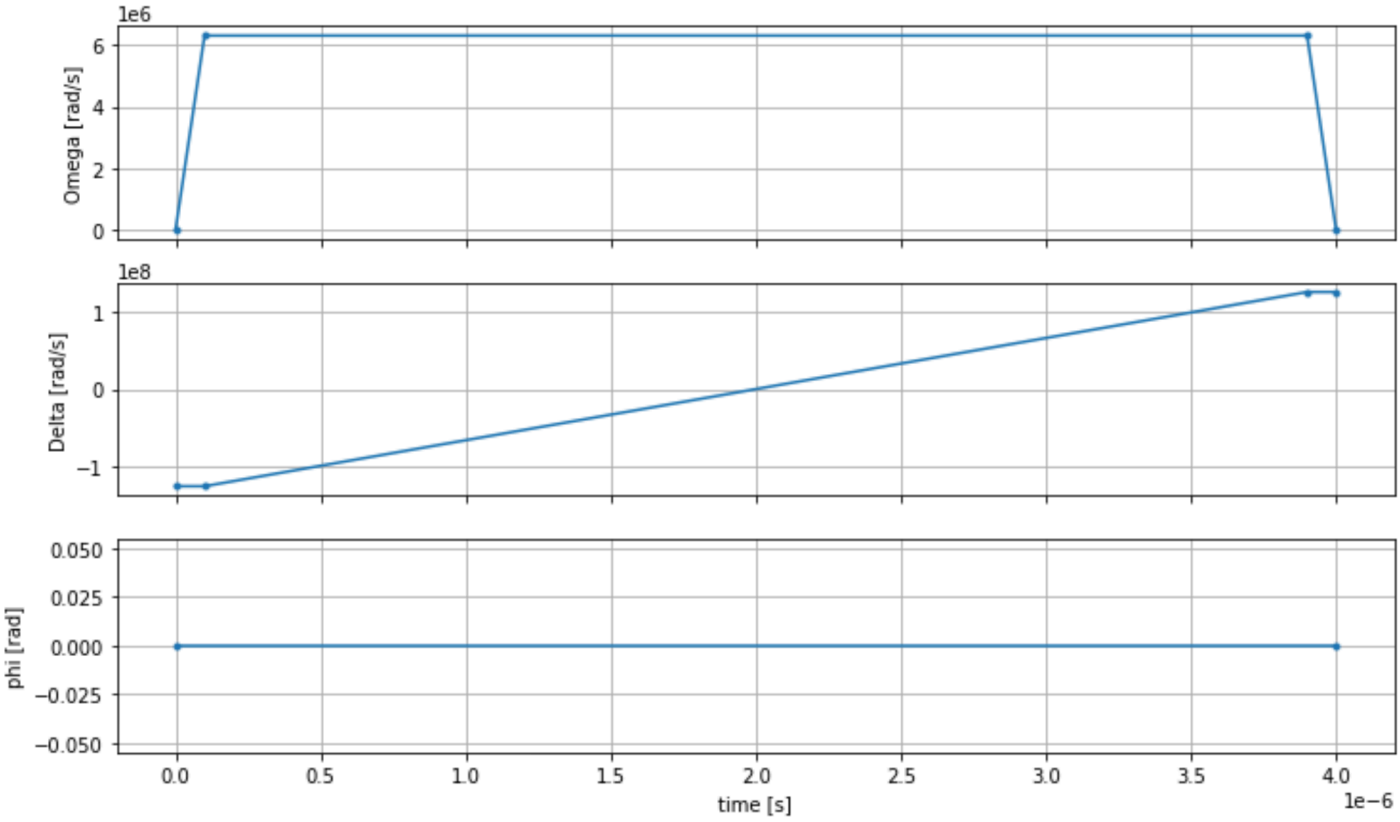
```
from braket.timings.time_series import TimeSeries
   from braket.ahs.driving_field import DrivingField

   # smooth transition from "down" to "up" state
   time_max = 4e-6  # seconds
   time_ramp = 1e-7  # seconds
   omega_max = 6300000.0  # rad / sec
```

```
    delta_start = -5 * omega_max
    delta_end = 5 * omega_max

    omega = TimeSeries()
    omega.put(0.0, 0.0)
    omega.put(time_ramp, omega_max)
    omega.put(time_max - time_ramp, omega_max)
    omega.put(time_max, 0.0)

    delta = TimeSeries()
    delta.put(0.0, delta_start)
    delta.put(time_ramp, delta_start)
    delta.put(time_max - time_ramp, delta_end)
    delta.put(time_max, delta_end)

    phi = TimeSeries().put(0.0, 0.0).put(time_max, 0.0)

    drive = DrivingField(
        amplitude=omega,
        phase=phi,
        detuning=delta
    )
```

Possiamo visualizzare le serie temporali del campo di guida con il seguente script.

```
fig, axes = plt.subplots(3, 1, figsize=(12, 7), sharex=True)

    ax = axes[0]
    time_series = drive.amplitude.time_series
    ax.plot(time_series.times(), time_series.values(), '.-');
    ax.grid()
    ax.set_ylabel('Omega [rad/s]')

    ax = axes[1]
    time_series = drive.detuning.time_series
    ax.plot(time_series.times(), time_series.values(), '.-');
    ax.grid()
    ax.set_ylabel('Delta [rad/s]')

    ax = axes[2]
    time_series = drive.phase.time_series
    # Note: time series of phase is understood as a piecewise constant function
    ax.step(time_series.times(), time_series.values(), '.-', where='post');
```

```
    ax.set_ylabel('phi [rad]')
    ax.grid()
    ax.set_xlabel('time [s]')

    plt.show()  # this will show the plot below in an ipython or jupyter session
```



AHSprogramma

Il registro, il campo di guida (e le interazioni implicite di van der Waals) costituiscono il programma di simulazione hamiltoniana analogica. `ahs_program`

```
from braket.ahs.analog_hamiltonian_simulation import AnalogHamiltonianSimulation

    ahs_program = AnalogHamiltonianSimulation(
        register=register,
        hamiltonian=drive
    )
```

In esecuzione su un simulatore locale

Poiché questo esempio è piccolo (meno di 15 giri), prima di eseguirlo su un dispositivo AHS compatibileQPU, possiamo eseguirlo sul AHS simulatore locale fornito con il Braket. SDK Poiché il simulatore locale è disponibile gratuitamente con BraketSDK, questa è la migliore pratica per garantire che il nostro codice possa essere eseguito correttamente.

Qui, possiamo impostare il numero di scatti su un valore elevato (ad esempio, 1 milione) perché il simulatore locale traccia l'evoluzione temporale dello stato quantistico e preleva campioni dallo stato finale, aumentando quindi il numero di scatti e aumentando la durata totale solo marginalmente.

```
from braket.devices import LocalSimulator
   device = LocalSimulator("braket_ahs")

   result_simulator = device.run(
       ahs_program,
       shots=1_000_000
   ).result()  # takes about 5 seconds
```

Analisi dei risultati del simulatore

Possiamo aggregare i risultati dei colpi con la seguente funzione che deduce lo stato di ogni rotazione (che può essere «d» per «giù», «u» per «su» o «e» per il sito vuoto) e conta quante volte ogni configurazione si è verificata tra i colpi.

```
from collections import Counter

   def get_counts(result):
       """Aggregate state counts from AHS shot results

       A count of strings (of length = # of spins) are returned, where
       each character denotes the state of a spin (site):
          e: empty site
          u: up state spin
          d: down state spin

       Args:
          result
 (braket.tasks.analog_hamiltonian_simulation_quantum_task_result.AnalogHamiltonianSimulationQua

       Returns
```

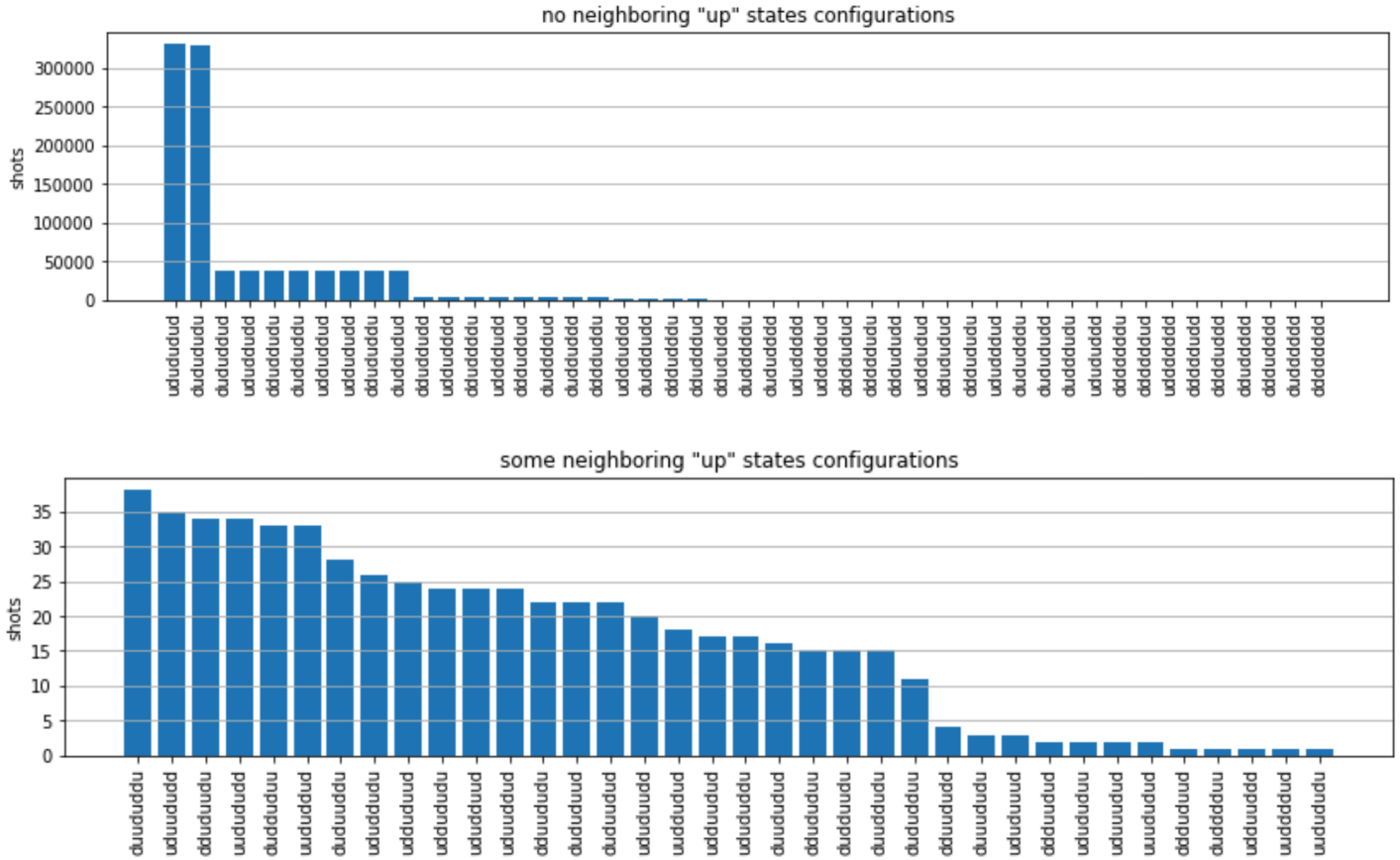
```
            dict: number of times each state configuration is measured

        """
        state_counts = Counter()
        states = ['e', 'u', 'd']
        for shot in result.measurements:
            pre = shot.pre_sequence
            post = shot.post_sequence
            state_idx = np.array(pre) * (1 + np.array(post))
            state = "".join(map(lambda s_idx: states[s_idx], state_idx))
            state_counts.update((state,))
        return dict(state_counts)

    counts_simulator = get_counts(result_simulator)  # takes about 5 seconds
    print(counts_simulator)
```

```
{'udududud': 330944, 'dudududu': 329576, 'dududdud': 38033, ...}
```

`counts`Ecco un dizionario che conta il numero di volte in cui ogni configurazione di stato viene osservata tra le riprese. Possiamo anche visualizzarli con il codice seguente.

```
from collections import Counter

    def has_neighboring_up_states(state):
        if 'uu' in state:
            return True
        if state[0] == 'u' and state[-1] == 'u':
            return True
        return False

    def number_of_up_states(state):
        return Counter(state)['u']

    def plot_counts(counts):
        non_blockaded = []
        blockaded = []
        for state, count in counts.items():
            if not has_neighboring_up_states(state):
                collection = non_blockaded
            else:
                collection = blockaded
            collection.append((state, count, number_of_up_states(state)))
```

```
    blockaded.sort(key=lambda _: _[1], reverse=True)
    non_blockaded.sort(key=lambda _: _[1], reverse=True)

    for configurations, name in zip((non_blockaded,
                                     blockaded),
                                    ('no neighboring "up" states',
                                     'some neighboring "up" states')):
        plt.figure(figsize=(14, 3))
        plt.bar(range(len(configurations)), [item[1] for item in configurations])
        plt.xticks(range(len(configurations)))
        plt.gca().set_xticklabels([item[0] for item in configurations], rotation=90)
        plt.ylabel('shots')
        plt.grid(axis='y')
        plt.title(f'{name} configurations')
        plt.show()

plot_counts(counts_simulator)
```



Dai grafici, possiamo leggere le seguenti osservazioni per verificare di aver preparato con successo la fase antiferromagnetica.

1. In genere, gli stati non bloccati (in cui non ci sono due spin adiacenti nello stato «attivo») sono più comuni degli stati in cui almeno una coppia di spin adiacenti si trova entrambi nello stato «positivo».
2. In genere, sono preferiti gli stati con più eccitazioni «in alto», a meno che la configurazione non sia bloccata.
3. Gli stati più comuni sono infatti gli stati antiferromagnetici perfetti e. `"dudududu" "udududud"`
4. I secondi stati più comuni sono quelli in cui ci sono solo 3 eccitazioni «verso l'alto» con separazioni consecutive di 1, 2, 2. Ciò dimostra che l'interazione di van der Waals ha un effetto (anche se molto minore) anche sui vicini più prossimi.

QuEraRunning On è l'Aquila QPU

Prerequisiti: oltre all'installazione pip di Braket SDK, se non utilizzi Amazon Braket, assicurati di aver completato i passaggi introduttivi necessari.

> **Note**
>
> Se utilizzi un'istanza di notebook ospitata da Braket, Braket viene preinstallato con l'istanza. SDK

Con tutte le dipendenze installate, possiamo connetterci a Aquila QPU.

```
from braket.aws import AwsDevice

   aquila_qpu = AwsDevice("arn:aws:braket:us-east-1::device/qpu/quera/Aquila")
```

Per rendere il nostro AHS programma adatto a QuEra macchina, dobbiamo arrotondare tutti i valori per rispettare i livelli di precisione consentiti dal Aquila QPU. (Questi requisiti sono regolati dai parametri del dispositivo con «Risoluzione» nel nome. Possiamo vederli `aquila_qpu.properties.dict()` eseguendoli su un notebook. Per ulteriori dettagli sulle funzionalità e sui requisiti di Aquila, vedere il notebook Introduzione ad Aquila.) Possiamo farlo chiamando il `discretize` metodo.

```
discretized_ahs_program = ahs_program.discretize(aquila_qpu)
```

Ora possiamo eseguire il programma (per ora eseguendo solo 100 scatti) sul Aquila QPU.

> **Note**
>
> Esecuzione di questo programma su Aquila il processore avrà un costo. Amazon Braket SDK include un Cost Tracker che consente ai clienti di impostare limiti di costo e di tenere traccia dei costi quasi in tempo reale.

```
task = aquila_qpu.run(discretized_ahs_program, shots=100)

   metadata = task.metadata()
   task_arn = metadata['quantumTaskArn']
   task_status = metadata['status']

   print(f"ARN: {task_arn}")
   print(f"status: {task_status}")
```

```
task ARN: arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef
   task status: CREATED
```

A causa della grande differenza tra il tempo di esecuzione di un'attività quantistica (a seconda delle finestre di disponibilità e QPU dell'utilizzo), è una buona idea annotare l'attività quantisticaARN, in modo da poterne controllare lo stato in un secondo momento con il seguente frammento di codice.

```
# Optionally, in a new python session

   from braket.aws import AwsQuantumTask

   SAVED_TASK_ARN = "arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef"

   task = AwsQuantumTask(arn=SAVED_TASK_ARN)
   metadata = task.metadata()
   task_arn = metadata['quantumTaskArn']
   task_status = metadata['status']

   print(f"ARN: {task_arn}")
   print(f"status: {task_status}")
```

```
*[Output]*
```

```
   task ARN: arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef
   task status: COMPLETED
```

Una volta raggiunto lo stato COMPLETED (che può essere verificato anche dalla pagina delle attività quantistiche della console Amazon Braket), possiamo interrogare i risultati con:

```
result_aquila = task.result()
```

Analisi dei risultati QPU

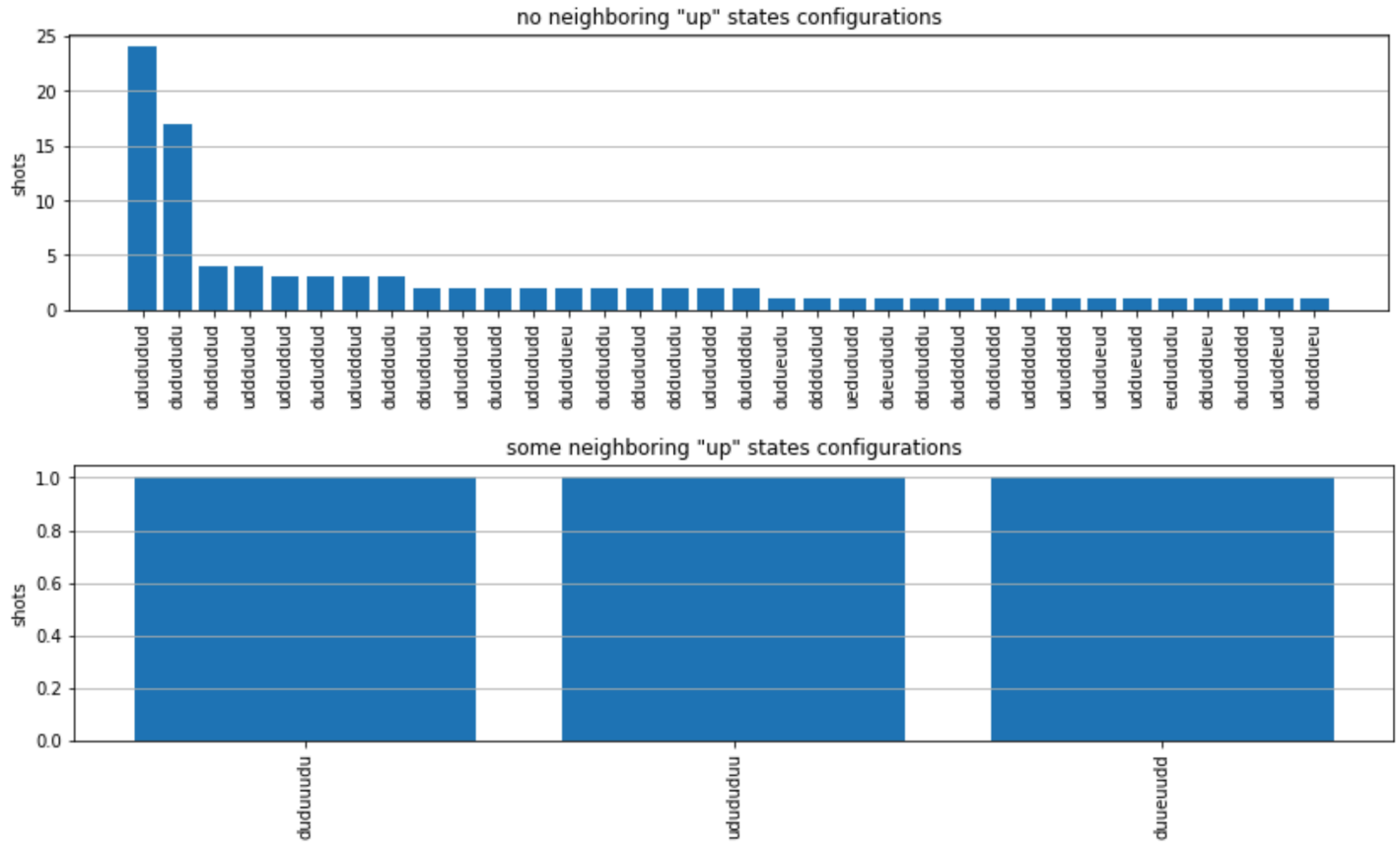
Utilizzando le stesse `get_counts` funzioni di prima, possiamo calcolare i conteggi:

```
counts_aquila = get_counts(result_aquila)
   print(counts_aquila)
```

```
*[Output]*
   {'udududud': 24, 'dudududu': 17, 'dududdud': 3, ...}
```

e tracciali con`plot_counts`:

```
plot_counts(counts_aquila)
```



Nota che una piccola parte delle riprese ha aree vuote (contrassegnate con «e»). Ciò è dovuto a un 1-2% per atomo di imperfezioni di preparazione del Aquila QPU. Inoltre, i risultati corrispondono alla simulazione nell'ambito della fluttuazione statistica prevista a causa del numero ridotto di scatti.

Passaggi successivi

Congratulazioni, ora hai eseguito il tuo primo AHS carico di lavoro su Amazon Braket utilizzando il simulatore AHS locale e il Aquila QPU.

Per saperne di più sulla fisica di Rydberg, sulla simulazione hamiltoniana analogica e sul Aquila dispositivo, fai riferimento ai nostri notebook di esempio.

Invia un programma analogico usando Aquila QuEra

Questa pagina fornisce una documentazione completa sulle funzionalità di Aquila macchina da QuEra. I dettagli trattati qui sono i seguenti: 1) L'hamiltoniano parametrizzato simulato da Aquila, 2) parametri del AHS programma, 3) contenuto dei risultati, 4) AHS Aquila parametro delle capacità. Ti consigliamo di utilizzare la ricerca testuale Ctrl+F per trovare i parametri pertinenti alle tue domande.

In questa sezione:

Hamiltoniano

Il Aquila macchina da QuEra simula nativamente il seguente hamiltoniano (dipendente dal tempo):

$$H(t) = \sum_{k=1}^{N} H_{\text{drive},k}(t) + \sum_{k=1}^{N} H_{\text{local detuning},k}(t) + \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} V_{\text{vdw},k,l}$$

> **Note**
>
> L'accesso alla detuning locale è una funzionalità sperimentale ed è disponibile su richiesta tramite Braket Direct.

dove

- H drive,k (t) =($^{1}/_{2}$Ω (t) e i $^{(t)}$ S −,k + $^{1}/_{2}$Ω (t) s) + $^{(-\Delta\ (t_{+,k})}$ n), global k
  - Ω (t) è l'ampiezza di guida globale, dipendente dal tempo (nota anche come frequenza Rabi), espressa in unità di (rad/s)
  - □ (t) è la fase globale, dipendente dal tempo, misurata in radianti
  - S −,k e S +,k sono gli operatori che abbassano e alzano lo spin dell'atomo k (nella base |↓#$^{=|g\#,\ |}$ ↑=|r#, sono S =|gr|, S =(S) † =|rg|) − + −
  - Δ ($_{global}$t) è la detonazione globale e dipendente dal tempo
  - n $_{k}$ è l'operatore di proiezione sullo stato di Rydberg dell'atomo k (cioè n=|r.co.r|)
- H (t) =-Δ (t) h n local detuning,k local k k
  - Δ local (t) è il fattore dipendente dal tempo dello spostamento di frequenza locale, espresso in unità di (rad/s)
  - h $_{k}$ è il fattore dipendente dal sito, un numero adimensionale compreso tra 0,0 e 1,0
- V vdw,k,l = C_6/($d_{k,l}$) 6 n n, k l
  - C $_{6}$ è il coefficiente di van der Waals, espresso in unità di (rad/ s) * (m) ^6

- $d_{k,l}$ è la distanza euclidea tra l'atomo k e l, misurata in metri.

Gli utenti hanno il controllo sui seguenti parametri tramite lo schema del programma Braket. AHS

- Disposizione bidimensionale degli atomi ($_k$coordinate x $_k$ e y di ciascun atomo k, in unità di um), che controlla le distanze atomiche a coppie $d_{k,l}$ con k, l=1,2,... N
- Ω (t), la frequenza Rabi globale dipendente dal tempo, in unità di (rad/ s)
- (t), la fase globale, dipendente dal tempo, in unità di (rad)
- Δ_{global} (t), la detonizzazione globale dipendente dal tempo, in unità di (rad/s)
- Δ_{local} (t), il fattore (globale) dipendente dal tempo dell'entità della detonizzazione locale, in unità di (rad/ s)
- h_k, il fattore (statico) dipendente dal sito dell'entità della detonizzazione locale, un numero adimensionale compreso tra 0,0 e 1,0

> **Note**
>
> L'utente non può controllare quali livelli sono coinvolti (ad esempio gli operatori S_-, S_+, n sono fissi) né l'intensità del coefficiente di interazione Rydberg-Rydberg (C). $_6$

Schema del programma Braket AHS

Oggetto Braket.ir.ahs.program_v1.program (esempio)

```
Program(
    braketSchemaHeader=BraketSchemaHeader(
        name='braket.ir.ahs.program',
        version='1'
    ),
    setup=Setup(
        ahs_register=AtomArrangement(
            sites=[
                [Decimal('0'), Decimal('0')],
                [Decimal('0'), Decimal('4e-6')],
                [Decimal('4e-6'), Decimal('0')],
            ],
            filling=[1, 1, 1]
        )
```

```
    ),
    hamiltonian=Hamiltonian(
        drivingFields=[
            DrivingField(
                amplitude=PhysicalField(
                    time_series=TimeSeries(
                        values=[Decimal('0'), Decimal('15700000.0'),
Decimal('15700000.0'), Decimal('0')],
                        times=[Decimal('0'), Decimal('0.000001'), Decimal('0.000002'),
Decimal('0.000003')]
                    ),
                    pattern='uniform'
                ),
                phase=PhysicalField(
                    time_series=TimeSeries(
                        values=[Decimal('0'), Decimal('0')],
                        times=[Decimal('0'), Decimal('0.000001')]
                    ),
                    pattern='uniform'
                ),
                detuning=PhysicalField(
                    time_series=TimeSeries(
                        values=[Decimal('-54000000.0'), Decimal('54000000.0')],
                        times=[Decimal('0'), Decimal('0.000001')]
                    ),
                    pattern='uniform'
                )
            )
        ],
        localDetuning=[
            LocalDetuning(
                magnitude=PhysicalField(
                    times_series=TimeSeries(
                        values=[Decimal('0'), Decimal('25000000.0'),
Decimal('25000000.0'), Decimal('0')],
                        times=[Decimal('0'), Decimal('0.000001'), Decimal('0.000002'),
Decimal('0.000003')]
                    ),
                    pattern=Pattern([Decimal('0.8'), Decimal('1.0'), Decimal('0.9')])
                )
            )
        ]
    )
```

```
)
```

JSON(esempio)

```
{
    "braketSchemaHeader": {
        "name": "braket.ir.ahs.program",
        "version": "1"
    },
    "setup": {
        "ahs_register": {
            "sites": [
                [0E-7, 0E-7],
                [0E-7, 4E-6],
                [4E-6, 0E-7],
            ],
            "filling": [1, 1, 1]
        }
    },
    "hamiltonian": {
        "drivingFields": [
            {
                "amplitude": {
                    "time_series": {
                        "values": [0.0, 15700000.0, 15700000.0, 0.0],
                        "times": [0E-9, 0.000001000, 0.000002000, 0.000003000]
                    },
                    "pattern": "uniform"
                },
                "phase": {
                    "time_series": {
                        "values": [0E-7, 0E-7],
                        "times": [0E-9, 0.000001000]
                    },
                    "pattern": "uniform"
                },
                "detuning": {
                    "time_series": {
                        "values": [-54000000.0, 54000000.0],
                        "times": [0E-9, 0.000001000]
                    },
                    "pattern": "uniform"
                }
```

```
            }
        ],
        "localDetuning": [
            {
                "magnitude": {
                    "time_series": {
                        "values": [0.0, 25000000.0, 25000000.0, 0.0],
                        "times": [0E-9, 0.000001000, 0.000002000, 0.000003000]
                    },
                    "pattern": [0.8, 1.0, 0.9]
                }
            }
        ]
    }
}
```

Campi principali

Campo del programma	tipo	description
setup.ahs_register.sites	Elenco [Elenco [Decimale]]	Elenco delle coordinate bidimensionali in cui le pinzette intrappolano gli atomi
setup.ahs_register.filling	Elenco [int]	Contrassegna gli atomi che occupano i siti trap con 1 e i siti vuoti con 0
hamiltoniano. drivingFields[] .amplitude.time_series.times	Elenco [Decimale]	punti temporali dell'ampiezza di guida, Omega (t)
hamiltoniano. drivingFields[] .amplitude.time_series.values	Elenco [Decimale]	valori dell'ampiezza di guida, Omega (t)

Campo del programma	tipo	description
hamiltoniano. drivingFields[] .ampiezza.modello	str	modello spaziale di ampiezza di guida, Omega (t); deve essere «uniforme»
hamiltoniana. drivingFields[] .phase.time_series.times	Elenco [Decimale]	punti temporali della fase di guida, phi (t)
hamiltoniano. drivingFields[] .phase.time_series .values	Elenco [Decimale]	valori della fase di guida, phi (t)
hamiltoniano. drivingFields[] .phase.pattern	str	modello spaziale della fase di guida, phi (t); deve essere «uniforme»
hamiltoniano. drivingFields[] .detuning.time_ser ies.times	Elenco [Decimale]	punti temporali di desintonizzazione della guida, delta_Global (t)
hamiltoniana. drivingFields[] .detuning.time_ser ies.values	Elenco [Decimale]	valori della detonizzazione della guida, delta_Global (t)
hamiltoniano. drivingFields[] .detuning.pattern	str	modello spaziale di detuning di guida, delta_glo bal (t); deve essere «uniforme»

Campo del programma	tipo	description
hamiltoniano. localDetuning[] .magnitude.time_series.times	Elenco [Decimale]	punti temporali del fattore dipendente dal tempo della magnitudine locale di detonizzazione, delta_local (t)
hamiltoniana. localDetuning[] .magnitude.time_series.values	Elenco [Decimale]	valori del fattore dipendente dal tempo della magnitudine locale di detonizzazione, delta_local (t)
hamiltoniana. localDetuning[] .magnitude.modello	Elenco [Decimale]	fattore dipendente dal sito della magnitudine di detonizzazione locale, h_k (i valori corrispondono ai siti in setup.ahs_register.sites)

Campi di metadati

Campo del programma	tipo	description
braketSchemaHeader.nome	str	nome dello schema; deve essere 'braket.ir.ahs.program'
braketSchemaHeader.versione	str	versione dello schema

schema dei risultati dell'AHSattività Braket

braket.tasks.analog_hamiltonian_simulation_quantum_task_result.
AnalogHamiltonianSimulationQuantumTaskResult(esempio)

```
AnalogHamiltonianSimulationQuantumTaskResult(
    task_metadata=TaskMetadata(
        braketSchemaHeader=BraketSchemaHeader(
            name='braket.task_result.task_metadata',
            version='1'
        ),
        id='arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef',
        shots=2,
        deviceId='arn:aws:braket:us-east-1::device/qpu/quera/Aquila',
        deviceParameters=None,
        createdAt='2022-10-25T20:59:10.788Z',
        endedAt='2022-10-25T21:00:58.218Z',
        status='COMPLETED',
        failureReason=None
    ),
    measurements=[
        ShotResult(
            status=<AnalogHamiltonianSimulationShotStatus.SUCCESS: 'Success'>,

            pre_sequence=array([1, 1, 1, 1]),
            post_sequence=array([0, 1, 1, 1])
        ),

        ShotResult(
            status=<AnalogHamiltonianSimulationShotStatus.SUCCESS: 'Success'>,

            pre_sequence=array([1, 1, 0, 1]),
            post_sequence=array([1, 0, 0, 0])
        )
    ]
)
```

JSON(esempio)

```
{
    "braketSchemaHeader": {
        "name": "braket.task_result.analog_hamiltonian_simulation_task_result",
```

```
        "version": "1"
    },
    "taskMetadata": {
        "braketSchemaHeader": {
            "name": "braket.task_result.task_metadata",
            "version": "1"
        },
        "id": "arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef",
        "shots": 2,
        "deviceId": "arn:aws:braket:us-east-1::device/qpu/quera/Aquila",

        "createdAt": "2022-10-25T20:59:10.788Z",
        "endedAt": "2022-10-25T21:00:58.218Z",
        "status": "COMPLETED"

    },
    "measurements": [
        {
            "shotMetadata": {"shotStatus": "Success"},
            "shotResult": {
                "preSequence": [1, 1, 1, 1],
                "postSequence": [0, 1, 1, 1]
            }
        },
        {
            "shotMetadata": {"shotStatus": "Success"},
            "shotResult": {
                "preSequence": [1, 1, 0, 1],
                "postSequence": [1, 0, 0, 0]
            }
        }
    ],
    "additionalMetadata": {
        "action": {...}
        "queraMetadata": {
            "braketSchemaHeader": {
                "name": "braket.task_result.quera_metadata",
                "version": "1"
            },
            "numSuccessfulShots": 100
        }
    }
```

```
}
```

Campi principali

Campo dei risultati dell'attività	tipo	description
misurazioni []. shotResult. preSequence	Elenco [int]	Bit di misurazione pre-sequenza (uno per ogni sito atomico) per ogni ripresa: 0 se il sito è vuoto, 1 se il sito è pieno, misurati prima delle sequenze di impulsi che eseguono l'evoluzione quantistica
misurazioni []. shotResult. postSequence	Elenco [int]	Bit di misurazione post-sequenza per ogni ripresa: 0 se l'atomo è nello stato di Rydberg o il sito è vuoto, 1 se l'atomo è allo stato fondamentale, misurati alla fine delle sequenze di impulsi che eseguono l'evoluzione quantistica

Campi di metadati

Campo dei risultati dell'attività	tipo	description
braketSchemaHeader.nome	str	nome dello schema; deve essere 'braket.task_result.analog_hamiltonian_simulation_task_result'
braketSchemaHeader.versione	str	versione dello schema

Campo dei risultati dell'attività	tipo	description
taskMetadata. braketSchemaHeader.nome	str	nome dello schema; deve essere 'braket.task_result.task_metadata'
taskMetadata. braketSchemaHeader .versione	str	versione dello schema
taskMetadata.id	str	L'ID del compito quantistico. In AWS compiti quantistici, questo è il compito quantistico. ARN
taskMetadata.scatti	int	Il numero di scatti per l'operazione quantistica
taskMetadata.colpi. deviceId	str	L'ID del dispositivo su cui è stata eseguita l'operazione quantistica. In AWS dispositivi, questo è il dispositivoARN.

Campo dei risultati dell'attività	tipo	description
taskMetadata.scatti. createdAt	str	Il timestamp della creazione ; il formato deve essere in ISO -8601/ RFC3339 string format -MM-:MM:s s.sssz. YYYY DDTHH L'impostazione predefinita è Nessuno.
taskMetadata.shots. endedAt	str	Il timestamp di quando è terminata l'attività quantistica; il formato deve essere in ISO RFC3339 -8601/ string format - MM-:MM:s s.sssz. YYYY DDTHH L'impostazione predefinita è Nessuno.

Campo dei risultati dell'attività	tipo	description
taskMetadata.shots.status	str	Lo stato dell'attività quantistica (CREATED, ,,,QUEUED). RUNNING COMPLETED FAILED L'impostazione predefinita è Nessuno.
taskMetadata.shots. failureReason	str	Il motivo del fallimento del compito quantistico. L'impostazione predefinita è Nessuno.
additionalMetadata.azione	Braket.ir.ahs.program_v1.program	(Vedi la sezione sullo schema del programma Braket) AHS
additionalMetadata.azione. braketSchemaHeader. queraMetadata.nome	str	nome dello schema; deve essere 'braket.task_result.quera_metadata'
additionalMetadata.azione. braketSchemaHeader. queraMetadata.versione	str	versione dello schema

Campo dei risultati dell'attività	tipo	description
additionalMetadata.azione. numSuccessfulShots	int	numero di tiri completamente riusciti; deve essere uguale al numero di tiri richiesto
misurazioni []. shotMetadata. shotStatus	int	Lo stato dello scatto, (Riuscito , Riuscito parzialmente, Fallimento); deve essere «Riuscito»

QuEra schema delle proprietà del dispositivo

braket.device_schema.quera.quera_device_capabilities_v1. QueraDeviceCapabilities(esempio)

```
QueraDeviceCapabilities(
    service=DeviceServiceProperties(
        braketSchemaHeader=BraketSchemaHeader(
            name='braket.device_schema.device_service_properties',
            version='1'
            ),
            executionWindows=[
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.MONDAY: 'Monday'>,
                    windowStartHour=datetime.time(1, 0),
                    windowEndHour=datetime.time(23, 59, 59)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.TUESDAY: 'Tuesday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(12, 0)
                ),
```

```
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.WEDNESDAY: 'Wednesday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(12, 0)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.FRIDAY: 'Friday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(23, 59, 59)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.SATURDAY: 'Saturday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(23, 59, 59)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.SUNDAY: 'Sunday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(12, 0)
                )
            ],
            shotsRange=(1, 1000),
            deviceCost=DeviceCost(
                price=0.01,
                unit='shot'
            ),
            deviceDocumentation=
                DeviceDocumentation(
                    imageUrl='https://
a.b.cdn.console.awsstatic.com/59534b58c709fc239521ef866db9ea3f1aba73ad3ebcf60c23914ad8c5c5c878/
a6cfc6fca26cf1c2e1c6.png',
                    summary='Analog quantum processor based on neutral atom arrays',
                    externalDocumentationUrl='https://www.quera.com/aquila'
                ),
                deviceLocation='Boston, USA',
                updatedAt=datetime.datetime(2024, 1, 22, 12, 0,
 tzinfo=datetime.timezone.utc),
                getTaskPollIntervalMillis=None
    ),
    action={
        <DeviceActionType.AHS: 'braket.ir.ahs.program'>: DeviceActionProperties(
                version=['1'],
                actionType=<DeviceActionType.AHS: 'braket.ir.ahs.program'>
            )
```

```
    },
    deviceParameters={},
    braketSchemaHeader=BraketSchemaHeader(
        name='braket.device_schema.quera.quera_device_capabilities',
        version='1'
    ),
    paradigm=QueraAhsParadigmProperties(
        ...
        # See https://github.com/amazon-braket/amazon-braket-schemas-python/blob/main/
src/braket/device_schema/quera/quera_ahs_paradigm_properties_v1.py
        ...
    )
)
```

JSON(esempio)

```
{
    "service": {
        "braketSchemaHeader": {
            "name": "braket.device_schema.device_service_properties",
            "version": "1"
        },
        "executionWindows": [
            {
                "executionDay": "Monday",
                "windowStartHour": "01:00:00",
                "windowEndHour": "23:59:59"
            },
            {
                "executionDay": "Tuesday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "12:00:00"
            },
            {
                "executionDay": "Wednesday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "12:00:00"
            },
            {
                "executionDay": "Friday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "23:59:59"
            },
```

```
            {
                "executionDay": "Saturday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "23:59:59"
            },
            {
                "executionDay": "Sunday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "12:00:00"
            }
        ],
        "shotsRange": [
            1,
            1000
        ],
        "deviceCost": {
            "price": 0.01,
            "unit": "shot"
        },
        "deviceDocumentation": {
            "imageUrl": "https://
a.b.cdn.console.awsstatic.com/59534b58c709fc239521ef866db9ea3f1aba73ad3ebcf60c23914ad8c5c5c878/
a6cfc6fca26cf1c2e1c6.png",
            "summary": "Analog quantum processor based on neutral atom arrays",
            "externalDocumentationUrl": "https://www.quera.com/aquila"
        },
        "deviceLocation": "Boston, USA",
        "updatedAt": "2024-01-22T12:00:00+00:00"
    },
    "action": {
        "braket.ir.ahs.program": {
            "version": [
                "1"
            ],
            "actionType": "braket.ir.ahs.program"
        }
    },
    "deviceParameters": {},
    "braketSchemaHeader": {
        "name": "braket.device_schema.quera.quera_device_capabilities",
        "version": "1"
    },
    "paradigm": {
        ...
```

```
        # See Aquila device page > "Calibration" tab > "JSON" page
        ...
    }
}
```

Campi delle proprietà del servizio

Campo delle proprietà del servizio	tipo	description
servizio. executionWindows[]. executionDay	ExecutionDay	Giorni della finestra di esecuzione; deve essere «Tutti i giorni», «Giorni feriali», «Fine settimana », «lunedì», «martedì», «mercoledì», «giovedì» , «venerdì», «sabato» o «domenica»
servizio. executionWindows[]. windowStartHour	datetime.ora	UTCFormato a 24 ore dell'ora di inizio della finestra di esecuzione
servizio. executionWindows[]. windowEndHour	datetime.ora	UTCFormato a 24 ore dell'ora in cui termina la finestra di esecuzione
service.qpu_capabilities.service. shotsRange	Tupla [int, int]	Numero minimo e massimo di scatti per il dispositivo
service.qpu_capabilities.service. deviceCos t.prezzo	float	Prezzo del dispositivo in dollari USA
service.qpu_capabilities.service. deviceCos t.unità	str	unità per addebitare il prezzo, ad esempio: 'minute', 'hour', 'shot', 'task'

Campi di metadati

Campo di metadati	tipo	description
action [] .version	str	versione dello schema del AHS programma
azione []. actionType	ActionType	AHSnome dello schema del programma; deve essere 'braket.ir.ahs.program'
servizio. braketSchemaHeader.nome	str	nome dello schema; deve essere 'braket.device_schema.device_service_properties'
servizio. braketSchemaHeader.versione	str	versione dello schema
servizio. deviceDocumentation. imageUrl	str	URLper l'immagine del dispositivo
servizio. deviceDocumentation.sommario	str	breve descrizione sul dispositivo
servizio. deviceDocumentation. externalDocumentationUrl	str	documentazione esterna URL
servizio. deviceLocation	str	posizione geografica del dispositivo
servizio. updatedAt	datetime	ora dell'ultimo aggiornamento delle proprietà del dispositivo

Lavorare con Boto3

Boto3 è il AWS SDKper Python. Con Boto3, gli sviluppatori Python possono creare, configurare e gestire Servizi AWS, ad esempio Amazon Staffa. Boto3 fornisce un approccio orientato agli oggetti API, oltre all'accesso a basso livello a Amazon Staffa.

Segui le istruzioni nella guida rapida di Boto3 per scoprire come installare e configurare Boto3.

Boto3 fornisce le funzionalità di base che funzionano insieme a Amazon Braket SDK Python per aiutarti a configurare ed eseguire le tue attività quantistiche. I clienti Python devono sempre installare Boto3, perché questa è l'implementazione principale. Se desideri utilizzare metodi di supporto aggiuntivi, devi anche installare Amazon SDKStaffa.

Ad esempio, quando chiami`CreateQuantumTask`, il Amazon Braket SDK invia la richiesta a Boto3, che quindi chiama AWS API.

In questa sezione:

- Attiva il client Amazon Braket Boto3
- Configura AWS CLI profili per Boto3 e Amazon Braket SDK

Attiva il client Amazon Braket Boto3

Per utilizzare Boto3 con Amazon Braket, devi importare Boto3 e quindi definire un client da utilizzare per connetterti ad Amazon Braket API. Nell'esempio seguente, viene denominato il client Boto3. `braket`

```
import boto3
import botocore

braket = boto3.client("braket",
 config=botocore.client.Config(user_agent_extra="BraketSchemas/1.8.0"))
```

Ora che hai stabilito un `braket` cliente, puoi effettuare richieste ed elaborare le risposte dal Amazon Servizio Braket. Puoi ottenere maggiori dettagli sui dati di richiesta e risposta nel APIReference.

Gli esempi seguenti mostrano come lavorare con dispositivi e attività quantistiche.

- Cerca dispositivi
- Recupera un dispositivo

- Crea un'attività quantistica
- Recupera un compito quantistico
- Cerca attività quantistiche
- Annulla attività quantistica

Cerca dispositivi

- `search_devices(**kwargs)`

Cerca dispositivi utilizzando i filtri specificati.

```
# Pass search filters and optional parameters when sending the
#  request and capture the response
response = braket.search_devices(filters=[{
    'name': 'deviceArn',
    'values': ['arn:aws:braket:::device/quantum-simulator/amazon/sv1']
}], maxResults=10)

print(f"Found {len(response['devices'])} devices")

for i in range(len(response['devices'])):
    device = response['devices'][i]
    print(device['deviceArn'])
```

Recupera un dispositivo

- `get_device(deviceArn)`

Recupera i dispositivi disponibili in Amazon Braket.

```
# Pass the device ARN when sending the request and capture the repsonse
response = braket.get_device(deviceArn='arn:aws:braket:::device/quantum-simulator/
amazon/sv1')

print(f"Device {response['deviceName']} is {response['deviceStatus']}")
```

Crea un'attività quantistica

- `create_quantum_task(**kwargs)`

Crea un'attività quantistica.

```
# Create parameters to pass into create_quantum_task()
kwargs = {
    # Create a Bell pair
    'action': '{"braketSchemaHeader": {"name": "braket.ir.jaqcd.program", "version":
 "1"}, "results": [], "basis_rotation_instructions": [], "instructions": [{"type": "h",
 "target": 0}, {"type": "cnot", "control": 0, "target": 1}]}',
    # Specify the SV1 Device ARN
    'deviceArn': 'arn:aws:braket:::device/quantum-simulator/amazon/sv1',
    # Specify 2 qubits for the Bell pair
    'deviceParameters': '{"braketSchemaHeader": {"name":
 "braket.device_schema.simulators.gate_model_simulator_device_parameters",
 "version": "1"}, "paradigmParameters": {"braketSchemaHeader": {"name":
 "braket.device_schema.gate_model_parameters", "version": "1"}, "qubitCount": 2}}',
    # Specify where results should be placed when the quantum task completes.
    #  You must ensure the S3 Bucket exists before calling create_quantum_task()
    'outputS3Bucket': 'amazon-braket-examples',
    'outputS3KeyPrefix': 'boto-examples',
    # Specify number of shots for the quantum task
    'shots': 100
}

# Send the request and capture the response
response = braket.create_quantum_task(**kwargs)

print(f"Quantum task {response['quantumTaskArn']} created")
```

Recupera un compito quantistico

- `get_quantum_task(quantumTaskArn)`

Recupera il task quantistico specificato.

```
# Pass the quantum task ARN when sending the request and capture the response
response = braket.get_quantum_task(quantumTaskArn='arn:aws:braket:us-
west-1:123456789012:quantum-task/ce78c429-cef5-45f2-88da-123456789012')
```

```
print(response['status'])
```

Cerca attività quantistiche

- `search_quantum_tasks(**kwargs)`

Cerca attività quantistiche che corrispondono ai valori di filtro specificati.

```
# Pass search filters and optional parameters when sending the
#  request and capture the response
response = braket.search_quantum_tasks(filters=[{
    'name': 'deviceArn',
    'operator': 'EQUAL',
    'values': ['arn:aws:braket:::device/quantum-simulator/amazon/sv1']
}], maxResults=25)

print(f"Found {len(response['quantumTasks'])} quantum tasks")

for n in range(len(response['quantumTasks'])):
    task = response['quantumTasks'][n]
    print(f"Quantum task {task['quantumTaskArn']} for {task['deviceArn']} is
 {task['status']}")
```

Annulla attività quantistica

- `cancel_quantum_task(quantumTaskArn)`

Annulla l'attività quantistica specificata.

```
# Pass the quantum task ARN when sending the request and capture the response
response = braket.cancel_quantum_task(quantumTaskArn='arn:aws:braket:us-
west-1:123456789012:quantum-task/ce78c429-cef5-45f2-88da-123456789012')

print(f"Quantum task {response['quantumTaskArn']} is {response['cancellationStatus']}")
```

Configura AWS CLI profili per Boto3 e Amazon Braket SDK

Amazon Braket SDK si basa sull'impostazione predefinita AWS CLI credenziali, a meno che tu non specifichi esplicitamente il contrario. Ti consigliamo di mantenere l'impostazione predefinita quando

esegui su un notebook Amazon Braket gestito perché devi fornire un IAM ruolo con le autorizzazioni per avviare l'istanza del notebook.

Facoltativamente, se esegui il codice localmente (su un'EC2istanza Amazon, ad esempio), puoi stabilire named AWS CLI profili. È possibile assegnare a ciascun profilo un set di autorizzazioni diverso, anziché sovrascrivere regolarmente il profilo predefinito.

Questa sezione fornisce una breve spiegazione di come configurarlo CLI `profile` e di come incorporarlo in Amazon Staffa in modo che API le chiamate vengono effettuate con le autorizzazioni di quel profilo.

In questa sezione:

- Fase 1: Configurare un locale AWS CLI profile
- Passaggio 2: stabilire un oggetto di sessione Boto3
- Fase 3: incorporare la sessione Boto3 nel Braket AwsSession

Fase 1: Configurare un locale AWS CLI **`profile`**

Spiegare come creare un utente e come configurare un profilo non predefinito non rientra nell'ambito di questo documento. Per informazioni su questi argomenti, consulta:

- Nozioni di base
- Configurazione del AWS CLI usare AWS IAM Identity Center

Per utilizzare Amazon Braket, devi fornire a questo utente e all'associato le CLI `profile` autorizzazioni Braket necessarie. Ad esempio, puoi allegare la politica. AmazonBraketFullAccess

Passaggio 2: stabilire un oggetto di sessione Boto3

Per stabilire un oggetto di sessione Boto3, utilizzate il seguente esempio di codice.

```
from boto3 import Session

# Insert CLI profile name here
boto_sess = Session(profile_name=`profile`)
```

> **Note**
>
> Se il previsto API le chiamate hanno restrizioni basate sulla regione che non sono allineate alla regione `profile` predefinita, è possibile specificare una regione per la sessione Boto3 come mostrato nell'esempio seguente.

```
# Insert CLI profile name _and_ region
boto_sess = Session(profile_name=`profile`, region_name=`region`)
```

All'argomento designato come`region`, sostituite un valore che corrisponda a uno dei Regioni AWS in cui Amazon Braket è disponibile come `us-east-1``us-west-1`, e così via.

Fase 3: incorporare la sessione Boto3 nel Braket AwsSession

L'esempio seguente mostra come inizializzare una sessione Boto3 Braket e creare un'istanza di un dispositivo in quella sessione.

```
from braket.aws import AwsSession, AwsDevice

# Initialize Braket session with Boto3 Session credentials
aws_session = AwsSession(boto_session=boto_sess)

# Instantiate any Braket QPU device with the previously initiated AwsSession
sim_arn = 'arn:aws:braket:::device/quantum-simulator/amazon/sv1'
device = AwsDevice(sim_arn, aws_session=aws_session)
```

Una volta completata questa configurazione, è possibile inviare attività quantistiche a quell'`AwsDevice`oggetto istanziato (ad esempio chiamando il comando). `device.run(…)` Tutti API le chiamate effettuate da quel dispositivo possono sfruttare le IAM credenziali associate al CLI profilo precedentemente designato. `profile`

Verifica le tue attività quantistiche con Amazon Braket

Braket fornisce l'accesso a simulatori di circuiti quantistici completamente gestiti e ad alte prestazioni. Puoi testare e convalidare i tuoi circuiti. Braket gestisce tutti i componenti software sottostanti e i cluster Amazon Elastic Compute Cloud EC2 (Amazon) per eliminare l'onere della simulazione di circuiti quantistici sulla classica infrastruttura di calcolo ad alte prestazioni (). HPC

In questa sezione:

- Invio di attività quantistiche ai simulatori
- Lavorare con Amazon Braket Hybrid Jobs

Invio di attività quantistiche ai simulatori

Amazon Braket fornisce l'accesso a diversi simulatori in grado di testare le tue attività quantistiche. Puoi inviare attività quantistiche singolarmente o configurare attività quantistiche in batch.

Simulatori

- Simulatore di matrici di densità, DM1 : `arn:aws:braket:::device/quantum-simulator/amazon/dm1`
- Simulatore vettoriale di stato, SV1 : `arn:aws:braket:::device/quantum-simulator/amazon/sv1`
- Simulatore di rete Tensor, TN1 : `arn:aws:braket:::device/quantum-simulator/amazon/tn1`
- Il simulatore locale: `LocalSimulator()`

> **Note**
>
> È possibile annullare le attività quantistiche `CREATED` nello stato QPUs e nei simulatori su richiesta. È possibile annullare le attività quantistiche `QUEUED` nello stato nel miglior modo possibile per i simulatori su richiesta e. QPUs Tieni presente che è improbabile che le attività QPU `QUEUED` quantistiche vengano annullate correttamente durante le finestre di disponibilità. QPU

In questa sezione:

Simulatore vettoriale dello stato locale () **`braket_sv`**

Il simulatore vettoriale dello stato locale (`braket_sv`) fa parte di Amazon SDK Braket che viene eseguito localmente nel tuo ambiente. È ideale per la prototipazione rapida su piccoli circuiti (fino a 25 qubits) a seconda delle specifiche hardware dell'istanza del notebook Braket o dell'ambiente locale.

Il simulatore locale supporta tutti i gate di Amazon SDK Braket, QPU ma i dispositivi supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo.

> **Note**
>
> Il simulatore locale supporta QASM funzionalità Open avanzate che potrebbero non essere supportate su QPU dispositivi o altri simulatori. Per ulteriori informazioni sulle funzionalità supportate, consultate gli esempi forniti nel notebook Open QASM Local Simulator.

Per ulteriori informazioni su come lavorare con i simulatori, consulta gli esempi di Amazon Braket.

Simulatore di matrici di densità locale () `braket_dm`

Il simulatore di matrice di densità locale (`braket_dm`) fa parte di Amazon Braket SDK che funziona localmente nel tuo ambiente. È ideale per la prototipazione rapida su piccoli circuiti con rumore (fino a 12). qubits) a seconda delle specifiche hardware dell'istanza del notebook Braket o dell'ambiente locale.

È possibile creare circuiti rumorosi comuni partendo da zero utilizzando operazioni di gate noise come bit-flip e depolarizing error. È inoltre possibile applicare operazioni di rumore a specifiche qubits e porte di circuiti esistenti progettati per funzionare con e senza rumore.

Il simulatore `braket_dm` locale può fornire i seguenti risultati, dato il numero specificato di shots:

- Matrice a densità ridotta: Shots = 0

> **Note**
>
> Il simulatore locale supporta QASM funzionalità Open avanzate, che potrebbero non essere supportate su QPU dispositivi o altri simulatori. Per ulteriori informazioni sulle funzionalità supportate, consultate gli esempi forniti nel notebook Open QASM Local Simulator.

Per ulteriori informazioni sul simulatore di matrice di densità locale, consultate l'esempio del simulatore di rumore introduttivo di Braket.

Simulatore locale () AHS `braket_ahs`

Il simulatore locale AHS (Analog Hamiltonian Simulation) (`braket_ahs`) fa parte di Amazon Braket che viene eseguito localmente nel tuo ambiente. SDK Può essere usato per simulare i risultati di un programma. AHS È ideale per la prototipazione su registri di piccole dimensioni (fino a 10-12 atomi) a seconda delle specifiche hardware dell'istanza del notebook Braket o dell'ambiente locale.

Il simulatore locale supporta AHS programmi con un campo di guida uniforme, un campo di spostamento (non uniforme) e disposizioni atomiche arbitrarie. Per i dettagli, fate riferimento alla AHSclasse Braket e allo schema del programma Braket. AHS

Per ulteriori informazioni sul AHS simulatore locale, consultate la pagina HelloAHS: Run your first Analog Hamiltonian Simulation e i taccuini di esempio Analog Hamiltonian Simulation.

Simulatore vettoriale di stato (SV1)

SV1 è un simulatore vettoriale di stato universale su richiesta, ad alte prestazioni. Può simulare circuiti fino a 34 qubits. Puoi aspettarti un 34-qubitIl completamento del circuito, denso e quadrato (profondità del circuito = 34) richiederà circa 1-2 ore, a seconda del tipo di porte utilizzate e di altri fattori. I circuiti con all-to-all porte sono adatti per SV1. Restituisce risultati in forme come un vettore di stato completo o una matrice di ampiezze.

SV1 ha una durata massima di 6 ore. Ha un valore predefinito di 35 attività quantistiche simultanee e un massimo di 100 (50 in us-west-1 e eu-west-2) attività quantistiche simultanee.

SV1 risultati

SV1 può fornire i seguenti risultati, dato il numero specificato di shots:

- Esempio: Shots > 0
- Aspettativa: Shots >= 0
- Varianza: Shots >= 0
- Probabilità: Shots > 0
- Ampiezza: Shots = 0
- Gradiente aggiunto: Shots = 0

Per ulteriori informazioni sui risultati, consulta Tipi di risultati.

SV1 è sempre disponibile, fa funzionare i circuiti su richiesta e può far funzionare più circuiti in parallelo. Il runtime si ridimensiona linearmente con il numero di operazioni e in modo esponenziale con il numero di qubits. Il numero di shots ha un impatto minimo sul tempo di esecuzione. Per saperne di più, visita Compare simulators.

I simulatori supportano tutti i gate del BraketSDK, ma QPU i dispositivi supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo.

Simulatore di matrici di densità (DM1)

DM1 è un simulatore di matrici di densità su richiesta e ad alte prestazioni. Può simulare circuiti fino a 17 qubits.

DM1 ha un'autonomia massima di 6 ore, un valore predefinito di 35 attività quantistiche simultanee e un massimo di 50 attività quantistiche simultanee.

DM1 risultati

DM1 può fornire i seguenti risultati, dato il numero specificato di shots:

- Esempio: Shots > 0
- Aspettativa: Shots >= 0
- Varianza: Shots >= 0
- Probabilità: Shots > 0
- Matrice a densità ridotta: Shots = 0, fino a un massimo di 8 qubits

Per ulteriori informazioni sui risultati, vedere Tipi di risultati.

DM1 è sempre disponibile, fa funzionare i circuiti su richiesta e può far funzionare più circuiti in parallelo. Il runtime si ridimensiona linearmente con il numero di operazioni e in modo esponenziale con il numero di qubits. Il numero di shots ha un impatto minimo sul tempo di esecuzione. Per saperne di più, consulta Confronta i simulatori.

Cancelli antirumore e limitazioni

```
AmplitudeDamping
    Probability has to be within [0,1]
BitFlip
    Probability has to be within [0,0.5]
Depolarizing
    Probability has to be within [0,0.75]
GeneralizedAmplitudeDamping
    Probability has to be within [0,1]
PauliChannel
    The sum of the probabilities has to be within [0,1]
Kraus
    At most 2 qubits
    At most 4 (16) Kraus matrices for 1 (2) qubit
PhaseDamping
    Probability has to be within [0,1]
PhaseFlip
    Probability has to be within [0,0.5]
TwoQubitDephasing
    Probability has to be within [0,0.75]
TwoQubitDepolarizing
    Probability has to be within [0,0.9375]
```

Simulatore di rete Tensor (TN1)

TN1 è un simulatore di rete tensoriale su richiesta e ad alte prestazioni. TN1 può simulare determinati tipi di circuiti con un massimo di 50 qubits e una profondità del circuito pari o inferiore a 1.000. TN1 è particolarmente potente per circuiti sparsi, circuiti con porte locali e altri circuiti con struttura speciale, come i circuiti a trasformata quantistica di Fourier (). QFT TN1 funziona in due fasi. Innanzitutto, la fase di prova tenta di identificare un percorso computazionale efficiente per il circuito, quindi TN1 può stimare la durata della fase successiva, chiamata fase di contrazione. Se il tempo di contrazione stimato supera il TN1 limite di durata della simulazione, TN1 non tenta la contrazione.

TN1 ha un limite di durata di 6 ore. È limitato a un massimo di 10 (5 in eu-west-2) compiti quantistici simultanei.

TN1 risultati

La fase di contrazione consiste in una serie di moltiplicazioni di matrici. La serie di moltiplicazioni continua fino al raggiungimento di un risultato o fino a quando non si determina che un risultato non può essere raggiunto.

Nota: Shots deve essere > 0.

I tipi di risultati includono:

- Project N.E.M.O.
- Aspettativa
- Varianza

Per ulteriori informazioni sui risultati, consulta Tipi di risultati.

TN1 è sempre disponibile, fa funzionare i circuiti su richiesta e può far funzionare più circuiti in parallelo. Per ulteriori informazioni, consulta Confronta i simulatori.

I simulatori supportano tutti i gate del BraketSDK, ma QPU i dispositivi supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo.

Visita il Amazon GitHub Archivio Braket per un taccuino di TN1 esempio con cui iniziare TN1.

Le migliori pratiche per lavorare con TN1

- Evita i all-to-all circuiti.

- Prova un nuovo circuito o una nuova classe di circuiti con un numero limitato di shots, per conoscere la «durezza» del circuito per TN1.
- Split grande shot simulazioni su più attività quantistiche.

Simulatori integrati

I simulatori integrati funzionano incorporando la simulazione con il codice dell'algoritmo nello stesso contenitore ed eseguendo la simulazione direttamente sull'istanza di lavoro ibrida. Ciò può essere utile per eliminare i colli di bottiglia associati alla comunicazione della simulazione con un dispositivo remoto. Ciò può comportare un utilizzo della memoria notevolmente inferiore, un numero ridotto di esecuzioni dei circuiti per ottenere il risultato desiderato e un miglioramento delle prestazioni di dieci volte o più. Per ulteriori informazioni sui simulatori incorporati, consulta la pagina Esegui un lavoro ibrido con Amazon Braket Hybrid Jobs.

PennyLanedei simulatori di fulmini

Puoi usare i simulatori PennyLane di fulmini come simulatori integrati su Braket. Con i simulatori PennyLane di fulmini, puoi sfruttare metodi avanzati di calcolo del gradiente, come la differenziazione aggiuntiva, per valutare i gradienti più velocemente. Il simulatore lightning.qubit è disponibile come dispositivo tramite Braket NBIs e come simulatore integrato, mentre il simulatore lightning.gpu deve essere eseguito come simulatore integrato con un'istanza. GPU Vedi i simulatori incorporati nel notebook Braket Hybrid Jobs per un esempio di utilizzo di lightning.gpu.

Confronta i simulatori

Questa sezione ti aiuta a selezionare il simulatore Amazon Braket più adatto al tuo compito quantistico, descrivendo alcuni concetti, limitazioni e casi d'uso.

Scelta tra simulatori locali e simulatori on-demand (SV1, TN1, DM1)

Le prestazioni dei simulatori locali dipendono dall'hardware che ospita l'ambiente locale, ad esempio un'istanza del notebook Braket, utilizzata per eseguire il simulatore. I simulatori su richiesta vengono eseguiti in AWS cloud e sono progettati per scalare oltre i tipici ambienti locali. I simulatori on-demand sono ottimizzati per circuiti più grandi, ma aggiungono un sovraccarico di latenza per attività quantistiche o batch di attività quantistiche. Ciò può comportare un compromesso se sono coinvolte molte attività quantistiche. Date queste caratteristiche prestazionali generali, le seguenti linee guida possono aiutarvi a scegliere come eseguire le simulazioni, comprese quelle con rumore.

Per le simulazioni:

- Quando si impiegano meno di 18 qubits, usa un simulatore locale.
- Quando si impiegano 18-24 anni qubits, scegli un simulatore in base al carico di lavoro.
- Quando ne impiegano più di 24 qubits, usa un simulatore su richiesta.

Per le simulazioni del rumore:

- Quando si impiegano meno di 9 qubits, usa un simulatore locale.
- Quando si impiegano 9-12 qubits, scegli un simulatore in base al carico di lavoro.
- Quando ne impiegano più di 12 qubits, usa DM1.

Cos'è un simulatore vettoriale di stato?

SV1 è un simulatore vettoriale di stato universale. Memorizza la funzione d'onda completa dello stato quantistico e applica in sequenza le operazioni di gate allo stato. Memorizza tutte le possibilità, anche quelle estremamente improbabili. Il SV1 il tempo di esecuzione del simulatore per un compito quantistico aumenta linearmente con il numero di porte nel circuito.

Cos'è un simulatore di matrici di densità?

DM1 simula circuiti quantistici con rumore. Memorizza la matrice a piena densità del sistema e applica in sequenza le porte e le operazioni di rumorosità del circuito. La matrice di densità finale contiene informazioni complete sullo stato quantistico dopo il funzionamento del circuito. Il runtime generalmente scala linearmente con il numero di operazioni e in modo esponenziale con il numero di qubits.

Cos'è un simulatore di rete tensoriale?

TN1 codifica i circuiti quantistici in un grafo strutturato.

- I nodi del grafo sono costituiti da porte quantistiche, oppure qubits.
- I bordi del grafico rappresentano le connessioni tra le porte.

Come risultato di questa struttura, TN1 può trovare soluzioni simulate per circuiti quantistici relativamente grandi e complessi.

TN1 richiede due fasi

In genere, TN1 opera secondo un approccio in due fasi alla simulazione del calcolo quantistico.

- La fase di prova: in questa fase, TN1 trova un modo per attraversare il grafico in modo efficiente, che prevede la visita di ogni nodo in modo da poter ottenere la misurazione desiderata. Come cliente, non vedi questa fase perché TN1 esegue entrambe le fasi insieme per te. Completa la prima fase e determina se eseguire la seconda fase da solo in base a vincoli pratici. Una volta iniziata la simulazione, non avete alcun contributo in merito a tale decisione.
- La fase di contrazione: questa fase è analoga alla fase di esecuzione di un calcolo in un computer classico. La fase consiste in una serie di moltiplicazioni matriciali. L'ordine di queste moltiplicazioni ha un grande effetto sulla difficoltà del calcolo. Pertanto, la fase di prova viene prima completata per trovare i percorsi di calcolo più efficaci sul grafico. Dopo aver trovato il percorso di contrazione durante la fase di prova, TN1 unisce le porte del circuito per produrre i risultati della simulazione.

TN1 i grafici sono analoghi a una mappa

Metaforicamente, puoi confrontare il sottostante TN1 grafico delle strade di una città. In una città con una griglia pianificata, è facile trovare un percorso verso la destinazione utilizzando una mappa. In una città con strade non pianificate, nomi di strade duplicati e così via, può essere difficile trovare un percorso verso la destinazione guardando una mappa.

Se TN1 non hai eseguito la fase di prova, sarebbe come camminare per le strade della città per trovare la tua destinazione, invece di guardare prima una mappa. Passare più tempo a guardare la mappa può davvero ripagare in termini di tempo a piedi. Allo stesso modo, la fase di prova fornisce informazioni preziose.

Si potrebbe dire che TN1 ha una certa «consapevolezza» della struttura del circuito sottostante che attraversa. Acquisisce questa consapevolezza durante la fase di prova.

Tipi di problemi più adatti a ciascuno di questi tipi di simulatori

SV1 è adatto a qualsiasi classe di problemi che si basano principalmente sull'avere un certo numero di qubits e cancelli. In genere, il tempo necessario cresce linearmente con il numero di porte, mentre non dipende dal numero di shots. SV1 è generalmente più veloce di TN1 per circuiti inferiori a 28 qubits.

SV1 può essere più lento se è più alto qubit numeri perché simula effettivamente tutte le possibilità, anche quelle estremamente improbabili. Non ha modo di determinare quali risultati siano probabili. Quindi, per un 30-qubit valutazione, SV1 deve calcolare 2^30 configurazioni. Il limite di 34 qubits per il Amazon Braket SV1 il simulatore è un vincolo pratico dovuto alle limitazioni di memoria e archiviazione. Puoi pensarlo in questo modo: ogni volta che aggiungi un qubit in SV1, il problema diventa due volte più difficile.

Per molte classi di problemi, TN1 può valutare circuiti molto più grandi in tempi realistici rispetto a SV1 perché TN1 sfrutta la struttura del grafico. Tiene essenzialmente traccia dell'evoluzione delle soluzioni dal punto di partenza e conserva solo le configurazioni che contribuiscono a un'attraversamento efficiente. In altre parole, salva le configurazioni per creare un ordinamento di moltiplicazione delle matrici che si traduce in un processo di valutazione più semplice.

In TN1, il numero di qubits e le porte sono importanti, ma la struttura del grafico conta molto di più. Ad esempio, TN1 è molto bravo a valutare circuiti (grafici) in cui le porte sono a corto raggio (cioè ciascuna qubit è collegato tramite cancelli solo al vicino più prossimo qubits) e circuiti (grafici) in cui le connessioni (o porte) hanno un intervallo simile. Un intervallo tipico per TN1 sta avendo ciascuno qubit parla solo con gli altri qubits sono 5 qubits lontano. Se la maggior parte della struttura può essere scomposta in relazioni più semplici come queste, che possono essere rappresentate in matrici più, più piccole o più uniformi, TN1 esegue la valutazione con facilità.

Limitazioni di TN1

TN1 può essere più lento di SV1 a seconda della complessità strutturale del grafico. Per alcuni grafici, TN1 termina la simulazione dopo la fase di prova e mostra uno stato di`FAILED`, per uno dei due motivi seguenti:

- Impossibile trovare un percorso: se il grafico è troppo complesso, è troppo difficile trovare un buon percorso trasversale e il simulatore rinuncia al calcolo. TN1 non può eseguire la contrazione. È possibile che venga visualizzato un messaggio di errore simile a questo: `No viable contraction path found.`
- La fase di contrazione è troppo difficile: in alcuni grafici, TN1 può trovare un percorso trasversale, ma la valutazione è molto lunga e richiede molto tempo. In questo caso, la contrazione è così costosa che il costo sarebbe proibitivo e invece TN1 esce dopo la fase di prova. È possibile che venga visualizzato un messaggio di errore simile a questo: `Predicted runtime based on best contraction path found exceeds TN1 limit.`

> **Note**
>
> Ti verrà addebitato il costo della fase di prova di TN1 anche se la contrazione non viene eseguita e viene visualizzato uno stato. `FAILED`

Il tempo di esecuzione previsto dipende anche dal shot contare. Nel peggiore dei casi, TN1 il tempo di contrazione dipende linearmente dal shot contare. Il circuito può essere contrattabile con meno

shots. Ad esempio, potresti inviare un compito quantistico con 100 shots, quale TN1 decide non è contrattabile, ma se lo invii nuovamente con solo 10, la contrazione procede. In questa situazione, per ottenere 100 campioni, è possibile inviare 10 compiti quantistici su 10 shots per lo stesso circuito e alla fine combina i risultati.

Come buona pratica, ti consigliamo di testare sempre il tuo circuito o la tua classe di circuito con alcuni shots (ad esempio, 10) per scoprire la resistenza del circuito TN1, prima di procedere con un numero maggiore di shots.

> **Note**
>
> La serie di moltiplicazioni che forma la fase di contrazione inizia con piccole matrici NxN. Ad esempio, un 2-qubit gate richiede una matrice 4x4. Le matrici intermedie necessarie durante una contrazione giudicata troppo difficile sono gigantesche. Un calcolo di questo tipo richiederebbe giorni per essere completato. Ecco perché Amazon Braket non tenta contrazioni estremamente complesse.

Concurrency (Simultaneità)

Tutti i simulatori Braket offrono la possibilità di eseguire più circuiti contemporaneamente. I limiti di concorrenza variano in base al simulatore e alla regione. Per ulteriori informazioni sui limiti di concorrenza, consulta la pagina Quote.

Esempi di attività quantistiche su Amazon Braket

Questa sezione illustra le fasi dell'esecuzione di un esempio di task quantistico, dalla selezione del dispositivo alla visualizzazione del risultato. Come best practice per Amazon Braket, ti consigliamo di iniziare eseguendo il circuito su un simulatore, ad esempio SV1.

In questa sezione:

- Specificare il dispositivo
- Invia un esempio di attività quantistica
- Invia un'attività parametrizzata
- Specificare shots
- Sondaggio per visualizzare i risultati
- Visualizza i risultati di esempio

Specificare il dispositivo

Innanzitutto, selezionate e specificate il dispositivo per il vostro compito quantistico. Questo esempio mostra come scegliere il simulatore, SV1.

```
# choose the on-demand simulator to run the circuit
from braket.aws import AwsDevice
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
```

È possibile visualizzare alcune delle proprietà di questo dispositivo come segue:

```
print (device.name)
for iter in device.properties.action['braket.ir.jaqcd.program']:
    print(iter)
```

```
SV1
('version', ['1.0', '1.1'])
('actionType', <DeviceActionType.JAQCD: 'braket.ir.jaqcd.program'>)
('supportedOperations', ['ccnot', 'cnot', 'cphaseshift', 'cphaseshift00',
 'cphaseshift01', 'cphaseshift10', 'cswap', 'cy', 'cz', 'h', 'i', 'iswap', 'pswap',
 'phaseshift', 'rx', 'ry', 'rz', 's', 'si', 'swap', 't', 'ti', 'unitary', 'v', 'vi',
 'x', 'xx', 'xy', 'y', 'yy', 'z', 'zz'])
('supportedResultTypes', [ResultType(name='Sample', observables=['x', 'y', 'z', 'h',
 'i', 'hermitian'], minShots=1, maxShots=100000), ResultType(name='Expectation',
 observables=['x', 'y', 'z', 'h', 'i', 'hermitian'], minShots=0, maxShots=100000),
 ResultType(name='Variance', observables=['x', 'y', 'z', 'h', 'i', 'hermitian'],
 minShots=0, maxShots=100000), ResultType(name='Probability', observables=None,
 minShots=1, maxShots=100000), ResultType(name='Amplitude', observables=None,
 minShots=0, maxShots=0)])
```

Invia un esempio di attività quantistica

Invia un esempio di attività quantistica da eseguire sul simulatore on-demand.

```
# create a circuit with a result type
circ = Circuit().rx(0, 1).ry(1, 0.2).cnot(0,2).variance(observable=Observable.Z(),
 target=0)
# add another result type
circ.probability(target=[0, 2])
```

```
# set up S3 bucket (where results are stored)
my_bucket = "amazon-braket-your-s3-bucket-name" # the name of the bucket
my_prefix = "your-folder-name" # the name of the folder in the bucket
s3_location = (my_bucket, my_prefix)

# submit the quantum task to run
my_task = device.run(circ, s3_location, shots=1000, poll_timeout_seconds = 100,
 poll_interval_seconds = 10)
# the positional argument for the S3 bucket is optional if you want to specify a bucket
 other than the default

# get results of the quantum task
result = my_task.result()
```

Il `device.run()` comando crea un'attività quantistica tramite. `CreateQuantumTask` API Dopo un breve periodo di inizializzazione, l'operazione quantistica viene messa in coda fino a quando non esiste la capacità necessaria per eseguirla su un dispositivo. In questo caso, il dispositivo è SV1. Dopo che il dispositivo ha completato il calcolo, Amazon Braket scrive i risultati nella posizione Amazon S3 specificata nella chiamata. L'argomento posizionale `s3_location` è obbligatorio per tutti i dispositivi tranne il simulatore locale.

> **Note**
>
> L'azione del task quantistico di Braket ha una dimensione limitata a 3 MB.

Invia un'attività parametrizzata

I simulatori locali e su richiesta di Amazon Braket supportano QPUs anche la specificazione di valori di parametri liberi al momento dell'invio delle attività. Puoi farlo utilizzando l'`inputs`argomento to`device.run()`, come mostrato nell'esempio seguente. `inputs`Deve essere un dizionario di coppie string-float, in cui le chiavi sono i nomi dei parametri.

La compilazione parametrica può migliorare le prestazioni di esecuzione di circuiti parametrici in alcuni casi. QPUs Quando invia un circuito parametrico come attività quantistica a un supportoQPU, Braket compilerà il circuito una volta e memorizzerà il risultato nella cache. Non è prevista alcuna ricompilazione per i successivi aggiornamenti dei parametri sullo stesso circuito, con conseguente tempi di esecuzione più rapidi per le attività che utilizzano lo stesso circuito. Braket utilizza automaticamente i dati di calibrazione aggiornati del fornitore di hardware durante la compilazione del circuito per garantire risultati della massima qualità.

> **Note**
>
> La compilazione parametrica è supportata su tutti i moduli superconduttori basati su gate QPUs Rigetti Computing ad eccezione dei programmi a livello di impulso.

```
from braket.circuits import Circuit, FreeParameter, Observable

# create the free parameters
alpha = FreeParameter('alpha')
beta = FreeParameter('beta')

# create a circuit with a result type
circ = Circuit().rx(0, alpha).ry(1, alpha).cnot(0,2).xx(0, 2, beta)
circ.variance(observable=Observable.Z(), target=0)
# add another result type
circ.probability(target=[0, 2])
# submit the quantum task to run
my_task = device.run(circ, inputs={'alpha': 0.1, 'beta':0.2})
```

Specificare shots

Il shots argomento si riferisce al numero di misurazioni desiderate shots. Simulatori come SV1 supporta due modalità di simulazione.

- In shots = 0, il simulatore esegue una simulazione esatta, restituendo i valori reali per tutti i tipi di risultati. (Non disponibile su TN1.)
- Per valori diversi da zero di shots, il simulatore campiona i campioni dalla distribuzione di uscita per emulare il shot rumore del reale. QPUs QPUi dispositivi lo consentono solo shots > 0.

Per informazioni sul numero massimo di scatti per operazione quantistica, consulta Braket Quotas.

Sondaggio per visualizzare i risultati

Durante l'esecuzione`my_task.result()`, SDK inizia il polling di un risultato con i parametri definiti al momento della creazione dell'attività quantistica:

- `poll_timeout_seconds`è il numero di secondi necessari per eseguire il polling dell'attività quantistica prima che scada durante l'esecuzione dell'attività quantistica sul simulatore e/o sui dispositivi on-demand. QPU Il valore predefinito è 432.000 secondi, ovvero 5 giorni.

- Nota: per QPU dispositivi come Rigetti e IonQ, ti consigliamo di attendere alcuni giorni. Se il timeout per i sondaggi è troppo breve, è possibile che i risultati non vengano restituiti entro il tempo di votazione. Ad esempio, quando a non QPU è disponibile, viene restituito un errore di timeout locale.
- `poll_interval_seconds`è la frequenza con cui viene interrogato il task quantistico. Specifica la frequenza con cui si chiama il Braket API per ottenere lo stato quando l'attività quantistica viene eseguita sul simulatore on-demand e sui dispositivi. QPU Il valore predefinito è 1 secondo.

Questa esecuzione asincrona facilita l'interazione con QPU dispositivi che non sono sempre disponibili. Ad esempio, un dispositivo potrebbe non essere disponibile durante una normale finestra di manutenzione.

Il risultato restituito contiene una serie di metadati associati al task quantistico. È possibile controllare il risultato della misurazione con i seguenti comandi:

```
print('Measurement results:\n',result.measurements)
print('Counts for collapsed states:\n',result.measurement_counts)
print('Probabilities for collapsed states:\n',result.measurement_probabilities)
```

```
Measurement results:
 [[1 0 1]
 [0 0 0]
 [1 0 1]
 ...
 [0 0 0]
 [0 0 0]
 [0 0 0]]
Counts for collapsed states:
 Counter({'000': 761, '101': 226, '010': 10, '111': 3})
Probabilities for collapsed states:
 {'101': 0.226, '000': 0.761, '111': 0.003, '010': 0.01}
```

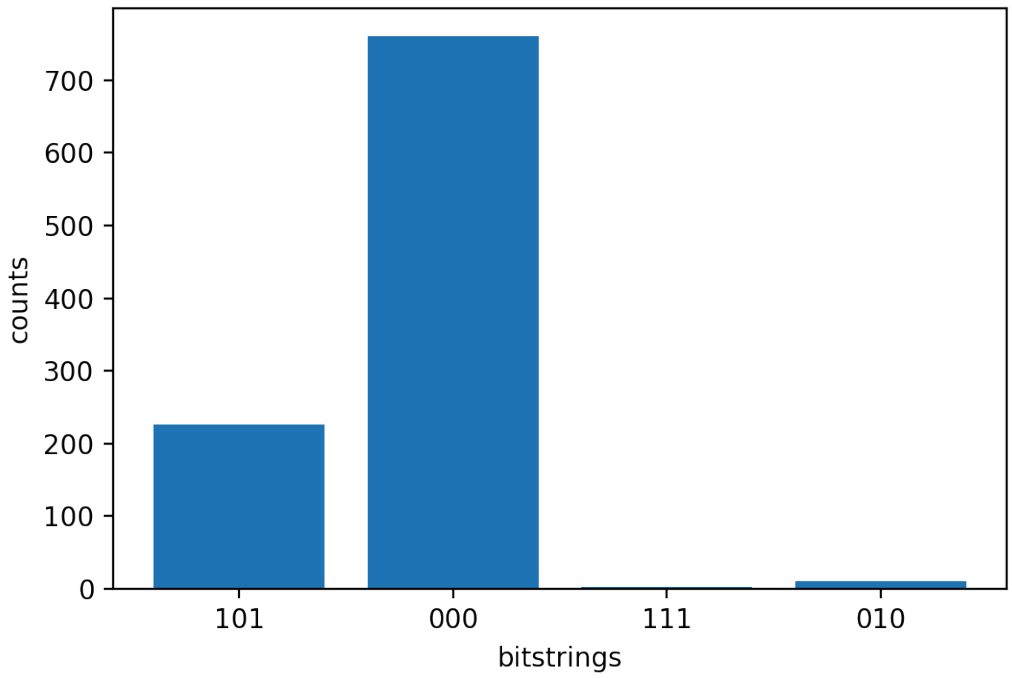
Visualizza i risultati di esempio

Poiché hai anche specificato il`ResultType`, puoi visualizzare i risultati restituiti. I tipi di risultati vengono visualizzati nell'ordine in cui sono stati aggiunti al circuito.

```
print('Result types include:\n', result.result_types)
print('Variance=',result.values[0])
```

```
print('Probability=',result.values[1])

# you can plot the result and do some analysis
import matplotlib.pyplot as plt
plt.bar(result.measurement_counts.keys(), result.measurement_counts.values());
plt.xlabel('bitstrings');
plt.ylabel('counts');
```

```
Result types include:
 [ResultTypeValue(type={'observable': ['z'], 'targets': [0], 'type': 'variance'},
 value=0.7062359999999999), ResultTypeValue(type={'targets': [0, 2], 'type':
 'probability'}, value=array([0.771, 0.   , 0.   , 0.229]))]
Variance= 0.7062359999999999
Probability= [0.771 0.    0.    0.229]
```



Test di un compito quantistico con il simulatore locale

Puoi inviare attività quantistiche direttamente a un simulatore locale per la prototipazione e il test rapidi. Questo simulatore viene eseguito nel tuo ambiente locale, quindi non è necessario specificare una posizione Amazon S3. I risultati vengono calcolati direttamente nella sessione. Per eseguire un'attività quantistica sul simulatore locale, è necessario specificare solo shots parametro.

> **Note**
>
> La velocità di esecuzione e il numero massimo di qubits l'elaborazione del simulatore locale dipende dal tipo di istanza del notebook Amazon Braket o dalle specifiche hardware locali.

I seguenti comandi sono tutti identici e istanziano il simulatore locale State Vector (Noise Free).

```
# import the LocalSimulator module
from braket.devices import LocalSimulator
# the following are identical commands
device = LocalSimulator()
device = LocalSimulator("default")
device = LocalSimulator(backend="default")
device = LocalSimulator(backend="braket_sv")
```

Quindi esegui un'attività quantistica con quanto segue.

```
my_task = device.run(circ, shots=1000)
```

Per creare un'istanza del simulatore di matrice di densità locale (rumore), i clienti modificano il backend come segue.

```
# import the LocalSimulator module
from braket.devices import LocalSimulator
device = LocalSimulator(backend="braket_dm")
```

Misurazione di qubit specifici sul simulatore locale

Il simulatore vettoriale statale locale e il simulatore di matrice di densità locale supportano circuiti in esecuzione in cui è possibile misurare un sottoinsieme dei qubit del circuito, operazione spesso denominata misurazione parziale.

Ad esempio, nel codice seguente è possibile creare un circuito a due qubit e misurare solo il primo qubit aggiungendo un'`measure`istruzione con i qubit target alla fine del circuito.

```
# Import the LocalSimulator module
from braket.devices import LocalSimulator
```

```
# Use the local simulator device
device = LocalSimulator()

# Define a bell circuit and only measure
circuit = Circuit().h(0).cnot(0, 1).measure(0)

# Run the circuit
task = device.run(circuit, shots=10)

# Get the results
result = task.result()

# Print the measurement counts for qubit 0
print(result.measurement_counts)
```

Raggruppamento quantistico delle attività

Il quantum task batching è disponibile su tutti i dispositivi Amazon Braket, ad eccezione del simulatore locale. Il batching è particolarmente utile per le attività quantistiche eseguite sui simulatori on-demand (TN1 oppure SV1) perché possono elaborare più attività quantistiche in parallelo. Per aiutarti a configurare varie attività quantistiche, Amazon Braket fornisce notebook di esempio.

Il batching consente di avviare attività quantistiche in parallelo. Ad esempio, se si desidera eseguire un calcolo che richiede 10 attività quantistiche e i circuiti utilizzati in tali attività quantistiche sono indipendenti l'uno dall'altro, è consigliabile utilizzare il batching. In questo modo, non è necessario attendere il completamento di un'attività quantistica prima di iniziare un'altra attività.

L'esempio seguente mostra come eseguire un batch di attività quantistiche:

```
circuits = [bell for _ in range(5)]
batch = device.run_batch(circuits, s3_folder, shots=100)
print(batch.results()[0].measurement_counts)  # The result of the first quantum task in
 the batch
```

Per ulteriori informazioni, consulta gli esempi di Amazon Braket GitHub o Quantum task batching, che contiene informazioni più specifiche sul batching.

In questa sezione:

- Informazioni sul raggruppamento quantistico delle attività e sui costi
- Quantum task batching e PennyLane

- Task batching e circuiti parametrizzati

Informazioni sul raggruppamento quantistico delle attività e sui costi

Alcune avvertenze da tenere a mente per quanto riguarda il raggruppamento quantistico delle attività e i costi di fatturazione:

- Per impostazione predefinita, il quantum task batching riprova ogni volta o fallisce le attività quantistiche 3 volte.
- Un batch di attività quantistiche di lunga durata, ad esempio 34 qubits for SV1, può comportare costi elevati. Assicurati di ricontrollare attentamente i valori di `run_batch` assegnazione prima di iniziare una serie di attività quantistiche. Si sconsiglia di utilizzare TN1 con`run_batch`.
- TN1 può comportare costi in caso di mancata riuscita delle attività della fase di prova (per ulteriori informazioni, consulta la TN1 descrizione). I nuovi tentativi automatici possono aumentare il costo, quindi consigliamo di impostare il numero di «max_retry» per il batch su 0 quando si utilizza TN1 (vedi Quantum Task Batching, riga 186).

Quantum task batching e PennyLane

Sfrutta i vantaggi del batch quando utilizzi PennyLane Amazon Braket `parallel = True` impostando quando crei un'istanza di un dispositivo Amazon Braket, come illustrato nell'esempio seguente.

```
device = qml.device("braket.aws.qubit",device_arn="arn:aws:braket:::device/quantum-
simulator/amazon/sv1",wires=wires,s3_destination_folder=s3_folder,parallel=True,)
```

Per ulteriori informazioni sul batching with, consulta Ottimizzazione parallelizzata dei circuiti PennyLane quantistici.

Task batching e circuiti parametrizzati

Quando si invia un batch di attività quantistiche che contiene circuiti parametrizzati, è possibile fornire un `inputs` dizionario, che viene utilizzato per tutte le attività quantistiche `list` del batch, o un dizionario di input, nel qual caso il dizionario `i` -th viene abbinato `i` al task -th, come mostrato nell'esempio seguente.

```
from braket.circuits import Circuit, FreeParameter, Observable
from braket.aws import AwsQuantumTaskBatch
```

```
# create the free parameters
alpha = FreeParameter('alpha')
beta = FreeParameter('beta')

# create two circuits
circ_a = Circuit().rx(0, alpha).ry(1, alpha).cnot(0,2).xx(0, 2, beta)
circ_a.variance(observable=Observable.Z(), target=0)

circ_b = Circuit().rx(0, alpha).rz(1, alpha).cnot(0,2).zz(0, 2, beta)
circ_b.expectation(observable=Observable.Z(), target=2)

# use the same inputs for both circuits in one batch

tasks = device.run_batch([circ_a, circ_b], inputs={'alpha': 0.1, 'beta':0.2})

# or provide each task its own set of inputs

inputs_list = [{'alpha': 0.3,'beta':0.1}, {'alpha': 0.1,'beta':0.4}]

tasks = device.run_batch([circ_a, circ_b], inputs=inputs_list)
```

È inoltre possibile preparare un elenco di dizionari di input per un singolo circuito parametrico e inviarli come batch di operazioni quantistiche. Se nell'elenco sono presenti N dizionari di input, il batch contiene N task quantistici. Il task quantistico `i` -th corrisponde al circuito eseguito con il dizionario di input -th. `i`

```
from braket.circuits import Circuit, FreeParameter

# create a parametric circuit
circ = Circuit().rx(0, FreeParameter('alpha'))

# provide a list of inputs to execute with the circuit
inputs_list = [{'alpha': 0.1}, {'alpha': 0.2}, {'alpha': 0.3}]

tasks = device.run_batch(circ, inputs=inputs_list)
```

Lavorare con Amazon Braket Hybrid Jobs

Questa sezione fornisce istruzioni di base sulla creazione e l'esecuzione di lavori ibridi in Amazon Braket.

Puoi accedere ai lavori ibridi in Braket utilizzando:

- Amazon Braket Python. SDK
- La console Amazon Braket.
- Amazon Braket API.

In questa sezione:

Esecuzione del codice locale come processo ibrido

Amazon Braket Hybrid Jobs fornisce un'orchestrazione completamente gestita di algoritmi ibridi quantistici classici, combinando le EC2 risorse di calcolo di Amazon con l'accesso ad Amazon Braket Quantum Processing Unit (). QPU Le attività quantistiche create in un processo ibrido hanno la priorità di essere messe in coda rispetto alle singole attività quantistiche, in modo che gli algoritmi non vengano interrotti dalle fluttuazioni nella coda delle attività quantistiche. Ciascuno QPU mantiene una coda di lavoro ibrida separata, garantendo che possa essere eseguito un solo lavoro ibrido alla volta.

In questa sezione:

Crea un lavoro ibrido dal codice Python locale

Puoi eseguire il codice Python locale come Amazon Braket Hybrid Job. Puoi farlo annotando il codice con un `@hybrid_job` decoratore, come mostrato nel seguente esempio di codice. Per gli ambienti

personalizzati, puoi scegliere di utilizzare un contenitore personalizzato da Amazon Elastic Container Registry (ECR).

> **Note**
>
> Per impostazione predefinita, è supportato solo Python 3.10.

È possibile utilizzare il `@hybrid_job` decoratore per annotare una funzione. Braket trasforma il codice all'interno del decoratore in uno script di algoritmo di lavoro ibrido Braket. Il lavoro ibrido richiama quindi la funzione all'interno del decoratore su un'istanza Amazon. EC2 Puoi monitorare lo stato di avanzamento del lavoro con `job.state()` o con la console Braket. Il seguente esempio di codice mostra come eseguire una sequenza di cinque stati su State Vector Simulator (SV1) device.

```
from braket.aws import AwsDevice
from braket.circuits import Circuit, FreeParameter, Observable
from braket.devices import Devices
from braket.jobs.hybrid_job import hybrid_job
from braket.jobs.metrics import log_metric

device_arn = Devices.Amazon.SV1


@hybrid_job(device=device_arn)  # choose priority device
def run_hybrid_job(num_tasks=1):
    device = AwsDevice(device_arn)  # declare AwsDevice within the hybrid job

    # create a parametric circuit
    circ = Circuit()
    circ.rx(0, FreeParameter("theta"))
    circ.cnot(0, 1)
    circ.expectation(observable=Observable.X(), target=0)

    theta = 0.0  # initial parameter

    for i in range(num_tasks):
        task = device.run(circ, shots=100, inputs={"theta": theta})  # input parameters
        exp_val = task.result().values[0]

        theta += exp_val  # modify the parameter (possibly gradient descent)

        log_metric(metric_name="exp_val", value=exp_val, iteration_number=i)
```

```
    return {"final_theta": theta, "final_exp_val": exp_val}
```

Crei il lavoro ibrido invocando la funzione come faresti con le normali funzioni di Python. Tuttavia, la funzione decorator restituisce l'handle del lavoro ibrido anziché il risultato della funzione. Per recuperare i risultati dopo il completamento, usa. `job.result()`

```
job = run_hybrid_job(num_tasks=1)
result = job.result()
```

L'argomento device nel `@hybrid_job` decoratore specifica il dispositivo a cui il lavoro ibrido ha accesso prioritario, in questo caso, SV1 simulatore. Per ottenere QPU la priorità, è necessario assicurarsi che il dispositivo ARN utilizzato all'interno della funzione corrisponda a quello specificato nel decoratore. Per comodità, è possibile utilizzare la funzione di supporto per `get_job_device_arn()` acquisire il dispositivo ARN dichiarato in. `@hybrid_job`

> **Note**
>
> Ogni processo ibrido ha un tempo di avvio di almeno un minuto poiché crea un ambiente containerizzato su Amazon. EC2 Quindi, per carichi di lavoro molto brevi, come un singolo circuito o un batch di circuiti, può essere sufficiente utilizzare attività quantistiche.

Iperparametri

La `run_hybrid_job()` funzione utilizza l'argomento `num_tasks` per controllare il numero di attività quantistiche create. Il processo ibrido lo acquisisce automaticamente come iperparametro.

> **Note**
>
> Gli iperparametri vengono visualizzati nella console Braket come stringhe, limitate a 2500 caratteri.

Metriche e registrazione

All'interno della `run_hybrid_job()` funzione, vengono registrate le metriche degli algoritmi iterativi con. `log_metrics` Le metriche vengono tracciate automaticamente nella pagina della console Braket nella scheda del lavoro ibrido. Puoi utilizzare le metriche per tracciare i costi quantistici delle

attività in tempo quasi reale durante l'esecuzione del lavoro ibrido con il tracker dei costi di Braket. L'esempio precedente utilizza il nome della metrica «probabilità» che registra la prima probabilità del tipo di risultato.

Recupero dei risultati

Una volta completato il lavoro ibrido, si utilizza `job.result()` per recuperare i risultati dei lavori ibridi. Tutti gli oggetti nell'istruzione return vengono acquisiti automaticamente da Braket. Nota che gli oggetti restituiti dalla funzione devono essere una tupla con ogni elemento serializzabile. Ad esempio, il codice seguente mostra un esempio funzionante e uno non riuscito.

```
@hybrid_job(device=Devices.Amazon.SV1)
def passing():
    np_array = np.random.rand(5)
    return np_array  # serializable

@hybrid_job(device=Devices.Amazon.SV1)
def failing():
    return MyObject() # not serializable
```

Nome del lavoro

Per impostazione predefinita, il nome di questo lavoro ibrido viene dedotto dal nome della funzione. È inoltre possibile specificare un nome personalizzato lungo fino a 50 caratteri. Ad esempio, nel codice seguente il nome del lavoro è "my-job-name».

```
@hybrid_job(device=Devices.Amazon.SV1, job_name="my-job-name")
def function():
    pass
```

modalità locale

I lavori locali vengono creati aggiungendo l'argomento `local=True` al decoratore. Questo esegue il lavoro ibrido in un ambiente containerizzato sull'ambiente di elaborazione locale, ad esempio il laptop. I lavori locali non prevedono la priorità delle code per le attività quantistiche. Per casi avanzati come i job multinodo o MPI locali possono avere accesso alle variabili di ambiente Braket richieste. Il codice seguente crea un processo ibrido locale con il dispositivo come simulatore. SV1

```
@hybrid_job(device=Devices.Amazon.SV1, local=True)
def run_hybrid_job(num_tasks = 1):
    return ...
```

Sono supportate tutte le altre opzioni di lavoro ibride. Per un elenco di opzioni, consultate il modulo braket.jobs.quantum_job_creation.

Installa pacchetti Python e codice sorgente aggiuntivi

Puoi personalizzare il tuo ambiente di runtime per usare i tuoi pacchetti Python preferiti. Puoi usare un `requirements.txt` file, un elenco di nomi di pacchetti o portare il tuo contenitore (BYOC). Per personalizzare un ambiente di runtime utilizzando un `requirements.txt` file, fate riferimento al seguente esempio di codice.

```
@hybrid_job(device=Devices.Amazon.SV1, dependencies="requirements.txt")
def run_hybrid_job(num_tasks = 1):
    return ...
```

Ad esempio, il `requirements.txt` file può includere altri pacchetti da installare.

```
qiskit
pennylane >= 0.31
mitiq == 0.29
```

In alternativa, puoi fornire i nomi dei pacchetti come elenco Python come segue.

```
@hybrid_job(device=Devices.Amazon.SV1, dependencies=["qiskit", "pennylane>=0.31",
 "mitiq==0.29"])
def run_hybrid_job(num_tasks = 1):
    return ...
```

Il codice sorgente aggiuntivo può essere specificato come elenco di moduli o come singolo modulo come nel seguente esempio di codice.

```
@hybrid_job(device=Devices.Amazon.SV1, include_modules=["my_module1", "my_module2"])
def run_hybrid_job(num_tasks = 1):
    return ...
```

Salva e carica i dati in un'istanza di lavoro ibrida

Specificazione dei dati di addestramento in ingresso

Quando crei un lavoro ibrido, puoi fornire un set di dati di addestramento di input specificando un bucket Amazon Simple Storage Service (Amazon S3). Puoi anche specificare un percorso

locale, quindi Braket carica automaticamente i dati su Amazon S3 all'indirizzo. `s3://<default_bucket_name>/jobs/<job_name>/<timestamp>/data/<channel_name>` Se specifichi un percorso locale, il nome del canale è predefinito su «input». Il codice seguente mostra un file numpy dal percorso locale. `data/file.npy`

```
@hybrid_job(device=Devices.Amazon.SV1, input_data="data/file.npy")
def run_hybrid_job(num_tasks = 1):
    data = np.load("data/file.npy")
    return ...
```

Per S3, è necessario utilizzare la funzione di `get_input_data_dir()` supporto.

```
s3_path = "s3://amazon-braket-us-west-1-961591465522/job-data/file.npy"

@hybrid_job(device=None, input_data=s3_path)
def job_s3_input():
    np.load(get_input_data_dir() + "/file.npy")

@hybrid_job(device=None, input_data={"channel": s3_path})
def job_s3_input_channel():
    np.load(get_input_data_dir("channel") + "/file.npy")
```

È possibile specificare più fonti di dati di input fornendo un dizionario dei valori dei canali e dei percorsi URIs S3 o locali.

```
input_data = {
    "input": "data/file.npy",
    "input_2": "s3://my-bucket/data.json"
}


@hybrid_job(device=None, input_data=input_data)
def multiple_input_job():
    np.load(get_input_data_dir("input") + "/file.npy")
    np.load(get_input_data_dir("input_2") + "/data.json")
```

> **Note**
>
> Quando i dati di input sono di grandi dimensioni (>1 GB), c'è un lungo tempo di attesa prima che il lavoro venga creato. Ciò è dovuto ai dati di input locali che vengono caricati per la

prima volta in un bucket S3, quindi il percorso S3 viene aggiunto alla richiesta di lavoro. Infine, la richiesta di lavoro viene inviata al servizio Braket.

Salvataggio dei risultati su S3

Per salvare i risultati non inclusi nell'istruzione return della funzione decorata, è necessario aggiungere la directory corretta a tutte le operazioni di scrittura dei file. L'esempio seguente mostra il salvataggio di un array numpy e di una figura matplotlib.

```
@hybrid_job(device=Devices.Amazon.SV1)
def run_hybrid_job(num_tasks = 1):
    result = np.random.rand(5)

    # save a numpy array
    np.save("result.npy", result)

    # save a matplotlib figure
    plt.plot(result)
    plt.savefig("fig.png")
    return ...
```

Tutti i risultati vengono compressi in un file denominato. `model.tar.gz` Puoi scaricare i risultati con la funzione `job.result()` Python o accedendo alla cartella dei risultati dalla pagina del lavoro ibrido nella console di gestione Braket.

Salvataggio e ripresa dai checkpoint

Per lavori ibridi di lunga durata, si consiglia di salvare periodicamente lo stato intermedio dell'algoritmo. È possibile utilizzare la funzione di `save_job_checkpoint()` supporto integrata o salvare i file nel percorso. `AMZN_BRAKET_JOB_RESULTS_DIR` Quest'ultima è disponibile con la funzione helper. `get_job_results_dir()`

Quello che segue è un esempio di funzionamento minimo per salvare e caricare i checkpoint con un Job Decorator ibrido:

```
from braket.jobs import save_job_checkpoint, load_job_checkpoint, hybrid_job

@hybrid_job(device=None, wait_until_complete=True)
def function():
```

```
    save_job_checkpoint({"a": 1})

job = function()
job_name = job.name
job_arn = job.arn

@hybrid_job(device=None, wait_until_complete=True, copy_checkpoints_from_job=job_arn)
def continued_function():
    load_job_checkpoint(job_name)

continued_job = continued_function()
```

Nel primo job ibrido, `save_job_checkpoint()` viene chiamato con un dizionario contenente i dati che vogliamo salvare. Per impostazione predefinita, ogni valore deve essere serializzabile come testo. Per controllare oggetti Python più complessi, come gli array numpy, puoi impostare. `data_format = PersistedJobDataFormat.PICKLED_V4` Questo codice crea e sovrascrive un file di checkpoint con nome predefinito negli artefatti del job ibrido `<jobname>.json` in una sottocartella chiamata «checkpoints».

Per creare un nuovo lavoro ibrido con cui continuare dal checkpoint, dobbiamo indicare `copy_checkpoints_from_job=job_arn` `job_arn` dov'è il lavoro ibrido del lavoro precedente. ARN Quindi eseguiamo `load_job_checkpoint(job_name)` il caricamento dal checkpoint.

Le migliori pratiche per arredatori di lavori ibridi

Abbraccia l'asincronicità

I lavori ibridi creati con l'annotazione decorator sono asincroni: vengono eseguiti non appena le risorse classiche e quantistiche sono disponibili. È possibile monitorare l'avanzamento dell'algoritmo utilizzando il Braket Management Console o Amazon CloudWatch. Quando invii l'algoritmo per l'esecuzione, Braket lo esegue in un ambiente containerizzato scalabile e i risultati vengono recuperati quando l'algoritmo è completo.

Esegui algoritmi variazionali iterativi

Hybrid jobs ti offre gli strumenti per eseguire algoritmi iterativi quantistici classici. Per problemi puramente quantistici, utilizzate attività quantistiche o una serie di attività quantistiche. L'accesso prioritario a determinati QPUs è particolarmente vantaggioso per gli algoritmi variazionali di lunga durata che richiedono più chiamate iterative a o con l'elaborazione classica intermedia. QPUs

Esegui il debug utilizzando la modalità locale

Prima di eseguire un lavoro ibrido su unQPU, si consiglia di eseguirlo prima sul simulatore SV1 per confermare che funzioni come previsto. Per i test su piccola scala, è possibile eseguirli in modalità locale per iterazioni e debug rapidi.

Migliora la riproducibilità con Bring your own container () BYOC

Crea un esperimento riproducibile incapsulando il software e le sue dipendenze in un ambiente containerizzato. Comprimendo tutto il codice, le dipendenze e le impostazioni in un contenitore, si evitano potenziali conflitti e problemi di versione.

Simulatori distribuiti a più istanze

Per eseguire un gran numero di circuiti, prendi in considerazione l'utilizzo del MPI supporto integrato per eseguire simulatori locali su più istanze all'interno di un singolo processo ibrido. Per ulteriori informazioni, consulta Simulatori incorporati.

Usa circuiti parametrici

I circuiti parametrici inviati da un processo ibrido vengono compilati automaticamente su determinati circuiti QPUs utilizzando la compilazione parametrica per migliorare i tempi di esecuzione degli algoritmi.

Checkpoint periodicamente

Per lavori ibridi di lunga durata, si consiglia di salvare periodicamente lo stato intermedio dell'algoritmo.

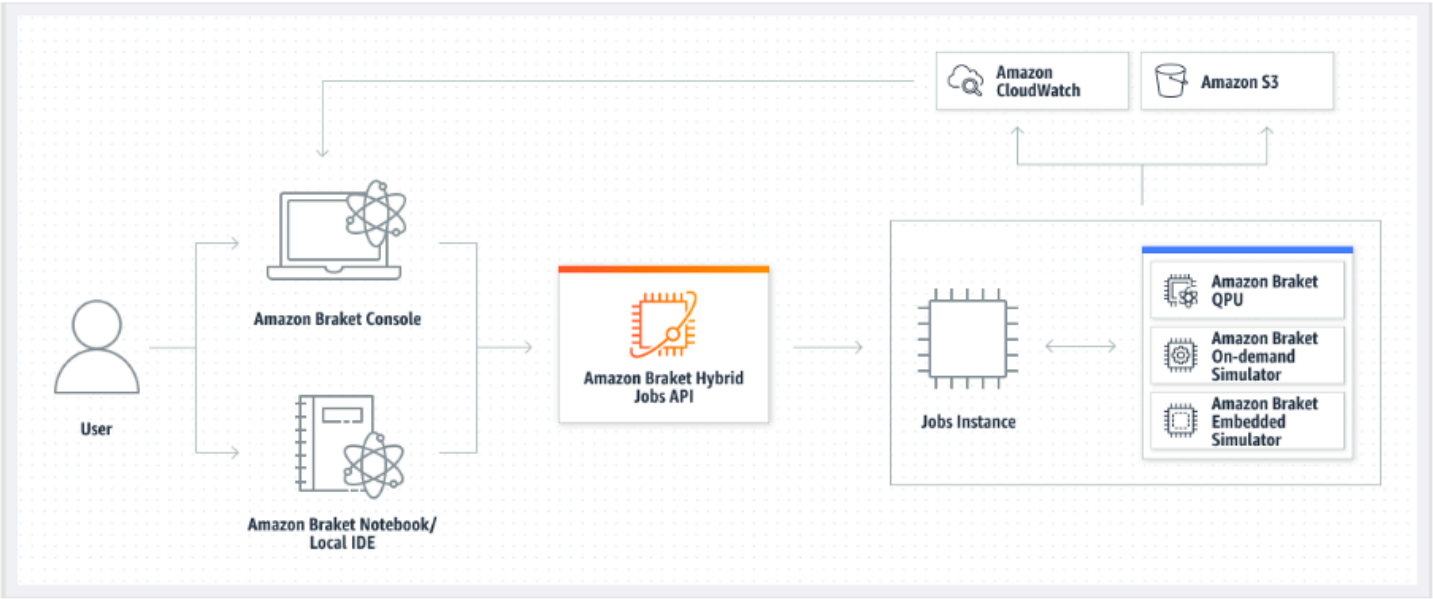
Per ulteriori esempi, casi d'uso e best practice, consulta gli esempi di Amazon GitHub Braket.

Esecuzione di un processo ibrido con Amazon Braket Hybrid Jobs

Per eseguire un processo ibrido con Amazon Braket Hybrid Jobs, devi prima definire il tuo algoritmo. Puoi definirlo scrivendo lo script dell'algoritmo e, facoltativamente, altri file di dipendenza utilizzando Amazon Braket Python o. SDK PennyLane Se desideri utilizzare altre librerie (open source o proprietarie), puoi definire un'immagine del contenitore personalizzata utilizzando Docker, che include queste librerie. Per ulteriori informazioni, consulta Bring your own container () BYOC.

In entrambi i casi, successivamente crei un lavoro ibrido utilizzando Amazon Braket API, se fornisci lo script o il contenitore dell'algoritmo, seleziona il dispositivo quantistico di destinazione che il lavoro ibrido deve utilizzare, quindi scegli tra una serie di impostazioni opzionali. I valori predefiniti

forniti per queste impostazioni opzionali funzionano per la maggior parte dei casi d'uso. Affinché il dispositivo di destinazione esegua il tuo Hybrid JobQPU, puoi scegliere tra un simulatore on-demand (ad esempio SV1, DM1 oppure TN1) o la classica istanza di job ibrida stessa. Con un simulatore on-demand oppureQPU, il tuo contenitore di lavori ibrido effettua API chiamate verso un dispositivo remoto. Con i simulatori incorporati, il simulatore è incorporato nello stesso contenitore dello script dell'algoritmo. I simulatori di fulmini di PennyLane sono integrati nel contenitore di job ibrido predefinito e preconfigurato che puoi utilizzare. Se esegui il codice utilizzando un PennyLane simulatore incorporato o un simulatore personalizzato, puoi specificare un tipo di istanza e quante istanze desideri utilizzare. Consulta la pagina dei prezzi di Amazon Braket per i costi associati a ciascuna scelta.



Se il dispositivo di destinazione è un simulatore on-demand o integrato, Amazon Braket inizia subito a eseguire il processo ibrido. Attiva l'istanza del processo ibrido (puoi personalizzare il tipo di istanza nel API call), esegue l'algoritmo, scrive i risultati su Amazon S3 e rilascia le tue risorse. Questa versione di risorse garantisce che paghi solo per ciò che usi.

Il numero totale di lavori ibridi simultanei per unità di elaborazione quantistica (QPU) è limitato. Oggi, è possibile eseguire solo un processo ibrido QPU alla volta. Le code vengono utilizzate per controllare il numero di processi ibridi consentiti per l'esecuzione in modo da non superare il limite consentito. Se il dispositivo di destinazione è unQPU, il lavoro ibrido entra innanzitutto nella coda dei lavori del selezionato. QPU Amazon Braket attiva l'istanza di lavoro ibrida necessaria ed esegue il processo ibrido sul dispositivo. Per tutta la durata dell'algoritmo, il processo ibrido ha accesso prioritario, il che significa che le attività quantistiche del lavoro ibrido vengono eseguite prima delle altre attività quantistiche di Braket in coda sul dispositivo, a condizione che le attività quantistiche

del lavoro vengano inviate una volta ogni pochi minuti. QPU Una volta completato il lavoro ibrido, vengono rilasciate risorse, il che significa che paghi solo per ciò che utilizzi.

> **Note**
>
> I dispositivi sono regionali e il processo ibrido viene eseguito nello stesso Regione AWS come dispositivo principale.

Sia nel simulatore che nello scenario QPU target, hai la possibilità di definire metriche personalizzate dell'algoritmo, come l'energia del tuo algoritmo hamiltoniano, come parte dell'algoritmo. Queste metriche vengono segnalate automaticamente ad Amazon CloudWatch e da lì vengono visualizzate quasi in tempo reale nella console Amazon Braket.

> **Note**
>
> Se desideri utilizzare un'istanza GPU basata, assicurati di utilizzare uno dei simulatori GPU basati disponibili con i simulatori incorporati su Braket (ad esempio,). `lightning.gpu` Se scegli uno dei simulatori incorporati CPU basati su di essi (ad esempio, `obraket:default-simulator`)`lightning.qubit`, non GPU verrà utilizzato e potresti incorrere in costi inutili.

Crea il tuo primo Hybrid Job

Questa sezione mostra come creare un Hybrid Job usando uno script Python. In alternativa, per creare un lavoro ibrido da codice Python locale, come il tuo ambiente di sviluppo integrato preferito (IDE) o un notebook Braket, vedi. Esecuzione del codice locale come processo ibrido

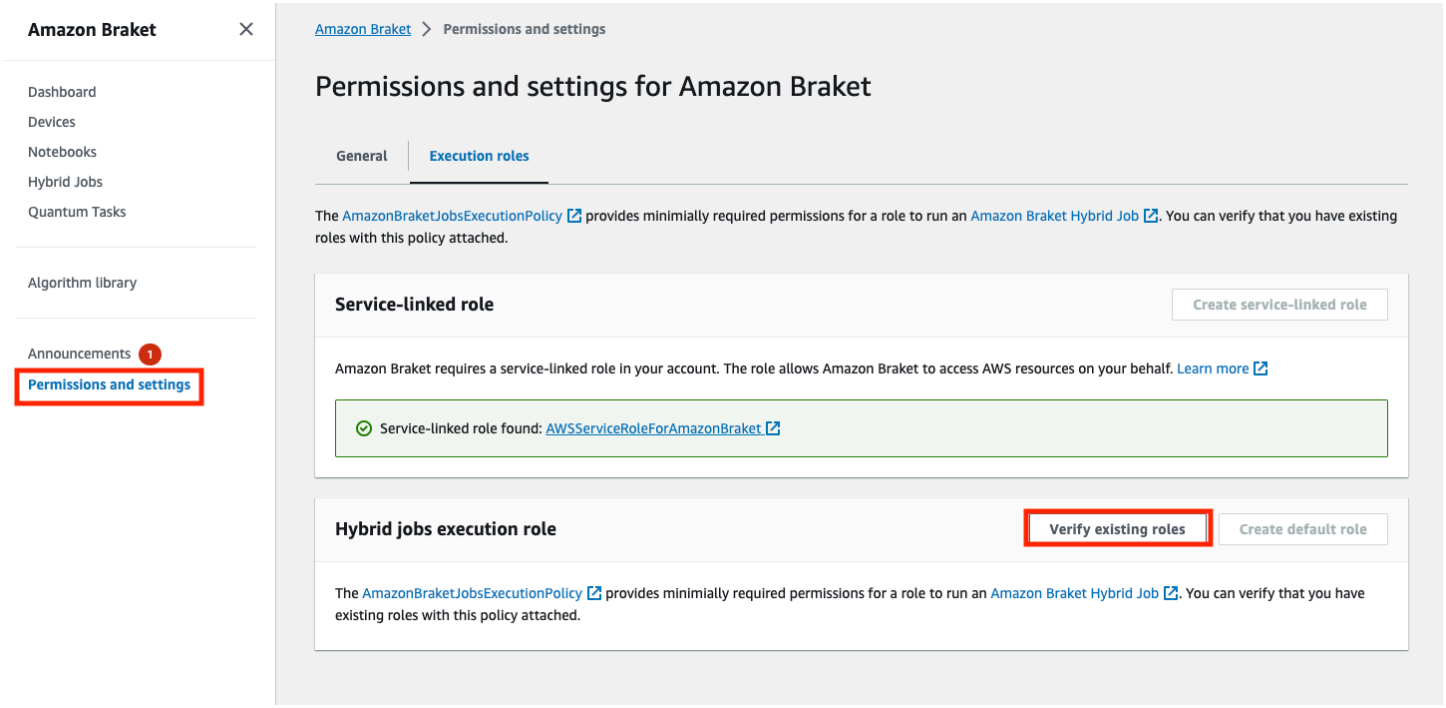
In questa sezione:

- Imposta le autorizzazioni
- Crea ed esegui
- Monitora i risultati

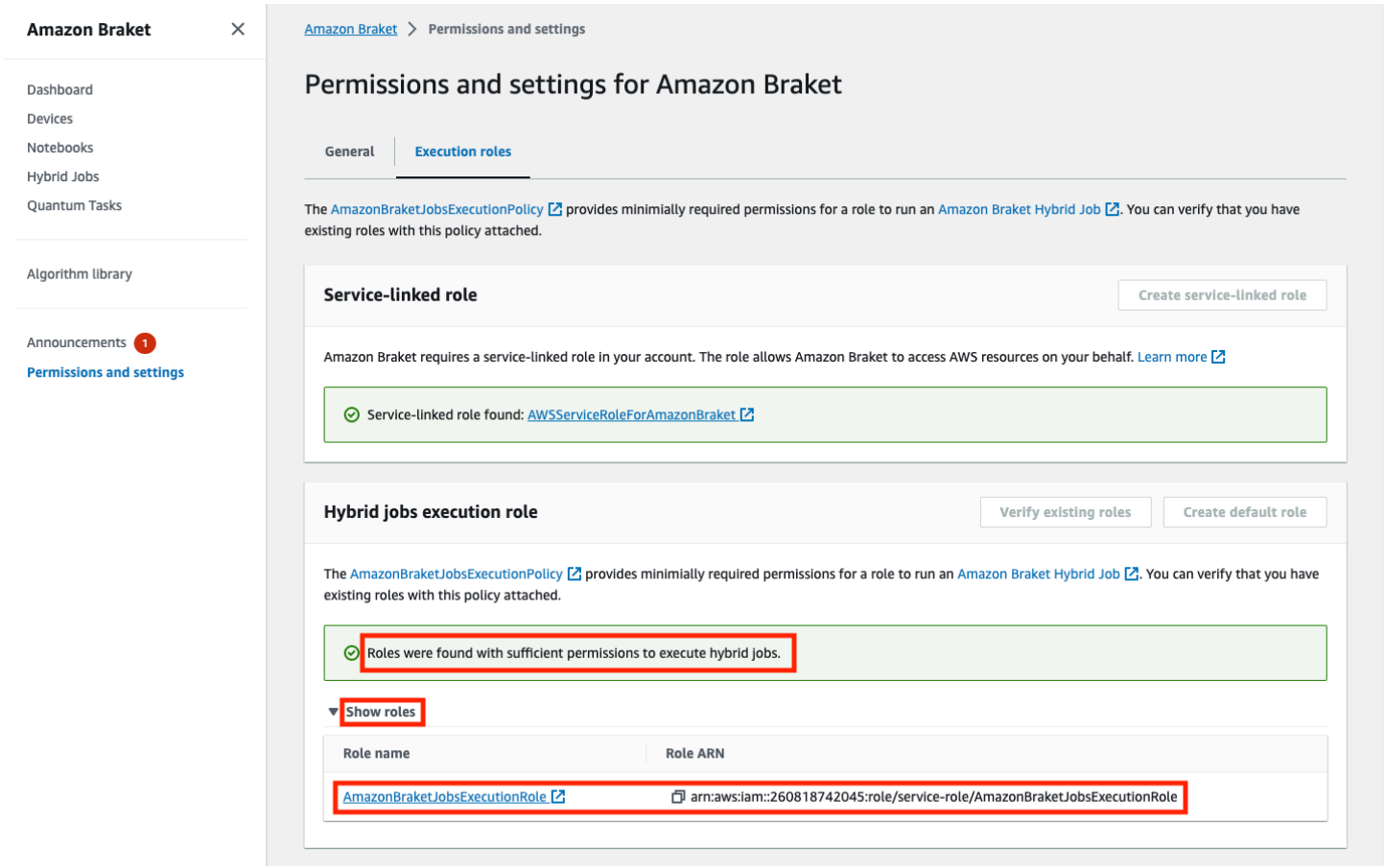
Imposta le autorizzazioni

Prima di eseguire il primo lavoro ibrido, è necessario assicurarsi di disporre delle autorizzazioni sufficienti per procedere con questa attività. Per stabilire di disporre delle autorizzazioni corrette,

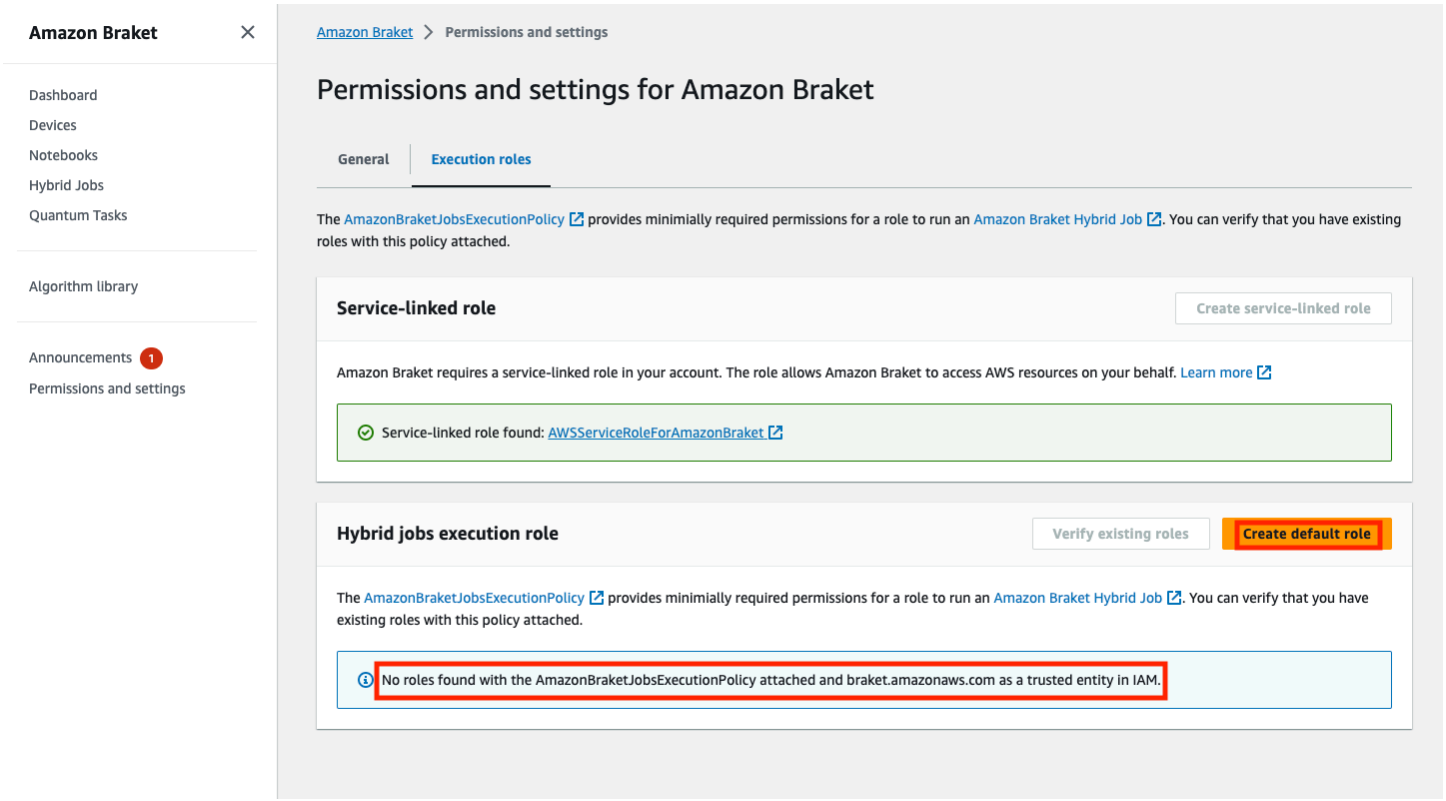
seleziona Autorizzazioni dal menu sul lato sinistro della Braket Console. La pagina Gestione delle autorizzazioni per Amazon Braket ti aiuta a verificare se uno dei tuoi ruoli esistenti dispone di autorizzazioni sufficienti per eseguire il tuo lavoro ibrido o ti guida attraverso la creazione di un ruolo predefinito che può essere utilizzato per eseguire il tuo lavoro ibrido se non disponi già di tale ruolo.



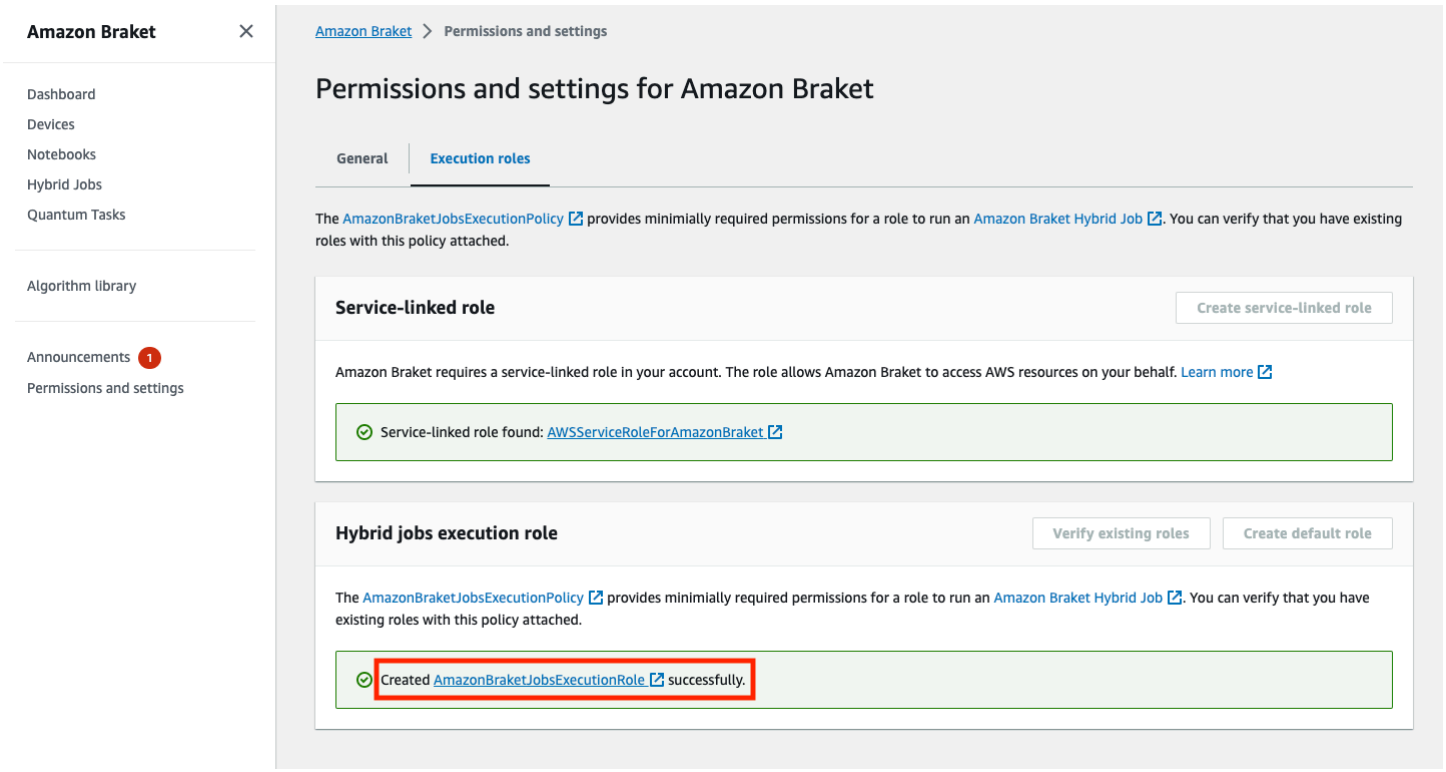
Per verificare di disporre di ruoli con autorizzazioni sufficienti per eseguire un lavoro ibrido, seleziona il pulsante Verifica il ruolo esistente. Se lo fai, ricevi un messaggio che indica che i ruoli sono stati trovati. Per visualizzare i nomi dei ruoli e il relativo ruoloARNs, seleziona il pulsante Mostra ruoli.



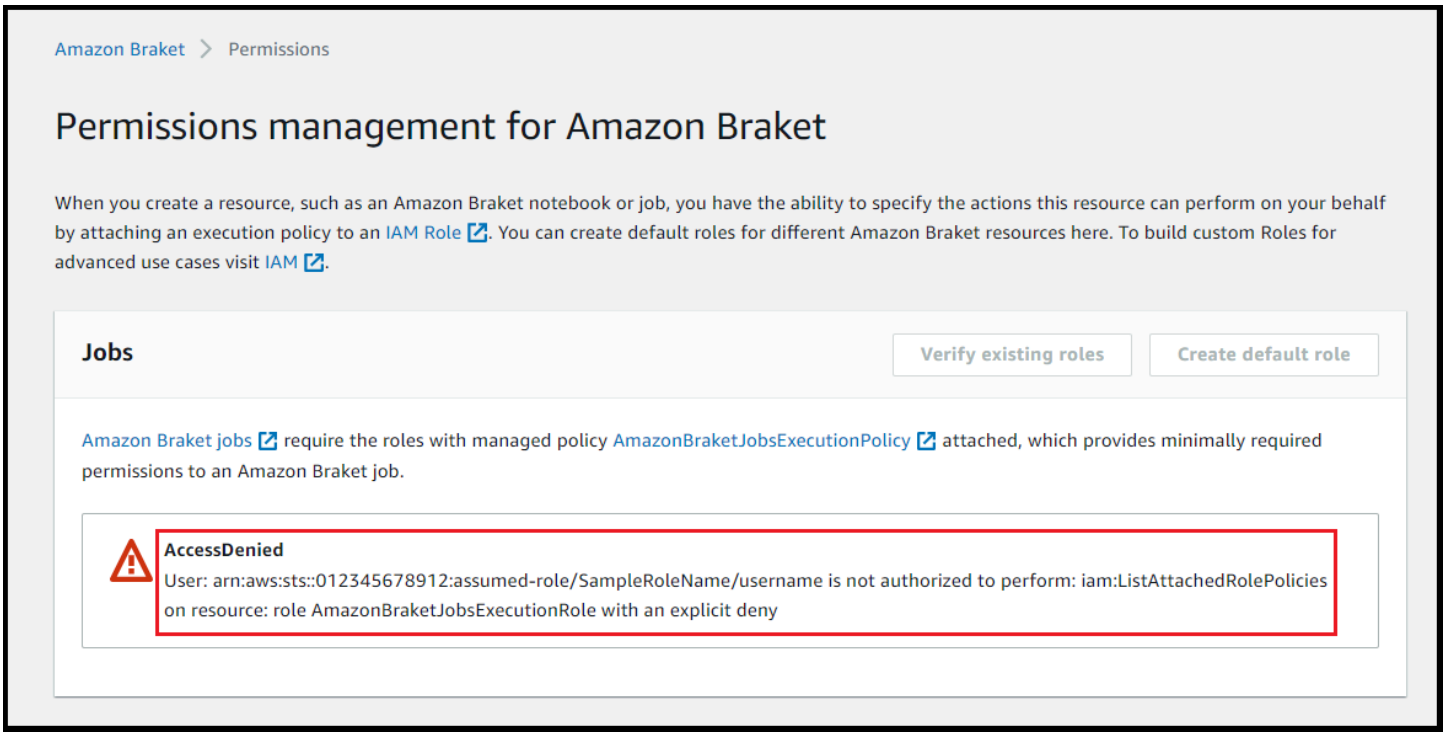
Se non disponi di un ruolo con autorizzazioni sufficienti per eseguire un lavoro ibrido, ricevi un messaggio che indica che tale ruolo non è stato trovato. Seleziona il pulsante Crea ruolo predefinito per ottenere un ruolo con autorizzazioni sufficienti.



Se il ruolo è stato creato correttamente, riceverai un messaggio di conferma.

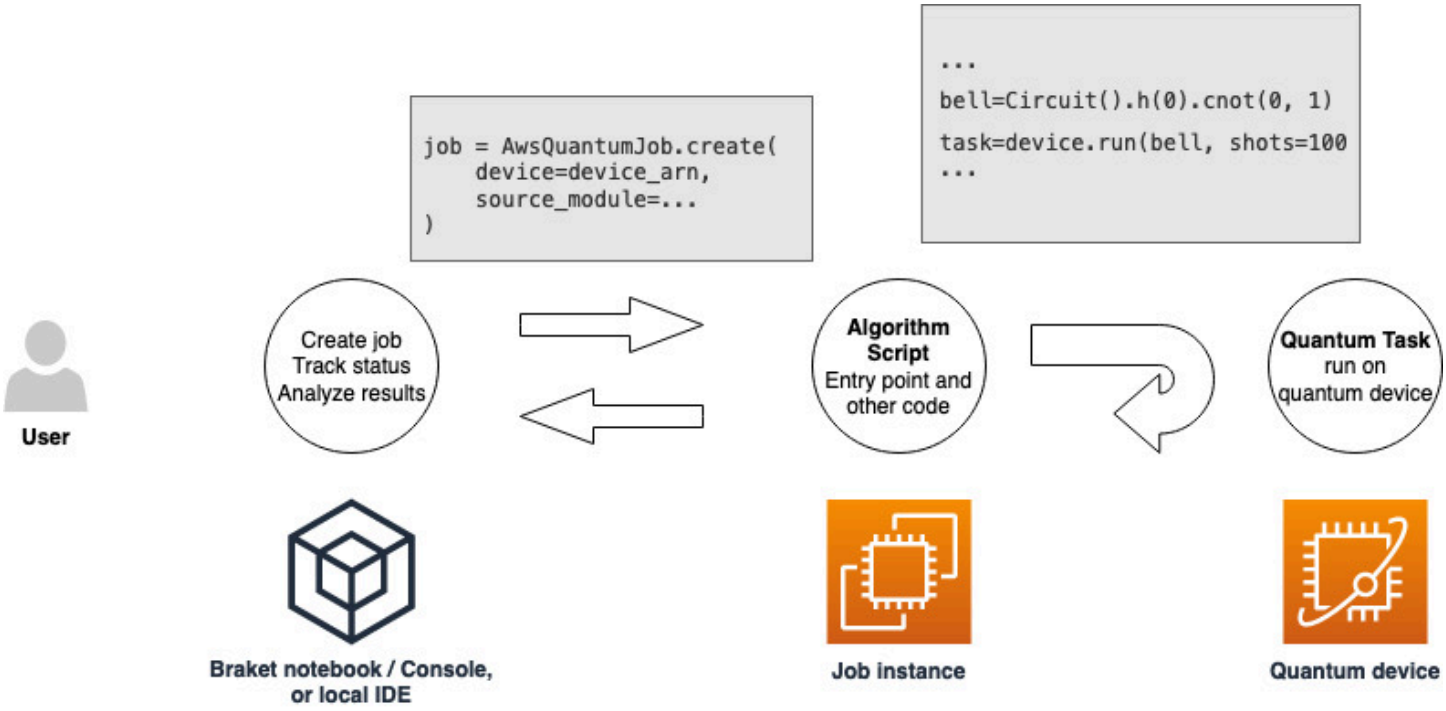


Se non disponi delle autorizzazioni necessarie per effettuare questa richiesta, ti verrà negato l'accesso. In questo caso, contatta il tuo interno AWS amministratore.



Crea ed esegui

Una volta ottenuto un ruolo con le autorizzazioni per eseguire un lavoro ibrido, sei pronto per procedere. L'elemento chiave del tuo primo lavoro ibrido con Braket è lo script dell'algoritmo. Definisce l'algoritmo da eseguire e contiene le classiche attività logiche e quantistiche che fanno parte dell'algoritmo. Oltre allo script dell'algoritmo, puoi fornire altri file di dipendenza. Lo script dell'algoritmo, insieme alle sue dipendenze, viene chiamato modulo sorgente. Il punto di ingresso definisce il primo file o funzione da eseguire nel modulo di origine all'avvio del processo ibrido.



Innanzitutto, consideriamo il seguente esempio di base di uno script di algoritmo che crea cinque stati a campana e stampa i risultati di misurazione corrispondenti.

```
import os

from braket.aws import AwsDevice
from braket.circuits import Circuit

def start_here():

    print("Test job started!")

    # Use the device declared in the job script
    device = AwsDevice(os.environ["AMZN_BRAKET_DEVICE_ARN"])

    bell = Circuit().h(0).cnot(0, 1)
    for count in range(5):
        task = device.run(bell, shots=100)
        print(task.result().measurement_counts)

    print("Test job completed!")
```

Salvate questo file con il nome algorithm_script.py nella directory di lavoro corrente sul notebook Braket o nell'ambiente locale. Il file algorithm_script.py ha `start_here()` come punto di ingresso pianificato.

Quindi, crea un file Python o un taccuino Python nella stessa directory del file algorithm_script.py. Questo script avvia il processo ibrido e gestisce qualsiasi elaborazione asincrona, come la stampa dello stato o dei risultati chiave che ci interessano. Come minimo, questo script deve specificare lo script di lavoro ibrido e il dispositivo principale.

> **Note**
>
> Per ulteriori informazioni su come creare un notebook Braket o caricare un file, ad esempio il file algorithm_script.py, nella stessa directory dei notebook, consulta Esegui il tuo primo circuito usando Amazon Braket Python SDK

Per questo primo caso di base, scegli come obiettivo un simulatore. Indipendentemente dal tipo di dispositivo quantistico scelto come target, un simulatore o un'unità di elaborazione quantistica effettiva (QPU), il dispositivo specificato `device` nello script seguente viene utilizzato per pianificare il processo ibrido ed è disponibile per gli script dell'algoritmo come variabile di ambiente. `AMZN_BRAKET_DEVICE_ARN`

> **Note**
>
> È possibile utilizzare solo dispositivi disponibili in Regione AWS del tuo lavoro ibrido. L'Amazon Braket SDK auto lo seleziona Regione AWS. Ad esempio, un lavoro ibrido in us-east-1 può utilizzare IonQ, SV1, DM1e TN1 dispositivi, ma non Rigetti dispositivi.

Se scegli un computer quantistico anziché un simulatore, Braket pianifica i tuoi lavori ibridi per eseguire tutte le attività quantistiche con accesso prioritario.

```
from braket.aws import AwsQuantumJob
from braket.devices import Devices

job = AwsQuantumJob.create(
    Devices.Amazon.SV1,
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
```

```
    wait_until_complete=True
)
```

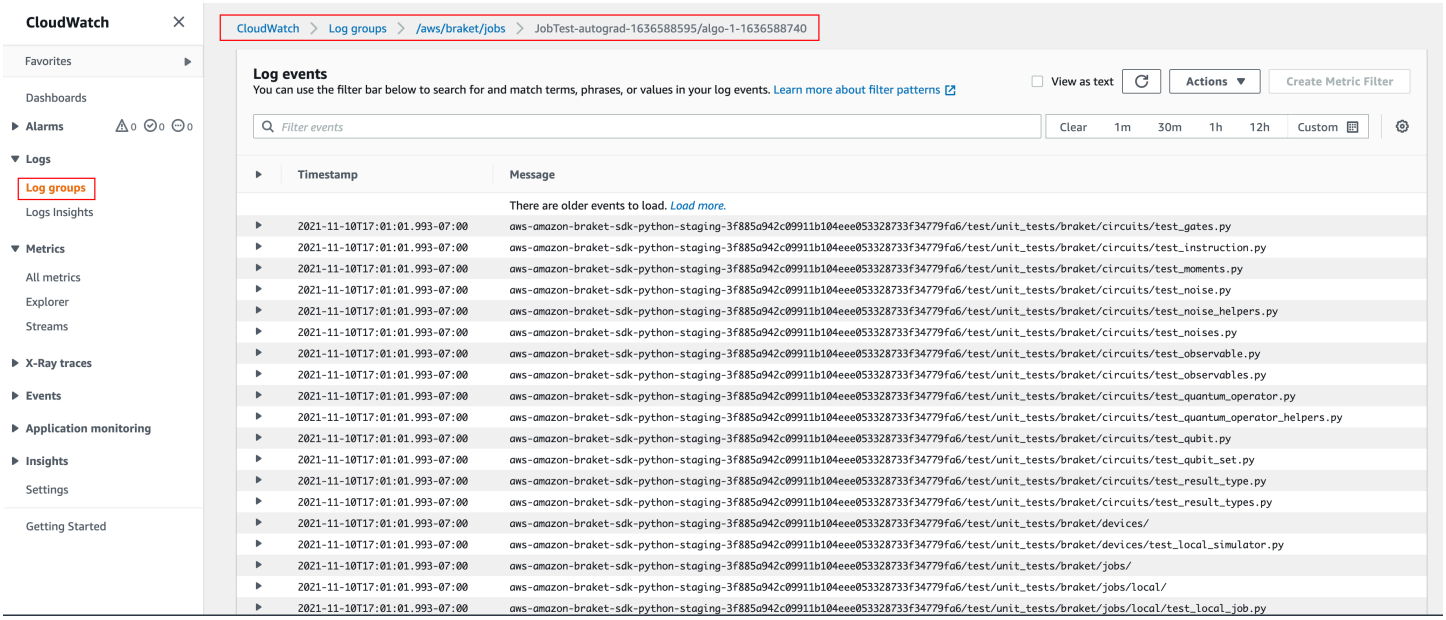
Il parametro `wait_until_complete=True` imposta una modalità dettagliata in modo che il lavoro stampi l'output del lavoro effettivo mentre è in esecuzione. Dovreste vedere un output simile a quello dell'esempio seguente.

```
job = AwsQuantumJob.create(
    Devices.Amazon.SV1,
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    wait_until_complete=True,
)
Initializing Braket Job: arn:aws:braket:us-west-2:<accountid>:job/<UUID>
.....................................
.
.
.

Completed 36.1 KiB/36.1 KiB (692.1 KiB/s) with 1 file(s) remaining#015download:
 s3://braket-external-assets-preview-us-west-2/HybridJobsAccess/models/
braket-2019-09-01.normal.json to ../../braket/additional_lib/original/
braket-2019-09-01.normal.json
Running Code As Process
Test job started!!!!!
Counter({'00': 55, '11': 45})
Counter({'11': 59, '00': 41})
Counter({'00': 55, '11': 45})
Counter({'00': 58, '11': 42})
Counter({'00': 55, '11': 45})
Test job completed!!!!!
Code Run Finished
2021-09-17 21:48:05,544 sagemaker-training-toolkit INFO     Reporting training SUCCESS
```

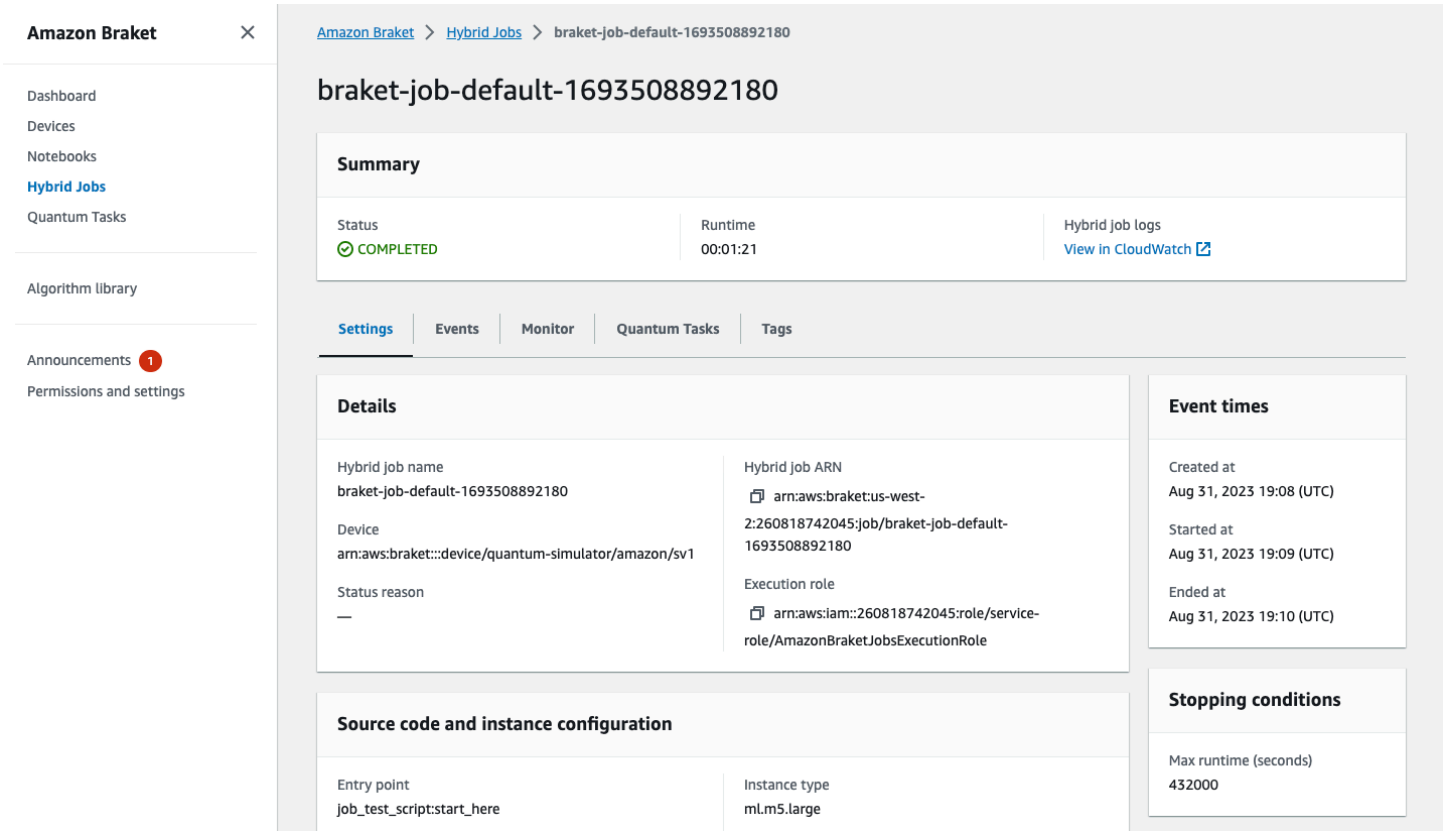
> **Note**
>
> Puoi anche usare il tuo modulo personalizzato con il metodo AwsQuantumJob.create passandone la posizione (il percorso di una directory o di un file locale o un file S3 URI di un file tar.gz). Per un esempio funzionante, consulta il file Parallelize_training_for_ QML .ipynb nella cartella hybrid jobs nel repository Amazon Braket examples Github.

Monitora i risultati

In alternativa, puoi accedere all'output del registro da Amazon CloudWatch. Per fare ciò, vai alla scheda Gruppi di log nel menu a sinistra della pagina dei dettagli del lavoro, seleziona il gruppo di log`aws/braket/jobs`, quindi scegli il flusso di log che contiene il nome del lavoro. Nell'esempio precedente è `braket-job-default-1631915042705/algo-1-1631915190`.



Puoi anche visualizzare lo stato del lavoro ibrido nella console selezionando la pagina Hybrid Jobs e quindi scegliendo Impostazioni.



Il tuo lavoro ibrido produce alcuni artefatti in Amazon S3 mentre è in esecuzione. Il nome predefinito del bucket S3 è `amazon-braket-<region>-<accountid>` e il contenuto si trova nella directory. `jobs/<jobname>/<timestamp>` Puoi configurare le posizioni S3 in cui vengono archiviati questi artefatti specificandone una diversa `code_location` quando il lavoro ibrido viene creato con Braket Python. SDK

> **Note**
>
> Questo bucket S3 deve trovarsi nello stesso Regione AWS come copione del tuo lavoro.

La `jobs/<jobname>/<timestamp>` directory contiene una sottocartella con l'output dello script del punto di ingresso in un `model.tar.gz` file. Esiste anche una directory denominata `script` che contiene gli elementi dello script dell'algoritmo in un file. `source.tar.gz` I risultati delle vostre attività quantistiche effettive si trovano nella directory denominata. `jobs/<jobname>/tasks`

Salvare i risultati del lavoro

È possibile salvare i risultati generati dallo script dell'algoritmo in modo che siano disponibili dall'oggetto di lavoro ibrido nello script del processo ibrido e dalla cartella di output in Amazon S3 (in un file tar-zip denominato model.tar.gz).

L'output deve essere salvato in un file utilizzando un formato JavaScript Object Notation (). JSON Se i dati non possono essere serializzati facilmente in testo, come nel caso di un array numpy, puoi passare un'opzione per serializzare utilizzando un formato di dati selezionato. Vedi il modulo braket.jobs.data_persistence per maggiori dettagli.

Per salvare i risultati dei lavori ibridi, aggiungi le seguenti righe commentate con # allo script dell'algoritmo. ADD

```
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.jobs import save_job_result #ADD

def start_here():

    print("Test job started!!!!!")

    device = AwsDevice(os.environ['AMZN_BRAKET_DEVICE_ARN'])

    results = []  #ADD

    bell = Circuit().h(0).cnot(0, 1)
    for count in range(5):
        task = device.run(bell, shots=100)
        print(task.result().measurement_counts)
        results.append(task.result().measurement_counts)  #ADD

        save_job_result({ "measurement_counts": results })  #ADD

    print("Test job completed!!!!!")
```

È quindi possibile visualizzare i risultati del lavoro dallo script di lavoro aggiungendo la riga **`print(job.result())`**commentata con #. ADD

```
import time
from braket.aws import AwsQuantumJob
```

```
job = AwsQuantumJob.create(
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    device_arn="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
)

print(job.arn)
while job.state() not in AwsQuantumJob.TERMINAL_STATES:
    print(job.state())
    time.sleep(10)

print(job.state())
print(job.result())   #ADD
```

In questo esempio, abbiamo rimosso `wait_until_complete=True` per sopprimere l'output verboso. Puoi aggiungerlo nuovamente per il debug. Quando si esegue questo processo ibrido, vengono emessi l'identificatore e il`job-arn`, seguiti dallo stato del lavoro ibrido ogni 10 secondi fino all'arrivo del lavoro ibrido`COMPLETED`, dopodiché vengono visualizzati i risultati del circuito a campana. Guarda l'esempio seguente.

```
arn:aws:braket:us-west-2:111122223333:job/braket-job-default-1234567890123
INITIALIZED
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
...
RUNNING
RUNNING
COMPLETED
{'measurement_counts': [{'11': 53, '00': 47},..., {'00': 51, '11': 49}]}
```

Salvare e riavviare i lavori ibridi utilizzando i checkpoint

È possibile salvare iterazioni intermedie dei job ibridi utilizzando i checkpoint. Nell'esempio di script di algoritmo della sezione precedente, dovresti aggiungere le seguenti righe commentate con # ADD per creare file di checkpoint.

```
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.jobs import save_job_checkpoint  #ADD
import os

def start_here():

    print("Test job starts!!!!!")

    device = AwsDevice(os.environ["AMZN_BRAKET_DEVICE_ARN"])

    #ADD the following code
    job_name = os.environ["AMZN_BRAKET_JOB_NAME"]
     save_job_checkpoint(
     checkpoint_data={"data": f"data for checkpoint from {job_name}"},
     checkpoint_file_suffix="checkpoint-1",
     ) #End of ADD

    bell = Circuit().h(0).cnot(0, 1)
    for count in range(5):
        task = device.run(bell, shots=100)
        print(task.result().measurement_counts)

    print("Test hybrid job completed!!!!!")
```

Quando si esegue il job ibrido, viene creato il file -checkpoint-1.json <jobname>negli artefatti del job ibrido nella directory checkpoints con un percorso predefinito. `/opt/jobs/checkpoints` Lo script di lavoro ibrido rimane invariato a meno che non si desideri modificare questo percorso predefinito.

Se si desidera caricare un lavoro ibrido da un checkpoint generato da un precedente lavoro ibrido, lo script dell'algoritmo utilizza. `from braket.jobs import load_job_checkpoint` La logica da caricare nello script dell'algoritmo è la seguente.

```
checkpoint_1 = load_job_checkpoint(
    "previous_job_name",
    checkpoint_file_suffix="checkpoint-1",
```

```
)
```

Dopo aver caricato questo checkpoint, puoi continuare la logica in base al contenuto caricato su. `checkpoint-1`

> **Note**
>
> Il checkpoint_file_suffix deve corrispondere al suffisso precedentemente specificato durante la creazione del checkpoint.

Lo script di orchestrazione deve specificare il precedente lavoro ibrido con la riga commentata con `job-arn` #. ADD

```
job = AwsQuantumJob.create(
    source_module="source_dir",
    entry_point="source_dir.algorithm_script:start_here",
    device_arn="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
    copy_checkpoints_from_job="<previous-job-ARN>", #ADD
    )
```

Creazione e debug di un lavoro ibrido con modalità locale

Se state creando un nuovo algoritmo ibrido, la modalità locale vi aiuta a eseguire il debug e testare lo script dell'algoritmo. La modalità locale è una funzionalità che ti consente di eseguire il codice che intendi utilizzare in Amazon Braket Hybrid Jobs, ma senza bisogno di Braket per gestire l'infrastruttura per l'esecuzione del lavoro ibrido. Invece, esegui lavori ibridi localmente sulla tua istanza di Braket Notebook o su un client preferito come un laptop o un computer desktop. In modalità locale, è comunque possibile inviare attività quantistiche a dispositivi reali, ma non si ottengono i vantaggi in termini di prestazioni se si esegue su un dispositivo effettivo in QPU modalità locale.

Per utilizzare la modalità locale, esegui la modifica `AwsQuantumJob` `LocalQuantumJob` ovunque si verifichi. Ad esempio, per eseguire l'esempio di Create your first hybrid job, modificate lo script del job ibrido come segue.

```
from braket.jobs.local import LocalQuantumJob

job = LocalQuantumJob.create(
```

```
    device="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
)
```

> **Note**
>
> Docker, che è già preinstallato nei notebook Amazon Braket, deve essere installato nel tuo ambiente locale per utilizzare questa funzionalità. Le istruzioni per l'installazione di Docker sono disponibili qui. Inoltre, non tutti i parametri sono supportati in modalità locale.

Esegui le tue attività quantistiche con Amazon Braket

Braket fornisce un accesso sicuro e su richiesta a diversi tipi di computer quantistici. Hai accesso ai computer quantistici basati su gate da IonQ, IQMe Rigetti, oltre a un simulatore hamiltoniano analogico di. QuEra Inoltre, non avete alcun impegno anticipato e non dovete procurarvi l'accesso tramite singoli provider.

- La console Amazon Braket fornisce informazioni e stato del dispositivo per aiutarti a creare, gestire e monitorare le tue risorse e le tue attività quantistiche.
- Invia ed esegui attività quantistiche tramite Amazon Braket SDK Python e tramite la console. SDKÈ accessibile tramite preconfigurato Amazon Notebook Braket.
- Amazon Braket API è accessibile tramite Amazon Braket SDK Python e taccuini. È possibile effettuare chiamate direttamente al API se stai creando applicazioni che funzionano con l'informatica quantistica a livello di codice.

Gli esempi presenti in questa sezione mostrano come lavorare con Amazon Braket. API utilizzando direttamente Amazon Braket SDK Python insieme a AWS Python SDK per Braket (Boto3).

Maggiori informazioni su Amazon Staffa Python SDK

Per lavorare con Amazon Braket PythonSDK, installa prima AWS Python SDK for Braket (Boto3) in modo da poter comunicare con AWS API. Puoi pensare ad Amazon Braket Python SDK come a un pratico involucro di Boto3 per i clienti quantistici.

- Boto3 contiene interfacce a cui devi attingere AWS API. (Nota che Boto3 è un Python SDK di grandi dimensioni che parla con AWS API. La maggior parte Servizi AWS supporta un'interfaccia Boto3.)
- Amazon Braket Python SDK contiene moduli software per circuiti, porte, dispositivi, tipi di risultati e altre parti di un'attività quantistica. Ogni volta che crei un programma, importi i moduli necessari per quell'attività quantistica.
- Amazon Braket Python SDK è accessibile tramite notebook, precaricati con tutti i moduli e le dipendenze necessari per eseguire attività quantistiche.
- Puoi importare moduli da Amazon Braket Python in SDK qualsiasi script Python se non desideri lavorare con i notebook.

Dopo aver installato Boto3, una panoramica dei passaggi per la creazione di un'attività quantistica tramite il Amazon Braket SDK Python è simile al seguente:

1. (Facoltativamente) Apri il notebook.
2. Importate i SDK moduli necessari per i vostri circuiti.
3. Specificate un simulatore QPU o.
4. Crea un'istanza del circuito.
5. Avvia il circuito.
6. Raccogli i risultati.

Gli esempi in questa sezione mostrano i dettagli di ogni passaggio.

Per altri esempi, consulta il repository Amazon Braket Examples su. GitHub

In questa sezione:

- Invio di attività quantistiche a QPUs
- Quando verrà eseguita la mia attività quantistica?
- Gestione del tuo lavoro in Amazon Braket Hybrid
- Lavorare con le prenotazioni
- Tecniche di mitigazione degli errori

Invio di attività quantistiche a QPUs

Amazon Braket fornisce l'accesso a diversi dispositivi in grado di eseguire attività quantistiche. Puoi inviare attività quantistiche singolarmente oppure configurare attività quantistiche in batch.

QPUs

Puoi inviare attività quantistiche a QPUs in qualsiasi momento, ma l'attività viene eseguita entro determinate finestre di disponibilità visualizzate nella pagina Dispositivi della console Amazon Braket. Puoi recuperare i risultati dell'attività quantistica con il quantum task ID, introdotto nella sezione successiva.

- IonQ Aria 1 : `arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1`
- IonQ Aria 2 : `arn:aws:braket:us-east-1::device/qpu/ionq/Aria-2`

- IonQ Forte 1 (solo su prenotazione): `arn:aws:braket:us-east-1::device/qpu/ionq/Forte-1`
- IQM Garnet : `arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet`
- QuEra Aquila : `arn:aws:braket:us-east-1::device/qpu/quera/Aquila`
- Rigetti Aspen-M-3 : `arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3`
- Rigetti Ankaa-2 : `arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-2`

> **Note**
>
> È possibile annullare le attività quantistiche `CREATED` nello stato per i simulatori QPUs e su richiesta. È possibile annullare le attività quantistiche `QUEUED` nello stato nel miglior modo possibile per i simulatori su richiesta e. QPUs Tieni presente che è improbabile che le attività QPU `QUEUED` quantistiche vengano annullate correttamente durante le finestre di disponibilità. QPU

In questa sezione:

IonQ

IonQ offre porte QPUs basate sulla tecnologia delle trappole ioniche. IonQ's QPUsgli ioni intrappolati sono costruiti su una catena di ioni 171Yb+ intrappolati che sono confinati spazialmente mediante una trappola per elettrodi superficiali microfabbricata all'interno di una camera a vuoto.

IonQ i dispositivi supportano le seguenti porte quantistiche.

```
'x', 'y', 'z', 'rx', 'ry', 'rz', 'h', 'cnot', 's', 'si', 't', 'ti', 'v', 'vi', 'xx',
 'yy', 'zz', 'swap'
```

Con la compilazione letterale, il IonQ QPUssupporta le seguenti porte native.

```
'gpi', 'gpi2', 'ms'
```

Se si specificano solo parametri a due fasi quando si utilizza la porta MS nativa, viene eseguita una porta MS completamente interconnessa. Un gate MS completamente aggrovigliato esegue sempre una rotazione π /2. Per specificare un angolo diverso e far funzionare una porta MS parzialmente aggrovigliata, si specifica l'angolo desiderato aggiungendo un terzo parametro. Per ulteriori informazioni, vedete il modulo braket.circuits.gate.

Queste porte native possono essere utilizzate solo con la compilazione letterale. Per ulteriori informazioni sulla compilazione letterale, consulta Verbatim Compilation.

IQM

IQM i processori quantistici sono dispositivi universali e modello gate basati su qubit transmonici superconduttori. Il IQM Garnet device è un dispositivo da 20 qubit con topologia a reticolo quadrato.

Il IQM i dispositivi supportano le seguenti porte quantistiche.

```
"ccnot", "cnot", "cphaseshift", "cphaseshift00", "cphaseshift01", "cphaseshift10",
 "cswap", "swap", "iswap", "pswap", "ecr", "cy", "cz", "xy", "xx", "yy", "zz", "h",
 "i", "phaseshift", "rx", "ry", "rz", "s", "si", "t", "ti", "v", "vi", "x", "y", "z"
```

Con la compilazione letterale, il IQM i dispositivi supportano le seguenti porte native.

```
'cz', 'prx'
```

Rigetti

Rigetti i processori quantistici sono macchine universali modello gate basate su superconduttori completamente sintonizzabili qubits.

- Il Aspen-M-3 system è un dispositivo da 79 qubit che sfrutta la sua tecnologia proprietaria multi-chip ed è assemblato da due processori da 40 qubit.
- Il Ankaa-2 il sistema è un dispositivo da 84 qubit che utilizza una tecnologia scalabile multi-chip.

Il Rigetti i dispositivi supportano le seguenti porte quantistiche.

```
'cz', 'xy', 'ccnot', 'cnot', 'cphaseshift', 'cphaseshift00', 'cphaseshift01',
 'cphaseshift10', 'cswap', 'h', 'i', 'iswap', 'phaseshift', 'pswap', 'rx', 'ry', 'rz',
 's', 'si', 'swap', 't', 'ti', 'x', 'y', 'z'
```

Con la compilazione letterale, Aspen-M-3 supporta le seguenti porte native.

```
'rx', 'rz', 'cz', 'cphaseshift', 'xy'
```

Ankaa-2 supporta le seguenti porte native.

```
'rx', 'rz', 'cz', 'iswap'
```

Rigetti i processori quantistici superconduttori possono far funzionare la porta 'rx' solo con gli angoli di ±π o ±π.

Il controllo a livello di impulsi è disponibile sul Rigetti dispositivi, che supportano una serie di frame predefiniti dei seguenti tipi per Aspen-M-3 sistema.

```
'rf', 'rf_f12', 'ro_rx', 'ro_rx', 'cz', 'cphase', 'xy'
```

Il Ankaa-2 il sistema supporta i seguenti tipi di frame predefiniti.

```
`flux_tx`, `charge_tx`, `readout_rx`, `readout_tx`
```

Per ulteriori informazioni su questi frame, vedete Ruoli dei frame e delle porte.

QuEra

QuEra offre dispositivi basati su atomi neutri in grado di eseguire attività quantistiche di Analog Hamiltonian Simulation (). AHS Questi dispositivi speciali riproducono fedelmente la dinamica quantistica dipendente dal tempo di centinaia di qubit che interagiscono simultaneamente.

È possibile programmare questi dispositivi secondo il paradigma della simulazione hamiltoniana analogica prescrivendo il layout del registro dei qubit e la dipendenza temporale e spaziale dei campi di manipolazione. Amazon Braket fornisce utilità per creare tali programmi tramite il AHS modulo python,. SDK `braket.ahs`

Per ulteriori informazioni, consulta i taccuini di esempio di Analog Hamiltonian Simulation o la pagina Invio di un programma analogico tramite Aquila. QuEra

Esempio: invio di un compito quantistico a un QPU

Amazon Braket ti consente di eseguire un circuito quantistico su un dispositivo. QPU L'esempio seguente mostra come inviare un'attività quantistica a Rigetti oppure IonQ dispositivi.

Scegli il Rigetti Aspen-M-3 dispositivo, quindi guarda il grafico di connettività associato

```
# import the QPU module
from braket.aws import AwsDevice
# choose the Rigetti device
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3")

# take a look at the device connectivity graph
device.properties.dict()['paradigm']['connectivity']
```

```
{'fullyConnected': False,
 'connectivityGraph': {'0': ['1', '7'],
  '1': ['0', '16'],
  '2': ['3', '15'],
  '3': ['2', '4'],
  '4': ['3', '5'],
  '5': ['4', '6'],
  '6': ['5', '7'],
  '7': ['0', '6'],
  '11': ['12', '26'],
  '12': ['13', '11'],
  '13': ['12', '14'],
  '14': ['13', '15'],
  '15': ['2', '14', '16'],
  '16': ['1', '15', '17'],
  '17': ['16'],
  '20': ['21', '27'],
  '21': ['20', '36'],
  '22': ['23', '35'],
  '23': ['22', '24'],
  '24': ['23', '25'],
  '25': ['24', '26'],
  '26': ['11', '25', '27'],
  '27': ['20', '26'],
  '30': ['31', '37'],
  '31': ['30', '32'],
  '32': ['31', '33'],
```

```
  '33': ['32', '34'],
  '34': ['33', '35'],
  '35': ['22', '34', '36'],
  '36': ['21', '35', '37'],
  '37': ['30', '36']}}
```

Il dizionario precedente `connectivityGraph` contiene informazioni sulla connettività dell'attuale Rigetti dispositivo.

Scegli il IonQ Aria-1 dispositivo

Per il IonQ Aria-1 device, `connectivityGraph` è vuoto, come illustrato nell'esempio seguente, perché il dispositivo offre all-to-allconnettività. Pertanto, non `connectivityGraph` è necessario un dettaglio.

```
# or choose the IonQ Aria-1 device
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")

# take a look at the device connectivity graph
device.properties.dict()['paradigm']['connectivity']
```

```
{'fullyConnected': True, 'connectivityGraph': {}}
```

Come illustrato nell'esempio seguente, è possibile modificare il shots (default=1000), `poll_timeout_seconds` (default = 432000 = 5 giorni), `poll_interval_seconds` (default = 1) e la posizione del bucket S3 (`s3_location`) in cui verranno archiviati i risultati se scegli di specificare una posizione diversa dal bucket predefinito.

```
my_task = device.run(circ, s3_location = 'amazon-braket-my-folder', shots=100,
 poll_timeout_seconds = 100, poll_interval_seconds = 10)
```

Il IonQ e Rigetti i dispositivi compilano automaticamente il circuito fornito nei rispettivi set di porte nativi e ne mappano l'estratto qubit da indici a fisici qubits sul QPU rispettivo.

> **Note**
>
> QPUi dispositivi hanno una capacità limitata. Una volta raggiunta la capacità, è possibile prevedere tempi di attesa più lunghi.

Amazon Braket può eseguire attività QPU quantistiche entro determinate finestre di disponibilità, ma puoi comunque inviare attività quantistiche in qualsiasi momento (24 ore su 24, 7 giorni su 7) perché tutti i dati e i metadati corrispondenti vengono archiviati in modo affidabile nel bucket S3 appropriato. Come illustrato nella sezione successiva, puoi ripristinare il tuo task quantistico utilizzando il tuo ID di attività quantistica univoco. `AwsQuantumTask`

Ispezione dei circuiti compilati

Quando un circuito funziona su un dispositivo hardware, deve essere compilato in un formato accettabile, ad esempio trasponendo il circuito alle porte native supportate da. QPU L'ispezione dell'effettivo output compilato può essere molto utile per scopi di debug. È possibile visualizzare questo circuito per entrambi Rigetti e IQM dispositivi che utilizzano il codice seguente.

```
task = AwsQuantumTask(arn=task_id, aws_session=session)
# after task finished
task_result = task.result()
compiled_circuit = task_result.get_compiled_circuit()
```

> **Note**
>
> Oggi non è possibile visualizzare il circuito compilato per IonQ dispositivi.

Quando verrà eseguita la mia attività quantistica?

Quando invii un circuito, Amazon Braket lo invia al dispositivo specificato. Quantum Processing Unit (QPU) e le attività quantistiche del simulatore su richiesta vengono messe in coda ed elaborate nell'ordine in cui vengono ricevute. Il tempo necessario per elaborare l'attività quantistica dopo l'invio varia a seconda del numero e della complessità delle attività inviate da altri clienti Amazon Braket e della disponibilità di QPU quelle selezionate.

In questa sezione:

- QPUfinestre di disponibilità e stato
- Visibilità della coda
- Configura e-mail o notifiche SMS

QPUfinestre di disponibilità e stato

QPUla disponibilità varia da dispositivo a dispositivo.

Nella pagina Dispositivi della console Amazon Braket, puoi visualizzare le finestre di disponibilità attuali e future e lo stato del dispositivo. Inoltre, ogni pagina del dispositivo mostra le profondità di coda individuali per attività quantistiche e lavori ibridi.

Un dispositivo è considerato offline se non è disponibile per i clienti, indipendentemente dalla finestra di disponibilità. Ad esempio, potrebbe essere offline a causa di manutenzione programmata, aggiornamenti o problemi operativi.

Visibilità della coda

Prima di inviare un'attività quantistica o un lavoro ibrido, puoi visualizzare quante attività quantistiche o ibride hai davanti a te controllando la profondità della coda del dispositivo.

Profondità della coda

Queue depth si riferisce al numero di attività quantistiche e lavori ibridi in coda per un particolare dispositivo. Le attività quantistiche di un dispositivo e il conteggio delle code di lavoro ibride sono accessibili tramite il Braket Software Development Kit (SDK) oppure Amazon Braket Management Console.

1. La profondità della coda delle attività si riferisce al numero totale di attività quantistiche attualmente in attesa di essere eseguite con priorità normale.
2. La profondità della coda delle attività prioritarie si riferisce al numero totale di attività quantistiche inviate in attesa di essere eseguite Amazon Braket Hybrid Jobs. Queste attività vengono eseguite prima delle attività autonome.
3. La profondità della coda dei lavori ibridi si riferisce al numero totale di lavori ibridi attualmente in coda su un dispositivo. Quantum tasks i lavori inviati come parte di un lavoro ibrido hanno la priorità e sono aggregati nel Priority Task Queue.

I clienti che desiderano visualizzare la profondità della coda tramite il Braket SDK possono modificare il seguente frammento di codice per ottenere la posizione in coda del loro task quantistico o del loro lavoro ibrido:

```
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")
```

```
# returns the number of quantum tasks queued on the device
print(device.queue_depth().quantum_tasks)
{<QueueType.NORMAL: 'Normal'>: '0', <QueueType.PRIORITY: 'Priority'>: '0'}


# returns the number of hybrid jobs queued on the device
print(device.queue_depth().jobs)
'3'
```

L'invio di un'attività quantistica o di un lavoro ibrido a QPU può causare lo stato del carico di lavoro. `QUEUED` Amazon Braket offre ai clienti visibilità sulle attività quantistiche e sulla posizione ibrida nella coda dei lavori.

Posizione in coda

Queue position si riferisce alla posizione corrente dell'attività quantistica o del lavoro ibrido all'interno della rispettiva coda del dispositivo. Può essere ottenuto per attività quantistiche o lavori ibridi tramite il Braket Software Development Kit (SDK) oppure Amazon Braket Management Console.

I clienti che desiderano visualizzare la posizione della coda tramite il Braket SDK possono modificare il seguente frammento di codice per ottenere la posizione in coda del loro task quantistico o del loro lavoro ibrido:

```
# choose the device to run your circuit
device = AwsDevice("arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet")

#execute the circuit
task = device.run(bell, s3_folder, shots=100)

# retrieve the queue position information
print(task.queue_position().queue_position)

# Returns the number of Quantum Tasks queued ahead of you
'2'


from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    "arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet",
    source_module="algorithm_script.py",
```

```
    entry_point="algorithm_script:start_here",
    wait_until_complete=False
)

# retrieve the queue position information
print(job.queue_position().queue_position)
'3' # returns the number of hybrid jobs queued ahead of you
```

Configura e-mail o notifiche SMS

Amazon Braket invia eventi ad Amazon EventBridge quando cambia la disponibilità di un'attività o quando QPU cambia lo stato di un'attività quantistica. Segui questi passaggi per ricevere notifiche di modifica dello stato del dispositivo e delle attività quantistiche tramite e-mail o messaggio: SMS

1. Crea un SNS argomento Amazon e un abbonamento per inviare e-mail oSMS. La disponibilità della posta elettronica o SMS dipende dalla tua regione. Per ulteriori informazioni, consulta la sezione Guida introduttiva ad Amazon SNS e invio SMS di messaggi.
2. Crea una regola EventBridge che attivi le notifiche relative al tuo SNS argomento. Per ulteriori informazioni, consulta Monitoring Amazon Braket with Amazon. EventBridge

(Facoltativo) Configura le notifiche SNS

Puoi configurare le notifiche tramite Amazon Simple Notification Service (SNS) in modo da ricevere un avviso quando l'attività quantistica di Amazon Braket è completata. Le notifiche attive sono utili se prevedi un lungo tempo di attesa, ad esempio quando invii un'attività quantistica di grandi dimensioni o quando invii un'attività quantistica al di fuori della finestra di disponibilità di un dispositivo. Se non volete attendere il completamento dell'operazione quantistica, potete impostare una notifica. SNS

Un notebook Amazon Braket ti guida attraverso i passaggi di configurazione. Per ulteriori informazioni, consulta gli esempi di Amazon Braket su GitHub e, in particolare, il notebook di esempio per l'impostazione delle notifiche.

Gestione del tuo lavoro in Amazon Braket Hybrid

Questa sezione fornisce istruzioni su come gestire i lavori ibridi in Amazon Braket.

Puoi accedere ai lavori ibridi in Braket utilizzando:

- Amazon Braket Python. SDK

- La console Amazon Braket.
- Amazon Braket API.

In questa sezione:

Configura l'istanza di job ibrida per eseguire lo script dell'algoritmo

A seconda dell'algoritmo, potresti avere requisiti diversi. Per impostazione predefinita, Amazon Braket esegue lo script dell'algoritmo su un'`ml.m5.large`istanza. Tuttavia, puoi personalizzare questo tipo di istanza quando crei un lavoro ibrido utilizzando il seguente argomento di importazione e configurazione.

```
from braket.jobs.config import InstanceConfig

job = AwsQuantumJob.create(
    ...
    instance_config=InstanceConfig(instanceType="ml.p3.8xlarge"), # Use NVIDIA Tesla
 V100 instance with 4 GPUs.
    ...
    ),
```

Se state eseguendo una simulazione incorporata e avete specificato un dispositivo locale nella configurazione del dispositivo, potrete inoltre richiedere più di un'istanza InstanceConfig specificando instanceCount e impostando che sia maggiore di una. Il limite massimo è 5. Ad esempio, puoi scegliere 3 istanze come segue.

```
from braket.jobs.config import InstanceConfig
job = AwsQuantumJob.create(
    ...
```

```
    instance_config=InstanceConfig(instanceType="ml.p3.8xlarge", instanceCount=3), #
Use 3 NVIDIA Tesla V100
    ...
    ),
```

Quando utilizzi più istanze, valuta la possibilità di distribuire il job ibrido utilizzando la funzionalità data parallel. Vedi il seguente esempio di notebook per maggiori dettagli su come vedere questo esempio di Braket.

Le tre tabelle seguenti elencano i tipi di istanze e le specifiche disponibili per le istanze di calcolo standard, ottimizzate per il calcolo e accelerate.

> **Note**
>
> Per visualizzare le quote predefinite delle istanze di calcolo classiche per Hybrid Jobs, consulta questa pagina.

Istanza	v CPU	Memoria
ml.m5.large (impostazione predefinita)	2	8 GiB
ml.m5.xlarge	4	16 GiB
ml.m5.2xlarge	8	32 GiB
ml.m5.4xlarge	16	64 GiB
ml.m5.12xlarge	48	192 GiB
ml.m5.24xlarge	96	384 GiB
ml.m4.xlarge	4	16 GiB
ml.m4.2xlarge	8	32 GiB
ml.m4.4xlarge	16	64 GiB
ml.m4.10xlarge	40	256 GiB

Istanze a calcolo ottimizzato	v CPU	Memoria
ml.c4.xlarge	4	7,5 GiB
ml.c4.2xlarge	8	15 GiB
ml.c4.4xlarge	16	30 GiB
ml.c4.8xlarge	36	192 GiB
ml.c5.xlarge	4	8 GiB
ml.c5.2xlarge	8	16 GiB
ml.c5.4xlarge	16	32 GiB
ml.c5.9xlarge	36	72 GiB
ml.c5.18xlarge	72	144 GiB
ml.c5n.xlarge	4	10,5 GiB
ml. c 5 n. 2 x grande	8	21 GiB
ml. c 5 n. 4 x grande	16	42 GiB
ml. c 5 n. 9 x grande	36	96 GiB
ml. c5 n. 18 x grande	72	192 GiB

Istanze di calcolo accelerato	v CPU	Memoria
ml.p2.xlarge	4	61 GiB
ml.p2.8xlarge	32	488 GiB
ml.p2.16xlarge	64	732 GiB
ml.p3.2xlarge	8	61 GiB

Istanze di calcolo accelerato	v CPU	Memoria
ml.p3.8xlarge	32	24 GiB
ml.p3.16xlarge	64	488 GiB
ml.g4dn.xlarge	4	16 GiB
ml.g4dn.2xlarge	8	32 GiB
ml.g4dn.4xlarge	16	64 GiB
ml.g4dn.8xlarge	32	128 GiB
ml.g4dn.12xlarge	48	192 GiB
ml.g4dn.16xlarge	64	256 GiB

> **Note**
>
> le istanze p3 non sono disponibili in us-west-1. Se il tuo lavoro ibrido non è in grado di fornire la capacità di calcolo ML richiesta, usa un'altra regione.

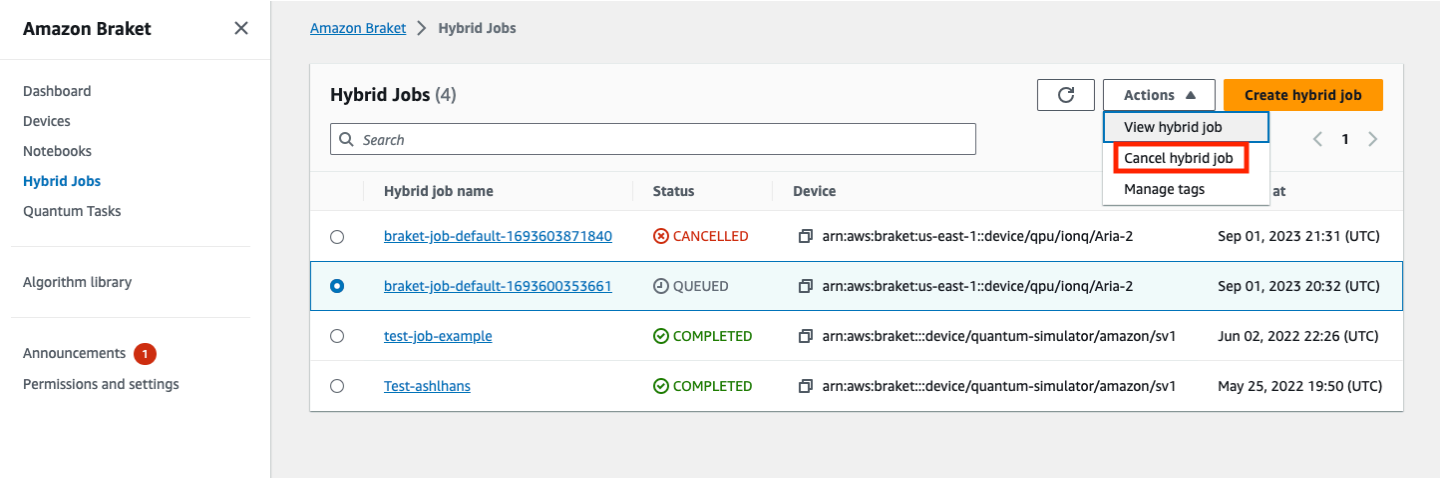
Ogni istanza utilizza una configurazione predefinita di archiviazione dei dati (SSD) di 30 GB. È tuttavia possibile regolare lo spazio di archiviazione nello stesso modo in cui si configura il`instanceType`. L'esempio seguente mostra come aumentare lo spazio di archiviazione totale a 50 GB.

```
from braket.jobs.config import InstanceConfig

job = AwsQuantumJob.create(
    ...
    instance_config=InstanceConfig(
        instanceType="ml.p3.8xlarge",
        volumeSizeInGb=50,
    ),
    ...
    ),
```

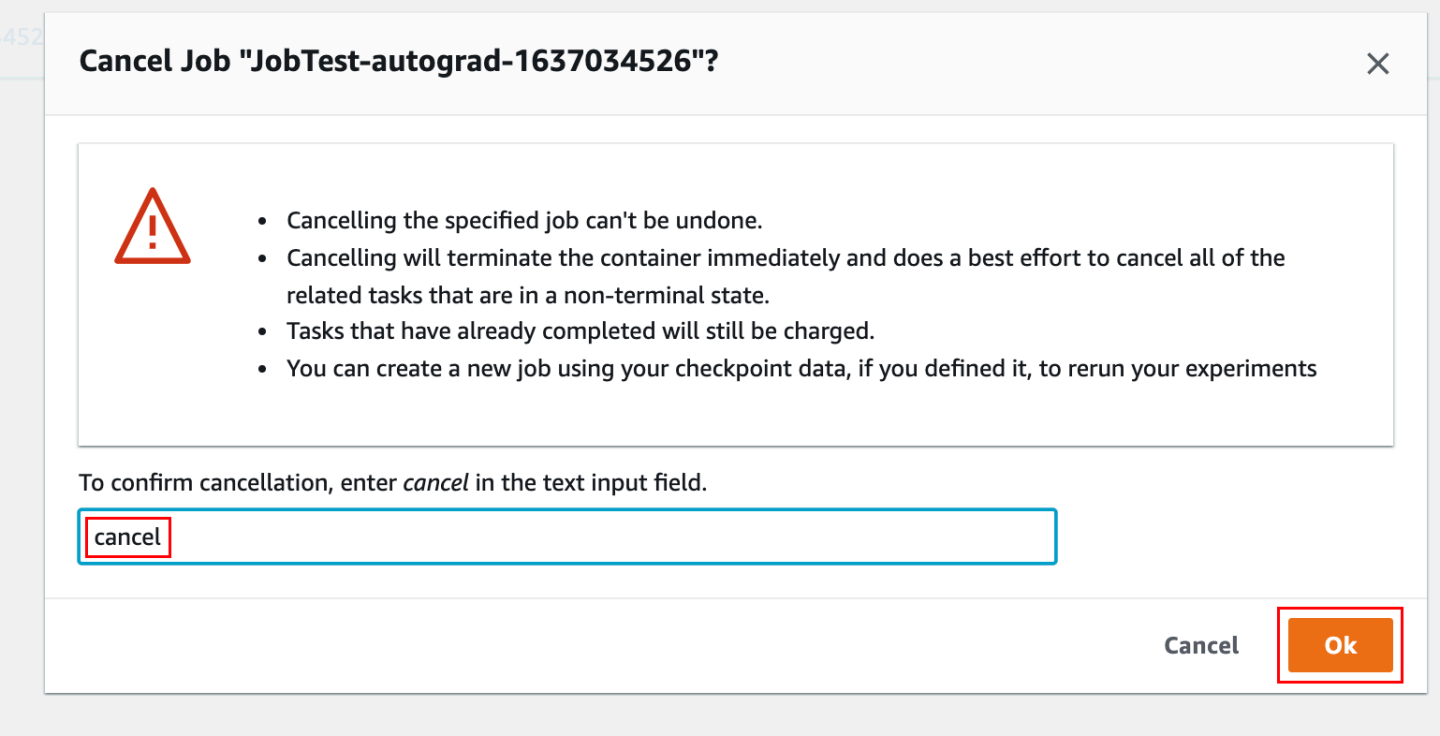
Annullare un Job ibrido

Potrebbe essere necessario annullare un processo ibrido in uno stato non terminale. Questa operazione può essere eseguita nella console o con il codice.

Per annullare il lavoro ibrido nella console, seleziona il lavoro ibrido da annullare dalla pagina Lavori ibridi, quindi seleziona Annulla lavoro ibrido dal menu a discesa Azioni.



Per confermare l'annullamento, inserisci cancel nel campo di immissione quando richiesto, quindi seleziona OK.



Per annullare il tuo lavoro ibrido utilizzando il codice di Braket SDK Python, usa per identificare `job_arn` il lavoro ibrido e poi chiama `cancel` il comando su di esso come mostrato nel codice seguente.

```
job = AwsQuantumJob(arn=job_arn)
job.cancel()
```

Il `cancel` comando termina immediatamente il classico contenitore di lavoro ibrido e fa del suo meglio per annullare tutte le attività quantistiche correlate che sono ancora in uno stato non terminale.

Utilizzo della compilazione parametrica per velocizzare i lavori ibridi

Amazon Braket supporta la compilazione parametrica su alcuni. QPUs Ciò consente di ridurre il sovraccarico associato alla fase di compilazione, dal punto di vista computazionalmente costoso, compilando un circuito una sola volta e non per ogni iterazione del tuo algoritmo ibrido. Ciò può migliorare notevolmente i tempi di esecuzione per Hybrid Jobs, poiché si evita la necessità di ricompilare il circuito in ogni fase. Basta inviare circuiti parametrizzati a uno dei nostri Braket Hybrid QPUs Job supportati. Per lavori ibridi di lunga durata, Braket utilizza automaticamente i dati di calibrazione aggiornati del fornitore di hardware durante la compilazione del circuito per garantire risultati della massima qualità.

Per creare un circuito parametrico, devi prima fornire i parametri come input nello script dell'algoritmo. In questo esempio, utilizziamo un piccolo circuito parametrico e ignoriamo qualsiasi elaborazione classica tra ogni iterazione. Per i carichi di lavoro tipici, è necessario inviare molti circuiti in batch ed eseguire l'elaborazione classica, ad esempio l'aggiornamento dei parametri in ogni iterazione.

```
import os

from braket.aws import AwsDevice
from braket.circuits import Circuit, FreeParameter

def start_here():

    print("Test job started.")

    # Use the device declared in the job script
    device = AwsDevice(os.environ["AMZN_BRAKET_DEVICE_ARN"])

    circuit = Circuit().rx(0, FreeParameter("theta"))
    parameter_list = [0.1, 0.2, 0.3]
```

```
    for parameter in parameter_list:
        result = device.run(circuit, shots=1000, inputs={"theta": parameter})

    print("Test job completed.")
```

È possibile inviare lo script dell'algoritmo per l'esecuzione come Hybrid Job con il seguente script di processo. Quando si esegue Hybrid Job su un QPU dispositivo che supporta la compilazione parametrica, il circuito viene compilato solo alla prima esecuzione. Nelle esecuzioni successive, il circuito compilato viene riutilizzato, aumentando le prestazioni di runtime di Hybrid Job senza righe di codice aggiuntive.

```
from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    device=device_arn,
    source_module="algorithm_script.py",
)
```

> **Note**
>
> La compilazione parametrica è supportata su tutti i formati superconduttori basati su gate QPUs Rigetti Computing ad eccezione dei programmi a livello di impulso.

Utilizzo PennyLane con Amazon Braket

Gli algoritmi ibridi sono algoritmi che contengono istruzioni sia classiche che quantistiche. Le istruzioni classiche vengono eseguite su hardware classico (un'EC2istanza o un laptop) e le istruzioni quantistiche vengono eseguite su un simulatore o su un computer quantistico. Si consiglia di eseguire algoritmi ibridi utilizzando la funzionalità Hybrid Jobs. Per ulteriori informazioni, consulta Quando usare Amazon Braket Jobs.

Amazon Braket ti consente di configurare ed eseguire algoritmi quantistici ibridi con l'assistenza del PennyLane plug-in Amazon Braket o con Amazon Braket Python ed esempi di repository di notebook. SDK I notebook di esempio Amazon Braket, basati suSDK, consentono di configurare ed eseguire determinati algoritmi ibridi senza il plug-in. PennyLane Tuttavia, lo consigliamo PennyLane perché offre un'esperienza più ricca.

Informazioni sugli algoritmi quantistici ibridi

Gli algoritmi quantistici ibridi sono importanti per il settore odierno perché i dispositivi informatici quantistici contemporanei generalmente producono rumore e quindi errori. Ogni porta quantistica aggiunta a un calcolo aumenta la possibilità di aggiungere rumore; pertanto, gli algoritmi di lunga durata possono essere sopraffatti dal rumore, con conseguenti errori di calcolo.

Algoritmi quantistici puri come quelli di Shor (esempio Quantum Phase Estimation) o Grover (esempio Grover) richiedono migliaia o milioni di operazioni. Per questo motivo, possono essere poco pratici per i dispositivi quantistici esistenti, che vengono generalmente definiti dispositivi quantistici () rumorosi su scala intermedia. NISQ

Negli algoritmi quantistici ibridi, le unità di elaborazione quantistica (QPUs) funzionano come coprocessori per la versione classica, in particolare per velocizzare determinati calcoli in un algoritmo classicoCPUs. Le esecuzioni dei circuiti diventano molto più brevi, alla portata delle funzionalità dei dispositivi odierni.

In questa sezione:

- Amazon Braket con PennyLane
- Algoritmi ibridi nei notebook di esempio Amazon Braket
- Algoritmi ibridi con simulatori integrati PennyLane
- Aggiungi il gradiente con i simulatori Amazon PennyLane Braket
- Utilizzo di Hybrid Jobs ed esecuzione di un algoritmo PennyLane QAOA
- Esegui carichi di lavoro ibridi con simulatori PennyLane integrati

Amazon Braket con PennyLane

Amazon Braket fornisce supporto per PennyLaneun framework software open source basato sul concetto di programmazione quantistica differenziabile. Puoi usare questo framework per addestrare i circuiti quantistici nello stesso modo in cui addestreresti una rete neurale per trovare soluzioni a problemi computazionali di chimica quantistica, apprendimento automatico quantistico e ottimizzazione.

La PennyLane libreria fornisce interfacce a strumenti di apprendimento automatico familiari, tra cui PyTorch e, per rendere l'addestramento dei circuiti quantistici TensorFlow rapido e intuitivo.

- La PennyLane libreria -— PennyLane è preinstallata in Amazon Notebook Braket. Per accedere a Amazon Proteggi i dispositivi da PennyLane, apri un notebook e importa la PennyLane libreria con il seguente comando.

```
import pennylane as qml
```

I taccuini tutorial ti aiutano a iniziare rapidamente. In alternativa, puoi usare su PennyLane Amazon Staffa tra quelle IDE a tua scelta.

- La Amazon PennyLane Plugin Braket: per utilizzare il tuoIDE, puoi installare il Amazon PennyLane Plugin Braket manualmente. Il plugin si connette PennyLane ad Amazon Braket Python SDK, quindi puoi eseguire circuiti su PennyLane Amazon Dispositivi Braket. Per installare il PennyLane plugin, usa il seguente comando.

```
pip install amazon-braket-pennylane-plugin
```

L'esempio seguente mostra come configurare l'accesso a Amazon Dispositivi di supporto in: PennyLane

```
# to use SV1
import pennylane as qml
sv1 = qml.device("braket.aws.qubit", device_arn="arn:aws:braket:::device/quantum-
simulator/amazon/sv1", wires=2)

# to run a circuit:
@qml.qnode(sv1)
def circuit(x):
    qml.RZ(x, wires=0)
    qml.CNOT(wires=[0,1])
    qml.RY(x, wires=1)
    return qml.expval(qml.PauliZ(1))

result = circuit(0.543)


#To use the local sim:
local = qml.device("braket.local.qubit", wires=2)
```

Per esempi di tutorial e ulteriori informazioni a riguardo PennyLane, consulta l'archivio degli esempi di Amazon Braket.

Il Amazon Il PennyLane plug-in Braket ti consente di passare da uno all'altro Amazon Braket QPU e dispositivi di simulazione integrati PennyLane con un'unica riga di codice. Ne offre due Amazon Dispositivi quantistici Braket con cui lavorare: PennyLane

- `braket.aws.qubit`per correre con Amazon Dispositivi quantistici di Braket Service, inclusi QPUs simulatori
- `braket.local.qubit`per correre con Amazon Il simulatore locale SDK di Braket

Il Amazon Il PennyLane plugin Braket è open source. Puoi installarlo dal GitHub repository dei PennyLane plugin.

Per ulteriori informazioni in merito PennyLane, consulta la documentazione sul PennyLane sito Web.

Algoritmi ibridi nei notebook di esempio Amazon Braket

Amazon Braket fornisce una serie di notebook di esempio che non si basano sul PennyLane plug-in per l'esecuzione di algoritmi ibridi. Puoi iniziare con uno qualsiasi di questi notebook di esempio ibridi Amazon Braket che illustrano metodi variazionali, come Quantum Approximate Optimization Algorithm () o Variational Quantum Eigensolver (). QAOA VQE

I notebook di esempio Amazon Braket si basano su Amazon Braket Python. SDK SDKFornisce un framework per interagire con i dispositivi hardware di elaborazione quantistica tramite Amazon Staffa. È una libreria open source progettata per assistervi nella parte quantistica del vostro flusso di lavoro ibrido.

Puoi esplorare Amazon Fatti strada ancora di più con i nostri taccuini di esempio.

Algoritmi ibridi con simulatori integrati PennyLane

Amazon Braket Hybrid Jobs ora include simulatori integrati ad alte prestazioni CPU GPU basati su. PennyLane Questa famiglia di simulatori integrati può essere incorporata direttamente nel tuo contenitore di lavori ibridi e include il veloce `lightning.qubit` simulatore di vettore di stato, la libreria di simulatori accelerati `lightning.gpu` utilizzando e altri. NVIDIA cuQuantum Questi simulatori integrati sono ideali per algoritmi variazionali come l'apprendimento automatico quantistico, che possono trarre vantaggio da metodi avanzati come il metodo di differenziazione aggiuntiva. È possibile eseguire questi simulatori incorporati su una o più istanze. CPU GPU

Con Hybrid Jobs, ora puoi eseguire il codice del tuo algoritmo variazionale utilizzando una combinazione di un coprocessore classico e un QPU Amazon Simulatore on-demand Braket come SV1, o utilizzando direttamente il simulatore incorporato di. PennyLane

Il simulatore incorporato è già disponibile con il contenitore Hybrid Jobs, devi semplicemente decorare la tua funzione Python principale con `@hybrid_job` il decoratore. Per utilizzare il PennyLane `lightning.gpu` simulatore, è inoltre necessario specificare un'GPUistanza `InstanceConfig` come mostrato nel seguente frammento di codice:

```
import pennylane as qml
from braket.jobs import hybird_job
from braket.jobs.config import InstanceConfig


@hybrid_job(device="local:pennylane/lightning.gpu",
 instance_config=InstanceConfig(instanceType="ml.p3.8xlarge"))
def function(wires):
    dev = qml.device("lightning.gpu", wires=wires)
    ...
```

Fate riferimento al notebook di esempio per iniziare a utilizzare un simulatore PennyLane incorporato con Hybrid Jobs.

Aggiungi il gradiente con i simulatori Amazon PennyLane Braket

Con il plugin PennyLane plug-in per Amazon Braket, puoi calcolare i gradienti utilizzando il metodo di differenziazione adjoint quando esegui sul simulatore vettoriale dello stato locale o. SV1

Nota: per utilizzare il metodo di differenziazione aggiuntiva, devi specificare nel tuo, e non. **`diff_method='device'` `qnode`** `diff_method='adjoint'` Guarda l'esempio seguente.

```
device_arn = "arn:aws:braket:::device/quantum-simulator/amazon/sv1"
dev = qml.device("braket.aws.qubit", wires=wires, shots=0, device_arn=device_arn)

@qml.qnode(dev, diff_method="device")
def cost_function(params):
    circuit(params)
    return qml.expval(cost_h)

gradient = qml.grad(circuit)
initial_gradient = gradient(params0)
```

> **Note**
>
> Attualmente, PennyLane calcolerà gli indici di raggruppamento per gli QAOA hamiltoniani e li utilizzerà per dividere gli hamiltoniani in più valori di aspettativa. Se si desidera utilizzare la funzionalità di differenziazione aggiuntiva di SV1 QAOA PennyLane, dovrai ricostruire il costo hamiltoniano rimuovendo gli indici di raggruppamento, in questo modo: `cost_h, mixer_h = qml.qaoa.max_clique(g, constrained=False) cost_h = qml.Hamiltonian(cost_h.coeffs, cost_h.ops)`

Utilizzo di Hybrid Jobs ed esecuzione di un algoritmo PennyLane QAOA

In questa sezione, utilizzerai ciò che hai imparato per scrivere un vero programma ibrido utilizzando PennyLane la compilazione parametrica. Lo script dell'algoritmo viene utilizzato per risolvere un problema relativo all'algoritmo di ottimizzazione approssimativa quantistica (). QAOA Il programma crea una funzione di costo corrispondente a un classico problema di ottimizzazione Max Cut, specifica un circuito quantistico parametrizzato e utilizza un semplice metodo di discesa del gradiente per ottimizzare i parametri in modo da ridurre al minimo la funzione di costo. In questo esempio, generiamo il grafico del problema nello script dell'algoritmo per semplicità, ma per i casi d'uso più tipici la migliore pratica consiste nel fornire le specifiche del problema attraverso un canale dedicato nella configurazione dei dati di input. L'`parametrize_differentiable`impostazione predefinita del flag consente di `True` ottenere automaticamente i vantaggi di prestazioni di runtime migliorate grazie alla compilazione parametrica su Support. QPUs

```
import os
import json
import time

from braket.jobs import save_job_result
from braket.jobs.metrics import log_metric

import networkx as nx
import pennylane as qml
from pennylane import numpy as np
from matplotlib import pyplot as plt

def init_pl_device(device_arn, num_nodes, shots, max_parallel):
    return qml.device(
        "braket.aws.qubit",
        device_arn=device_arn,
```

```
        wires=num_nodes,
        shots=shots,
        # Set s3_destination_folder=None to output task results to a default folder
        s3_destination_folder=None,
        parallel=True,
        max_parallel=max_parallel,
        parametrize_differentiable=True, # This flag is True by default.
    )

def start_here():
    input_dir = os.environ["AMZN_BRAKET_INPUT_DIR"]
    output_dir = os.environ["AMZN_BRAKET_JOB_RESULTS_DIR"]
    job_name = os.environ["AMZN_BRAKET_JOB_NAME"]
    checkpoint_dir = os.environ["AMZN_BRAKET_CHECKPOINT_DIR"]
    hp_file = os.environ["AMZN_BRAKET_HP_FILE"]
    device_arn = os.environ["AMZN_BRAKET_DEVICE_ARN"]

    # Read the hyperparameters
    with open(hp_file, "r") as f:
        hyperparams = json.load(f)

    p = int(hyperparams["p"])
    seed = int(hyperparams["seed"])
    max_parallel = int(hyperparams["max_parallel"])
    num_iterations = int(hyperparams["num_iterations"])
    stepsize = float(hyperparams["stepsize"])
    shots = int(hyperparams["shots"])

    # Generate random graph
    num_nodes = 6
    num_edges = 8
    graph_seed = 1967
    g = nx.gnm_random_graph(num_nodes, num_edges, seed=graph_seed)

    # Output figure to file
    positions = nx.spring_layout(g, seed=seed)
    nx.draw(g, with_labels=True, pos=positions, node_size=600)
    plt.savefig(f"{output_dir}/graph.png")

    # Set up the QAOA problem
    cost_h, mixer_h = qml.qaoa.maxcut(g)

    def qaoa_layer(gamma, alpha):
        qml.qaoa.cost_layer(gamma, cost_h)
```

```
        qml.qaoa.mixer_layer(alpha, mixer_h)

    def circuit(params, **kwargs):
        for i in range(num_nodes):
            qml.Hadamard(wires=i)
        qml.layer(qaoa_layer, p, params[0], params[1])

    dev = init_pl_device(device_arn, num_nodes, shots, max_parallel)

    np.random.seed(seed)
    cost_function = qml.ExpvalCost(circuit, cost_h, dev, optimize=True)
    params = 0.01 * np.random.uniform(size=[2, p])

    optimizer = qml.GradientDescentOptimizer(stepsize=stepsize)
    print("Optimization start")

    for iteration in range(num_iterations):
        t0 = time.time()

        # Evaluates the cost, then does a gradient step to new params
        params, cost_before = optimizer.step_and_cost(cost_function, params)
        # Convert cost_before to a float so it's easier to handle
        cost_before = float(cost_before)

        t1 = time.time()

        if iteration == 0:
            print("Initial cost:", cost_before)
        else:
            print(f"Cost at step {iteration}:", cost_before)

        # Log the current loss as a metric
        log_metric(
            metric_name="Cost",
            value=cost_before,
            iteration_number=iteration,
        )

        print(f"Completed iteration {iteration + 1}")
        print(f"Time to complete iteration: {t1 - t0} seconds")

    final_cost = float(cost_function(params))
    log_metric(
        metric_name="Cost",
```

```
        value=final_cost,
        iteration_number=num_iterations,
    )

    # We're done with the hybrid job, so save the result.
    # This will be returned in job.result()
    save_job_result({"params": params.numpy().tolist(), "cost": final_cost})
```

> **Note**
>
> La compilazione parametrica è supportata su tutti i formati superconduttori basati su gate QPUs Rigetti Computing ad eccezione dei programmi a livello di impulso.

Esegui carichi di lavoro ibridi con simulatori PennyLane integrati

Diamo un'occhiata a come utilizzare i simulatori integrati di Amazon Braket Hybrid Jobs per eseguire carichi di lavoro ibridi. PennyLane Il simulatore integrato GPU basato su Pennylane utilizza la libreria cuQuantum Nvidia per accelerare `lightning.gpu` le simulazioni di circuiti. Il GPU simulatore integrato è preconfigurato in tutti i contenitori di lavoro Braket che gli utenti possono utilizzare immediatamente. In questa pagina, ti mostriamo come utilizzarlo per `lightning.gpu` velocizzare i carichi di lavoro ibridi.

Utilizzo **`lightning.gpu`** per carichi di lavoro QAOA

Considerate gli esempi di Quantum Approximate Optimization Algorithm (QAOA) tratti da questo taccuino. Per selezionare un simulatore incorporato, si specifica che l'`device`argomento sia una stringa del formato:. `"local:<provider>/<simulator_name>"` Ad esempio, imposteresti `"local:pennylane/lightning.gpu"` per`lightning.gpu`. La stringa del dispositivo fornita a Hybrid Job all'avvio viene passata al lavoro come variabile di ambiente`"AMZN_BRAKET_DEVICE_ARN"`.

```
device_string = os.environ["AMZN_BRAKET_DEVICE_ARN"]
prefix, device_name = device_string.split("/")
device = qml.device(simulator_name, wires=n_wires)
```

In questa pagina, confrontiamo i due simulatori vettoriali di PennyLane stato incorporati `lightning.qubit` (CPUbasato) e `lightning.gpu` (GPUbasato). Dovrai fornire ai simulatori alcune scomposizioni di gate personalizzate per calcolare vari gradienti.

Ora sei pronto per preparare lo script ibrido di avvio del lavoro. Eseguirete l'QAOAalgoritmo utilizzando due tipi di istanza: `m5.2xlarge` ep`3.2xlarge`. Il tipo di `m5.2xlarge` istanza è paragonabile a un laptop per sviluppatori standard. `p3.2xlarge`Si tratta di un'istanza di elaborazione accelerata che ha una singola NVIDIA Volta GPU con 16 GB di memoria.

`hyperparameters`Per tutti i tuoi lavori ibridi sarà lo stesso. Tutto quello che devi fare per provare diverse istanze e simulatori è cambiare due righe come segue.

```
# Specify device that the hybrid job will primarily be targeting
device = "local:pennylane/lightning.qubit"
# Run on a CPU based instance with about as much power as a laptop
instance_config = InstanceConfig(instanceType='ml.m5.2xlarge')
```

oppure:

```
# Specify device that the hybrid job will primarily be targeting
device = "local:pennylane/lightning.gpu"
# Run on an inexpensive GPU based instance
instance_config = InstanceConfig(instanceType='ml.p3.2xlarge')
```

> **Note**
>
> Se si specifica l'`instance_config`as utilizzando un'istanza GPU basata, ma si sceglie come simulatore CPU basato su incorporato (`lightning.qubit`), non GPU verrà utilizzato. `device` Assicurati di utilizzare il simulatore GPU basato su embedded se desideri scegliere come target il! GPU

Innanzitutto, puoi creare due lavori ibridi e risolvere Max-Cut utilizzando un grafico con QAOA 18 vertici. Ciò si traduce in un circuito da 18 qubit, relativamente piccolo e facile da eseguire rapidamente sul laptop o sull'istanza. `m5.2xlarge`

```
num_nodes = 18
num_edges = 24
seed = 1967

graph = nx.gnm_random_graph(num_nodes, num_edges, seed=seed)

# And similarly for the p3 job
m5_job = AwsQuantumJob.create(
```

```
    device=device,
    source_module="qaoa_source",
    job_name="qaoa-m5-" + str(int(time.time())),
    image_uri=image_uri,
    # Relative to the source_module
    entry_point="qaoa_source.qaoa_algorithm_script",
    copy_checkpoints_from_job=None,
    instance_config=instance_config,
    # general parameters
    hyperparameters=hyperparameters,
    input_data={"input-graph": input_file_path},
    wait_until_complete=True,
)
```

Il tempo di iterazione medio per l'istanza è di circa 25 secondi, mentre per l'`m5.2xlarge`istanza è di circa 12 secondi. `p3.2xlarge` Per questo flusso di lavoro da 18 qubit, l'GPUistanza ci offre una velocità di 2 volte superiore. Se guardi la pagina dei prezzi di Amazon Braket Hybrid Jobs, puoi vedere che il costo al minuto per un'`m5.2xlarge`istanza è di 0,00768 USD, mentre per l'`p3.2xlarge`istanza è di 0,06375 USD. L'esecuzione per 5 iterazioni totali, come hai fatto qui, costerebbe 0,016 USD utilizzando l'istanza o 0,06375 USD utilizzando l'CPUistanza, entrambe piuttosto economiche! GPU

Ora rendiamo il problema più difficile e proviamo a risolvere un problema Max-Cut su un grafico a 24 vertici, che si tradurrà in 24 qubit. Esegui nuovamente i processi ibridi sulle stesse due istanze e confronta i costi.

> **Note**
>
> Vedrai che il tempo necessario per eseguire questo processo ibrido sull'CPUistanza potrebbe essere di circa cinque ore!

```
num_nodes = 24
num_edges = 36
seed = 1967

graph = nx.gnm_random_graph(num_nodes, num_edges, seed=seed)

# And similarly for the p3 job
m5_big_job = AwsQuantumJob.create(
    device=device,
```

```
    source_module="qaoa_source",
    job_name="qaoa-m5-big-" + str(int(time.time())),
    image_uri=image_uri,
    # Relative to the source_module
    entry_point="qaoa_source.qaoa_algorithm_script",
    copy_checkpoints_from_job=None,
    instance_config=instance_config,
    # general parameters
    hyperparameters=hyperparameters,
    input_data={"input-graph": input_file_path},
    wait_until_complete=True,
)
```

Il tempo medio di iterazione per l'`m5.2xlarge`istanza è di circa un'ora, mentre per l'`p3.2xlarge`istanza è di circa due minuti. Per questo problema più ampio, l'GPUistanza è un ordine di grandezza più veloce! Tutto quello che dovevi fare per trarre vantaggio da questa accelerazione era modificare due righe di codice, sostituendo il tipo di istanza e il simulatore locale utilizzato. L'esecuzione per un totale di 5 iterazioni, come è stato fatto qui, costerebbe circa 2,27072 dollari utilizzando l'istanza o circa 0,775625 dollari utilizzando l'CPUistanza. GPU L'CPUutilizzo non è solo più costoso, ma richiede anche più tempo per l'esecuzione. Accelerazione di questo flusso di lavoro con un'GPUistanza disponibile su AWS, utilizzando PennyLane il simulatore integrato supportato da NVIDIA CuQuantum, consente di eseguire flussi di lavoro con conteggi di qubit intermedi (tra 20 e 30) a un costo totale inferiore e in meno tempo. Ciò significa che puoi sperimentare con l'informatica quantistica anche per problemi troppo grandi per essere eseguiti rapidamente sul tuo laptop o su un'istanza di dimensioni simili.

Apprendimento automatico quantistico e parallelismo dei dati

Se il tuo tipo di carico di lavoro è l'apprendimento automatico quantistico (QML) che si addestra su set di dati, puoi accelerare ulteriormente il carico di lavoro utilizzando il parallelismo dei dati. NelQML, il modello contiene uno o più circuiti quantistici. Il modello può contenere o meno anche reti neurali classiche. Quando si addestra il modello con il set di dati, i parametri del modello vengono aggiornati per ridurre al minimo la funzione di perdita. Di solito viene definita una funzione di perdita per un singolo punto dati e la perdita totale per la perdita media sull'intero set di dati. NelQML, le perdite vengono generalmente calcolate in serie prima di calcolare la media della perdita totale per i calcoli a gradiente. Questa procedura richiede molto tempo, soprattutto quando sono presenti centinaia di punti dati.

Poiché la perdita da un punto dati non dipende da altri punti dati, le perdite possono essere valutate in parallelo! Le perdite e i gradienti associati a diversi punti dati possono essere valutati

contemporaneamente. Questo è noto come parallelismo dei dati. Con SageMaker la sua libreria parallela di dati distribuiti, Amazon Braket Hybrid Jobs ti consente di sfruttare più facilmente il parallelismo dei dati per accelerare la formazione.

Considera il seguente QML carico di lavoro per il parallelismo dei dati, che utilizza il set di dati Sonar del noto repository come esempio di classificazione binaria. UCI Il set di dati Sonar ha 208 punti dati ciascuno con 60 caratteristiche raccolte dai segnali sonar che rimbalzano sui materiali. Ogni punto dati è etichettato come «M» per le miniere o «R» per le rocce. Il nostro QML modello è costituito da uno strato di input, un circuito quantistico come livello nascosto e un livello di output. I livelli di input e output sono reti neurali classiche implementate in. PyTorch Il circuito quantistico è integrato con le reti PyTorch neurali utilizzando il modulo qml.qnn. PennyLane Consulta i nostri taccuini di esempio per maggiori dettagli sul carico di lavoro. Come nell'QAOAesempio precedente, puoi sfruttare la potenza di GPU utilizzando simulatori integrati come quelli PennyLane per migliorare le prestazioni rispetto `lightning.gpu` ai simulatori GPU basati su sistemi integrati. CPU

Per creare un processo ibrido, puoi chiamare `AwsQuantumJob.create` e specificare lo script dell'algoritmo, il dispositivo e altre configurazioni tramite i relativi argomenti delle parole chiave.

```
instance_config = InstanceConfig(instanceType='ml.p3.2xlarge')

hyperparameters={"nwires": "10",
                 "ndata": "32",
                 ...
}

job = AwsQuantumJob.create(
    device="local:pennylane/lightning.gpu",
    source_module="qml_source",
    entry_point="qml_source.train_single",
    hyperparameters=hyperparameters,
    instance_config=instance_config,
    ...
)
```

Per utilizzare il parallelismo dei dati, è necessario modificare alcune righe di codice nello script dell'algoritmo per la libreria SageMaker distribuita per parallelizzare correttamente l'addestramento. Innanzitutto, importate il `smdistributed` pacchetto che svolge la maggior parte del lavoro necessario per distribuire i carichi di lavoro su più e più istanze. GPUs Questo pacchetto è preconfigurato nel Braket e nei contenitori. PyTorch TensorFlow Il `dist` modulo indica al nostro script di algoritmo qual è il numero totale di GPUs per il training (`world_size`) `rank` e la fine `local_rank`

di un GPU core. `rank`è l'indice assoluto di a GPU in tutte le istanze, mentre `local_rank` è l'indice di a GPU all'interno di un'istanza. Ad esempio, se ci sono quattro istanze, ognuna delle quali otto GPUs allocate per l'addestramento, i `rank` valori vanno da 0 a 31 e gli `local_rank` intervalli da 0 a 7.

```
import smdistributed.dataparallel.torch.distributed as dist

dp_info = {
    "world_size": dist.get_world_size(),
    "rank": dist.get_rank(),
    "local_rank": dist.get_local_rank(),
}
batch_size //= dp_info["world_size"] // 8
batch_size = max(batch_size, 1)
```

Successivamente, si definisce un `DistributedSampler` in base a `world_size` `rank` e quindi lo si passa nel caricatore di dati. Questo campionatore evita di GPUs accedere alla stessa porzione di un set di dati.

```
train_sampler = torch.utils.data.distributed.DistributedSampler(
    train_dataset,
    num_replicas=dp_info["world_size"],
    rank=dp_info["rank"]
)
train_loader = torch.utils.data.DataLoader(
    train_dataset,
    batch_size=batch_size,
    shuffle=False,
    num_workers=0,
    pin_memory=True,
    sampler=train_sampler,
)
```

Successivamente, si utilizza la `DistributedDataParallel` classe per abilitare il parallelismo dei dati.

```
from smdistributed.dataparallel.torch.parallel.distributed import
 DistributedDataParallel as DDP

model = DressedQNN(qc_dev).to(device)
model = DDP(model)
torch.cuda.set_device(dp_info["local_rank"])
```

```
model.cuda(dp_info["local_rank"])
```

Quanto sopra sono le modifiche necessarie per utilizzare il parallelismo dei dati. InQML, spesso si desidera salvare i risultati e stampare i progressi della formazione. Se ciascuno GPU esegue il comando di salvataggio e stampa, il registro verrà riempito con le informazioni ripetute e i risultati si sovrascriveranno a vicenda. Per evitare ciò, è possibile salvare e stampare solo da file con 0GPU. `rank`

```
if dp_info["rank"]==0:
    print('elapsed time: ', elapsed)
    torch.save(model.state_dict(), f"{output_dir}/test_local.pt")
    save_job_result({"last loss": loss_before})
```

Amazon Braket Hybrid Jobs supporta i tipi di `ml.p3.16xlarge` istanza per la libreria parallela di SageMaker dati distribuiti. Puoi configurare il tipo di istanza tramite l'`InstanceConfig`argomento in Hybrid Jobs. Affinché la libreria parallela di dati SageMaker distribuiti sappia che il parallelismo dei dati è abilitato, devi aggiungere due iperparametri aggiuntivi, `"sagemaker_distributed_dataparallel_enabled"` impostandoli `"true"` e `"sagemaker_instance_type"` impostandoli sul tipo di istanza che stai utilizzando. Questi due iperparametri vengono utilizzati per pacchetto. `smdistributed` Lo script dell'algoritmo non deve utilizzarli in modo esplicito. In Amazon BraketSDK, fornisce un comodo argomento per le parole chiave. `distribution` `distribution="data_parallel"`Nella creazione di posti di lavoro ibridi, Amazon Braket inserisce SDK automaticamente i due iperparametri per te. Se utilizzi Amazon BraketAPI, devi includere questi due iperparametri.

Con il parallelismo di istanze e dati configurato, ora puoi inviare il tuo lavoro ibrido. Ce ne sono 8 GPUs in un'`ml.p3.16xlarge`istanza. Quando si imposta`instanceCount=1`, il carico di lavoro viene distribuito tra gli 8 componenti GPUs dell'istanza. Se ne imposti `instanceCount` più di uno, il carico di lavoro viene distribuito tra quelli GPUs disponibili in tutte le istanze. Quando si utilizzano più istanze, ogni istanza comporta un addebito in base al tempo di utilizzo. Ad esempio, quando utilizzi quattro istanze, il tempo fatturabile è quattro volte il tempo di esecuzione per istanza perché ci sono quattro istanze che eseguono i carichi di lavoro contemporaneamente.

```
instance_config = InstanceConfig(instanceType='ml.p3.16xlarge',
                                 instanceCount=1,
)

hyperparameters={"nwires": "10",
```

```
                "ndata": "32",
                ...,
}

job = AwsQuantumJob.create(
    device="local:pennylane/lightning.gpu",
    source_module="qml_source",
    entry_point="qml_source.train_dp",
    hyperparameters=hyperparameters,
    instance_config=instance_config,
    distribution="data_parallel",
    ...
)
```

> **Note**
>
> Nella creazione di posti di lavoro ibridi di cui sopra, `train_dp.py` si tratta dello script algoritmico modificato per l'utilizzo del parallelismo dei dati. Tieni presente che il parallelismo dei dati funziona correttamente solo quando modifichi lo script dell'algoritmo in base alla sezione precedente. Se l'opzione di parallelismo dei dati è abilitata senza uno script di algoritmo modificato correttamente, il processo ibrido può generare errori o ciascuno di essi GPU può elaborare ripetutamente la stessa porzione di dati, il che è inefficiente.

Confrontiamo il tempo di esecuzione e il costo in un esempio in cui si addestra un modello con un circuito quantistico a 26 qubit per il problema di classificazione binaria sopra menzionato. L'`ml.p3.16xlarge`istanza utilizzata in questo esempio costa 0,4692 USD al minuto. Senza il parallelismo dei dati, il simulatore impiega circa 45 minuti per addestrare il modello per un'epoca (ovvero oltre 208 punti dati) e costa circa 20 dollari. Con il parallelismo dei dati su 1 e 4 istanze, bastano rispettivamente solo 6 minuti e 1,5 minuti, il che si traduce in circa 2,8 dollari per entrambe. Utilizzando il parallelismo dei dati su 4 istanze, non solo migliorerai il tempo di esecuzione di 30 volte, ma riduci anche i costi di un ordine di grandezza!

Porta il tuo contenitore () BYOC

Amazon Braket Hybrid Jobs offre tre contenitori predefiniti per l'esecuzione di codice in ambienti diversi. Se uno di questi contenitori supporta il tuo caso d'uso, devi fornire lo script dell'algoritmo solo quando crei un lavoro ibrido. Le dipendenze minori mancanti possono essere aggiunte dallo script dell'algoritmo o da un `requirements.txt` file utilizzando`pip`.

Se nessuno di questi contenitori supporta il tuo caso d'uso o se desideri ampliarli, Braket Hybrid Jobs supporta l'esecuzione di lavori ibridi personalizzati Docker immagine del contenitore o porta il tuo contenitore (BYOC). Ma prima di approfondire, assicuriamoci che sia effettivamente la funzionalità giusta per il tuo caso d'uso.

In questa sezione:

Quando portare il mio container è la decisione giusta?

Portare il proprio contenitore (BYOC) su Braket Hybrid Jobs offre la flessibilità di utilizzare il proprio software installandolo in un ambiente impacchettato. A seconda delle esigenze specifiche, potrebbero esserci modi per ottenere la stessa flessibilità senza dover ricorrere alla versione completa BYOC Docker build - Amazon ECR upload - URI ciclo di immagini personalizzato.

> **Note**
>
> BYOCpotrebbe non essere la scelta giusta se si desidera aggiungere un numero limitato di pacchetti Python aggiuntivi (generalmente meno di 10) che sono disponibili pubblicamente. Ad esempio, se stai usando PyPi.

In questo caso, puoi utilizzare una delle immagini Braket predefinite e quindi includere un `requirements.txt` file nella directory dei sorgenti al momento dell'invio del lavoro. Il file viene letto automaticamente e `pip` installerà i pacchetti con le versioni specificate normalmente. Se state installando un gran numero di pacchetti, la durata dei vostri job potrebbe aumentare notevolmente. Controlla Python e, se applicabile, la CUDA versione del contenitore precostruito che desideri utilizzare per verificare se il tuo software funzionerà.

BYOCè necessario quando si desidera utilizzare un linguaggio non Python (come C++ o Rust) per lo script di lavoro o se si desidera utilizzare una versione Python non disponibile tramite i contenitori predefiniti di Braket. È anche una buona scelta se:

- Stai utilizzando un software con una chiave di licenza e devi autenticarla con un server di licenza per eseguire il software. ConBYOC, puoi incorporare la chiave di licenza nel tuo Docker immagine e includi il codice per autenticarla.

- Stai utilizzando un software che non è disponibile pubblicamente. Ad esempio, il software è ospitato su un archivio privato GitLab o su un GitHub repository a cui è necessaria una SSH chiave particolare per accedere.
- È necessario installare un'ampia suite di software che non sia inclusa nei contenitori forniti da Braket. BYOCvi consentirà di eliminare i lunghi tempi di avvio per i contenitori di job ibridi dovuti all'installazione del software.

BYOCconsente inoltre di rendere disponibili ai clienti la propria personalizzazione SDK o il proprio algoritmo creando un Docker contenete il vostro software e mettetelo a disposizione dei vostri utenti. Puoi farlo impostando le autorizzazioni appropriate in AmazonECR.

> **Note**
>
> È necessario rispettare tutte le licenze software applicabili.

Ricetta per portare il proprio contenitore

In questa sezione, forniamo una step-by-step guida su ciò di cui avrai bisogno bring your own container (BYOC) a Braket Hybrid Jobs: gli script, i file e i passaggi per combinarli in modo da renderli operativi con i tuoi progetti personalizzati Docker immagini. Forniamo ricette per due casi comuni:

1. Installa software aggiuntivo in un Docker immagina e usa solo script di algoritmi Python nei tuoi lavori.
2. Usa script di algoritmi scritti in un linguaggio non Python con Hybrid Jobs o un'CPUarchitettura diversa da x86.

La definizione dello script di immissione del contenitore è più complessa nel caso 2.

Quando Braket esegue il tuo Hybrid Job, avvia il numero e il tipo richiesti di EC2 istanze Amazon, quindi esegue Docker immagine specificata dall'immagine URI inserita per la creazione di posti di lavoro su di esse. Quando utilizzi la BYOC funzionalità, specifichi un'immagine URI ospitata in un ECRrepository Amazon privato a cui hai accesso in lettura. Braket Hybrid Jobs utilizza quell'immagine personalizzata per eseguire il lavoro.

I componenti specifici necessari per creare un Docker immagine che può essere utilizzata con Hybrid Jobs. Se non hai familiarità con la scrittura e la creazione`Dockerfiles`, ti suggeriamo di fare

riferimento alla documentazione di Dockerfile e al Amazon ECR CLI la documentazione necessaria durante la lettura di queste istruzioni.

Ecco una panoramica di ciò di cui avrai bisogno:

- Un'immagine di base per il tuo Dockerfile
- (Facoltativo) Uno script modificato per il punto di ingresso del contenitore
- Installa il software e lo script del contenitore necessari con Dockerfile

Un'immagine di base per il tuo Dockerfile

Se utilizzi Python e desideri installare software in aggiunta a quanto fornito nei contenitori forniti da Braket, un'opzione per un'immagine di base è una delle immagini del contenitore Braket, ospitate nel nostro GitHub repository e su Amazon. ECR Dovrai autenticarti su Amazon ECR per estrarre l'immagine e crearla su di essa. Ad esempio, la prima riga del tuo BYOC Docker il file potrebbe essere: `FROM [IMAGE_URI_HERE]`

Quindi, compila il resto del Dockerfile per installare e configurare il software che desideri aggiungere al contenitore. Le immagini Braket predefinite conterranno già lo script del punto di ingresso del contenitore appropriato, quindi non devi preoccuparti di includerlo.

Se vuoi usare un linguaggio non Python, come C++, Rust o Julia, o se vuoi creare un'immagine per un'CPUarchitettura non x86, ad esempioARM, potresti dover creare su un'immagine pubblica scarna. Puoi trovare molte di queste immagini nella galleria pubblica di Amazon Elastic Container Registry. Assicurati di sceglierne una adatta all'CPUarchitettura e, se necessario, a quella che GPU desideri utilizzare.

(Facoltativo) Uno script modificato per il punto di ingresso del contenitore

> **Note**
>
> Se stai solo aggiungendo software aggiuntivo a un'immagine Braket predefinita, puoi saltare questa sezione.

Per eseguire codice non Python come parte del tuo lavoro ibrido, dovrai modificare lo script Python che definisce il punto di ingresso del contenitore. Ad esempio, lo script `braket_container.py` python su Amazon Braket Github. Questo è lo script utilizzato dalle immagini predefinite da Braket per avviare lo script dell'algoritmo e impostare le variabili di ambiente appropriate. Lo script del

punto di ingresso del contenitore stesso deve essere in Python, ma può avviare script non Python. Nell'esempio predefinito, puoi vedere che gli script degli algoritmi Python vengono avviati come sottoprocesso Python o come processo completamente nuovo. Modificando questa logica, è possibile abilitare lo script del punto di ingresso per avviare script di algoritmi non Python. Ad esempio, è possibile modificare la `thekick_off_customer_script()`funzione per avviare i processi Rust in base alla fine dell'estensione del file.

Puoi anche scegliere di scriverne uno completamente nuovo`braket_container.py`. Dovrebbe copiare i dati di input, gli archivi di origine e altri file necessari da Amazon S3 nel contenitore e definire le variabili di ambiente appropriate.

Installa il software e lo script del contenitore necessari con **`Dockerfile`**

> **Note**
>
> Se usi un'immagine Braket predefinita come Docker immagine di base, lo script del contenitore è già presente.

Se hai creato uno script del contenitore modificato nel passaggio precedente, dovrai copiarlo nel contenitore e definire la variabile di ambiente o il nome che hai dato `SAGEMAKER_PROGRAM` al `braket_container.py` nuovo script del punto di ingresso del contenitore.

Di seguito è riportato un esempio `Dockerfile` che consente di utilizzare Julia su istanze di Jobs GPU con accelerazione:

```
FROM nvidia/cuda:12.2.0-devel-ubuntu22.04


 ARG DEBIAN_FRONTEND=noninteractive
 ARG JULIA_RELEASE=1.8
 ARG JULIA_VERSION=1.8.3


 ARG PYTHON=python3.11
 ARG PYTHON_PIP=python3-pip
 ARG PIP=pip


 ARG JULIA_URL = https://julialang-s3.julialang.org/bin/linux/x64/${JULIA_RELEASE}/
 ARG TAR_NAME = julia-${JULIA_VERSION}-linux-x86_64.tar.gz
```

```
ARG PYTHON_PKGS = # list your Python packages and versions here


RUN curl -s -L ${JULIA_URL}/${TAR_NAME} | tar -C /usr/local -x -z --strip-components=1
-f -


RUN apt-get update \

   && apt-get install -y --no-install-recommends \

   build-essential \

   tzdata \

   openssh-client \

   openssh-server \

   ca-certificates \

   curl \

   git \

   libtemplate-perl \

   libssl1.1 \

   openssl \

   unzip \

   wget \

   zlib1g-dev \

   ${PYTHON_PIP} \

   ${PYTHON}-dev \
```

```
 RUN ${PIP} install --no-cache --upgrade ${PYTHON_PKGS}


 RUN ${PIP} install --no-cache --upgrade sagemaker-training==4.1.3


 # Add EFA and SMDDP to LD library path
 ENV LD_LIBRARY_PATH="/opt/conda/lib/python${PYTHON_SHORT_VERSION}/site-packages/
smdistributed/dataparallel/lib:$LD_LIBRARY_PATH"
 ENV LD_LIBRARY_PATH=/opt/amazon/efa/lib/:$LD_LIBRARY_PATH


 # Julia specific installation instructions
 COPY Project.toml /usr/local/share/julia/environments/v${JULIA_RELEASE}/
 RUN JULIA_DEPOT_PATH=/usr/local/share/julia \

    julia -e 'using Pkg; Pkg.instantiate(); Pkg.API.precompile()'
 # generate the device runtime library for all known and supported devices
 RUN JULIA_DEPOT_PATH=/usr/local/share/julia \

    julia -e 'using CUDA; CUDA.precompile_runtime()'


 # Open source compliance scripts
 RUN HOME_DIR=/root \

 && curl -o ${HOME_DIR}/oss_compliance.zip https://aws-dlinfra-
utilities.s3.amazonaws.com/oss_compliance.zip \

 && unzip ${HOME_DIR}/oss_compliance.zip -d ${HOME_DIR}/ \

 && cp ${HOME_DIR}/oss_compliance/test/testOSSCompliance /usr/local/bin/
testOSSCompliance \

 && chmod +x /usr/local/bin/testOSSCompliance \

 && chmod +x ${HOME_DIR}/oss_compliance/generate_oss_compliance.sh \

 && ${HOME_DIR}/oss_compliance/generate_oss_compliance.sh ${HOME_DIR} ${PYTHON} \

 && rm -rf ${HOME_DIR}/oss_compliance*
```

```
# Copying the container entry point script
COPY braket_container.py /opt/ml/code/braket_container.py
ENV SAGEMAKER_PROGRAM braket_container.py
```

Questo esempio scarica ed esegue gli script forniti da AWS per garantire la conformità con tutte le licenze Open-Source pertinenti. Ad esempio, attribuendo correttamente qualsiasi codice installato governato da un MIT license.

Se è necessario includere codice non pubblico, ad esempio codice ospitato in un GitLab archivio GitHub o in un archivio privato, non incorporare SSH chiavi in Docker immagine per accedervi. Invece, usa Docker Compose quando costruisci per consentire Docker per accedere alla macchina host SSH su cui è basato. Per ulteriori informazioni, consulta la guida Uso sicuro delle SSH chiavi in Docker per accedere ai repository privati di Github.

Creare e caricare il tuo Docker immagine

Una volta definito correttamente`Dockerfile`, ora sei pronto a seguire i passaggi per creare un ECR repository Amazon privato, se non ne esiste già uno. Puoi anche creare, etichettare e caricare l'immagine del contenitore nel repository.

Sei pronto per creare, etichettare e inserire l'immagine. Consulta la documentazione della build di Docker per una spiegazione completa delle opzioni `docker build` e alcuni esempi.

Per il file di esempio sopra definito, puoi eseguire:

```
aws ecr get-login-password --region ${your_region} | docker login --username AWS --
password-stdin ${aws_account_id}.dkr.ecr.${your_region}.amazonaws.com
 docker build -t braket-julia .
 docker tag braket-julia:latest ${aws_account_id}.dkr.ecr.${your_region}.amazonaws.com/
braket-julia:latest
 docker push ${aws_account_id}.dkr.ecr.${your_region}.amazonaws.com/braket-julia:latest
```

Assegnazione delle autorizzazioni Amazon ECR appropriate

Braket Hybrid Jobs Docker le immagini devono essere ospitate in ECR repository privati di Amazon. Per impostazione predefinita, un ECR repository Amazon privato non fornisce l'accesso in lettura a Braket Hybrid Jobs IAM role o a qualsiasi altro utente che desideri utilizzare la tua immagine, ad esempio un collaboratore o uno studente. È necessario impostare una politica di archiviazione per concedere le autorizzazioni appropriate. In generale, concedi l'autorizzazione solo a quegli

utenti specifici e IAM ruoli in cui desideri accedere alle tue immagini, anziché consentire a chiunque disponga del image URI per tirarli.

Lavori ibridi Running Braket nel tuo container

Per creare un lavoro ibrido con il tuo contenitore, chiama `AwsQuantumJob.create()` con l'argomento `image_uri` specificato. È possibile utilizzare un QPU simulatore su richiesta o eseguire il codice localmente sul processore classico disponibile con Braket Hybrid Jobs. Ti consigliamo di testare il codice su un simulatore come SV1DM1, o TN1 prima di eseguirlo su un simulatore reale. QPU

Per eseguire il codice sul processore classico, specifica l'`instanceType`and `instanceCount` che usi aggiornando il`InstanceConfig`. Tieni presente che se specifichi un valore `instance_count` > 1, devi assicurarti che il codice possa essere eseguito su più host. Il limite massimo per il numero di istanze che puoi scegliere è 5. Per esempio:

```
job = AwsQuantumJob.create(
    source_module="source_dir",
    entry_point="source_dir.algorithm_script:start_here",
    image_uri="111122223333.dkr.ecr.us-west-2.amazonaws.com/my-byoc-container:latest",
    instance_config=InstanceConfig(instanceType="ml.p3.8xlarge", instanceCount=3),
    device="local:braket/braket.local.qubit",
    # ...)
```

> **Note**
>
> Usa il dispositivo ARN per tracciare il simulatore che hai usato come metadati del lavoro ibrido. I valori accettabili devono seguire il formato. `device = "local:<provider>/<simulator_name>"` Ricordatelo `<provider>` e `<simulator_name>` deve essere composto solo da lettere, numeri`_`,`-`, e`.`. La stringa è limitata a 256 caratteri.
> Se intendi utilizzare Braket BYOC e non lo stai utilizzando SDK per creare attività quantistiche, dovresti passare il valore della variabile ambientale `AMZN_BRAKET_JOB_TOKEN` al `jobToken` parametro nella `CreateQuantumTask` richiesta. In caso contrario, le attività quantistiche non hanno la priorità e vengono fatturate come normali attività quantistiche autonome.

Configura il bucket predefinito in **`AwsSession`**

Fornendone uno personalizzato, si `AwsSession` ottiene una maggiore flessibilità, ad esempio, nella posizione del bucket predefinito. Per impostazione predefinita, an `AwsSession` ha una posizione predefinita del bucket di. `f"amazon-braket-{id}-{region}"` Ma puoi sovrascrivere tale impostazione predefinita quando crei un. `AwsSession` Gli utenti possono facoltativamente passare un `AwsSession` oggetto `AwsQuantumJob.create` con il nome del parametro, `aws_session` come illustrato nel seguente esempio di codice.

```
aws_session = AwsSession(default_bucket="other-default-bucket")

# then you can use that AwsSession when creating a hybrid job
job = AwsQuantumJob.create(
    ...
    aws_session=aws_session
)
```

Interagisci direttamente con i lavori ibridi utilizzando il API

Puoi accedere e interagire con Amazon Braket Hybrid Jobs direttamente utilizzando API. Tuttavia, le impostazioni predefinite e i metodi pratici non sono disponibili quando si utilizza API direttamente.

> **Note**
>
> Ti consigliamo vivamente di interagire con Amazon Braket Hybrid Jobs utilizzando Amazon Braket Python. SDK Offre comode impostazioni predefinite e protezioni che aiutano i processi ibridi a funzionare correttamente.

In questo argomento vengono illustrate le nozioni di base sull'utilizzo di API. Se scegli di utilizzare ilAPI, tieni presente che questo approccio può essere più complesso ed essere pronto a diverse iterazioni per eseguire il processo ibrido.

Per utilizzare ilAPI, il tuo account deve avere un ruolo nella policy `AmazonBraketFullAccess` gestita.

> **Note**
>
> Per ulteriori informazioni su come ottenere un ruolo con la policy `AmazonBraketFullAccess` gestita, consulta la pagina Abilita Amazon Braket.

Inoltre, è necessario un ruolo di esecuzione. Questo ruolo verrà passato al servizio. Puoi creare il ruolo utilizzando la console Amazon Braket. Utilizza la scheda Ruoli di esecuzione nella pagina Autorizzazioni e impostazioni per creare un ruolo predefinito per i lavori ibridi.

Il `CreateJob` API richiede di specificare tutti i parametri richiesti per il lavoro ibrido. Per usare Python, comprimi i file di script dell'algoritmo in un pacchetto tar, ad esempio un file input.tar.gz, ed esegui lo script seguente. Aggiorna le parti del codice tra parentesi angolate (<>) in modo che corrispondano alle informazioni dell'account e al punto di ingresso che specificano il percorso, il file e il metodo con cui inizia il processo ibrido.

```
from braket.aws import AwsDevice, AwsSession
import boto3
from datetime import datetime

s3_client = boto3.client("s3")
client = boto3.client("braket")

project_name = "job-test"
job_name = project_name + "-" + datetime.strftime(datetime.now(), "%Y%m%d%H%M%S")
bucket = "amazon-braket-<your_bucket>"
s3_prefix = job_name

job_script = "input.tar.gz"
job_object = f"{s3_prefix}/script/{job_script}"
s3_client.upload_file(job_script, bucket, job_object)

input_data = "inputdata.csv"
input_object = f"{s3_prefix}/input/{input_data}"
s3_client.upload_file(input_data, bucket, input_object)

job = client.create_job(
    jobName=job_name,
    roleArn="arn:aws:iam::<your_account>:role/service-role/
AmazonBraketJobsExecutionRole",  # https://docs.aws.amazon.com/braket/latest/
developerguide/braket-manage-access.html#about-amazonbraketjobsexecution
```

```
    algorithmSpecification={
        "scriptModeConfig": {
            "entryPoint": "<your_execution_module>:<your_execution_method>",
            "containerImage": {"uri": "292282985366.dkr.ecr.us-west-1.amazonaws.com/
amazon-braket-base-jobs:1.0-cpu-py37-ubuntu18.04"}   # Change to the specific region
 you are using
            "s3Uri": f"s3://{bucket}/{job_object}",
            "compressionType": "GZIP"
        }
    },
    inputDataConfig=[
        {
            "channelName": "hellothere",
            "compressionType": "NONE",
            "dataSource": {
                "s3DataSource": {
                    "s3Uri": f"s3://{bucket}/{s3_prefix}/input",
                    "s3DataType": "S3_PREFIX"
                }
            }
        }
    ],
    outputDataConfig={
        "s3Path": f"s3://{bucket}/{s3_prefix}/output"
    },
    instanceConfig={
        "instanceType": "ml.m5.large",
        "instanceCount": 1,
        "volumeSizeInGb": 1
    },
    checkpointConfig={
        "s3Uri":  f"s3://{bucket}/{s3_prefix}/checkpoints",
        "localPath": "/opt/omega/checkpoints"
    },
    deviceConfig={
        "priorityAccess": {
            "devices": [
                "arn:aws:braket:us-west-1::device/qpu/rigetti/Aspen-M-3"
            ]
        }
    },
    hyperParameters={
        "hyperparameter key you wish to pass": "<hyperparameter value you wish to
 pass>",
```

```
    },
    stoppingCondition={
        "maxRuntimeInSeconds": 1200,
        "maximumTaskLimit": 10
    },
)
```

Una volta creato il lavoro ibrido, puoi accedere ai dettagli del lavoro ibrido tramite `GetJob` API o la console. Per ottenere i dettagli del lavoro ibrido dalla sessione Python in cui hai eseguito il `createJob` codice come nell'esempio precedente, usa il seguente comando Python.

```
getJob = client.get_job(jobArn=job["jobArn"])
```

Per annullare un lavoro ibrido, chiamate il `CancelJob` API con Amazon Resource Name del lavoro ('JobArn').

```
cancelJob = client.cancel_job(jobArn=job["jobArn"])
```

È possibile specificare i checkpoint come parte di `createJob` API utilizzando il `checkpointConfig` parametro.

```
    checkpointConfig = {
        "localPath" : "/opt/omega/checkpoints",
        "s3Uri": f"s3://{bucket}/{s3_prefix}/checkpoints"
    },
```

> **Note**
>
> L' localPath of `checkpointConfig` non può iniziare con nessuno dei seguenti percorsi riservati: `/opt/ml/opt/braket`,`/tmp`, o`/usr/local/nvidia`.

Lavorare con le prenotazioni

Le prenotazioni ti danno accesso esclusivo al dispositivo quantistico di tua scelta. Puoi programmare una prenotazione quando preferisci, in modo da sapere esattamente quando inizia e termina l'esecuzione del carico di lavoro. Le prenotazioni sono disponibili con incrementi di 1 ora e possono essere cancellate fino a 48 ore di anticipo, senza costi aggiuntivi. Puoi scegliere di mettere in coda

in anticipo le attività quantistiche e i lavori ibridi per una prenotazione imminente o inviare carichi di lavoro durante la prenotazione.

Il costo dell'accesso dedicato ai dispositivi si basa sulla durata della prenotazione, indipendentemente dal numero di attività quantistiche e lavori ibridi eseguiti sulla Quantum Processing Unit (). QPU

I seguenti computer quantistici sono disponibili per le prenotazioni:

- Aria di IonQ
- IQMè Garnet
- QuEraè L'Aquila
- L'Aspen-M-3 di Rigetti

Quando utilizzare una prenotazione

L'utilizzo dell'accesso dedicato ai dispositivi per le prenotazioni offre la comodità e la prevedibilità di sapere esattamente quando il carico di lavoro quantistico inizia e termina l'esecuzione. Rispetto all'invio di attività e lavori ibridi su richiesta, non è necessario attendere in coda per le attività di altri clienti. Poiché hai accesso esclusivo al dispositivo durante la prenotazione, solo i tuoi carichi di lavoro verranno eseguiti sul dispositivo per l'intera prenotazione.

Ti consigliamo di utilizzare l'accesso su richiesta per la fase di progettazione e prototipazione della ricerca, per consentire un'iterazione rapida ed economica degli algoritmi. Quando siete pronti a produrre i risultati finali dell'esperimento, prendete in considerazione la possibilità di prenotare un dispositivo in base alle vostre esigenze per assicurarvi di rispettare le scadenze del progetto o della pubblicazione. Ti consigliamo inoltre di utilizzare le prenotazioni quando desideri eseguire le attività in orari specifici, ad esempio quando esegui una demo dal vivo o un workshop su un computer quantistico.

In questa sezione:

- Come creare una prenotazione
- Esecuzione di attività quantistiche durante una prenotazione
- Esecuzione di lavori ibridi durante una prenotazione
- Cosa succede alla fine della prenotazione
- Annulla o riprogramma una prenotazione esistente

Come creare una prenotazione

Per creare una prenotazione, contatta il team di Braket seguendo questi passaggi:

1. Apri la console Amazon Braket.
2. Scegli Braket Direct nel riquadro a sinistra, quindi nella sezione Prenotazioni, scegli Prenota dispositivo.
3. Seleziona il dispositivo che desideri prenotare.
4. Fornisci le tue informazioni di contatto tra cui nome ed e-mail. Assicurati di fornire un indirizzo email valido che controlli regolarmente.
5. Nella sezione Comunicaci il tuo carico di lavoro, fornisci tutti i dettagli sul carico di lavoro da eseguire utilizzando la tua prenotazione. Ad esempio, la durata della prenotazione desiderata, i vincoli pertinenti o la pianificazione desiderata.
6. Se sei interessato a contattare un esperto di Braket per una sessione di preparazione alla prenotazione dopo la conferma della prenotazione, seleziona facoltativamente Sono interessato a una sessione di preparazione.

Puoi anche contattarci per creare una prenotazione seguendo questi passaggi:

1. Apri la console Amazon Braket.
2. Scegli Dispositivi nel riquadro a sinistra e scegli il dispositivo che desideri prenotare.
3. Nella sezione Riepilogo, scegli Prenota dispositivo.
4. Segui i passaggi 4-6 della procedura precedente.

Dopo aver inviato il modulo, riceverai un'email dal team di Braket con i passaggi successivi per creare la tua prenotazione. Una volta confermata la prenotazione, riceverai la prenotazione ARN tramite e-mail.

> **Note**
>
> La prenotazione è confermata solo dopo la ricezione della prenotazioneARN.

Le prenotazioni sono disponibili con incrementi minimi di 1 ora e alcuni dispositivi potrebbero avere ulteriori vincoli di durata della prenotazione (inclusa la durata minima e massima della prenotazione). Il team di Braket condivide tutte le informazioni pertinenti con te prima di confermare la prenotazione.

Se hai indicato interesse a una sessione di preparazione alla prenotazione, il team di Braket ti contatterà tramite e-mail per organizzare una sessione di 30 minuti con un esperto di Braket.

Esecuzione di attività quantistiche durante una prenotazione

Dopo aver ottenuto una prenotazione valida ARN da Crea una prenotazione, puoi creare attività quantistiche da eseguire durante la prenotazione. Queste attività rimangono nello `QUEUED` stato fino all'inizio della prenotazione.

> **Note**
>
> Le prenotazioni sono AWS specifiche per account e dispositivo. Solo il AWS l'account che ha creato la prenotazione può utilizzare la prenotazioneARN.

> **Note**
>
> Non c'è visibilità in coda per le attività e i lavori inviati con una prenotazione ARN perché solo le attività vengono eseguite durante la prenotazione.

È possibile creare attività quantistiche utilizzando Python SDKs come Braket, Qiskit, PennyLaneo direttamente con boto3 (Funziona con Boto3). Per utilizzare le prenotazioni, devi disporre della versione v1.79.0 o successiva di Amazon Braket Python SDK. Puoi eseguire l'aggiornamento alla versione più recente di BraketSDK, Qiskit fornitore e PennyLane plugin con il seguente codice.

```
pip install --upgrade amazon-braket-sdk amazon-braket-pennylane-plugin qiskit-braket-provider
```

Esegui attività con il gestore di **`DirectReservation`** contesto

Il modo consigliato per eseguire un'attività all'interno della prenotazione pianificata consiste nell'utilizzare il gestore di `DirectReservation` contesto. Specificando il dispositivo di destinazione e la prenotazioneARN, il gestore di contesto assicura che tutte le attività create all'interno dell'istruzione `with` Python vengano eseguite con accesso esclusivo al dispositivo.

Innanzitutto, definisci un circuito quantistico e il dispositivo. Quindi utilizza il contesto di prenotazione ed esegui l'attività.

```
from braket.aws import AwsDevice, DirectReservation
from braket.circuits import Circuit
from braket.devices import Devices

bell = Circuit().h(0).cnot(0, 1)
device = AwsDevice(Devices.IonQ.Aria1)

# run the circuit in a reservation
with DirectReservation(device, reservation_arn="<my_reservation_arn>"):
    task = device.run(bell, shots=100)
```

È possibile creare attività quantistiche in una prenotazione utilizzando PennyLane e Qiskit plugin, purché il `DirectReservation` contesto sia attivo durante la creazione di attività quantistiche. Ad esempio, con Qiskit-Braket provider, è possibile eseguire le attività nel modo seguente.

```
from braket.devices import Devices
from braket.aws import DirectReservation
from qiskit import QuantumCircuit
from qiskit_braket_provider import BraketProvider


qc = QuantumCircuit(2)
qc.h(0)
qc.cx(0, 1)

aria = BraketProvider().get_backend("Aria 1")

# run the circuit in a reservation
with DirectReservation(Devices.IonQ.Aria1, reservation_arn="<my_reservation_arn>"):
    aria_task = aria.run(qc, shots=10)
```

Analogamente, il codice seguente esegue un circuito durante una prenotazione utilizzando il Braket-PennyLane .

```
from braket.devices import Devices
from braket.aws import DirectReservation
import pennylane as qml


dev = qml.device("braket.aws.qubit", device_arn=Devices.IonQ.Aria1.value, wires=2,
 shots=10)
```

```
@qml.qnode(dev)
def bell_state():
    qml.Hadamard(wires=0)
    qml.CNOT(wires=[0, 1])
    return qml.probs(wires=[0, 1])

# run the circuit in a reservation
with DirectReservation(Devices.IonQ.Aria1, reservation_arn="<my_reservation_arn>"):
    probs = bell_state()
```

Impostazione manuale del contesto di prenotazione

In alternativa, puoi impostare manualmente il contesto di prenotazione con il seguente codice.

```
# set reservation context
reservation = DirectReservation(device, reservation_arn="<my_reservation_arn>").start()

# run circuit during reservation
task = device.run(bell, shots=100)
```

Questo è ideale per i notebook Jupyter in cui il contesto può essere eseguito nella prima cella e tutte le attività successive verranno eseguite nella prenotazione.

> **Note**
>
> La cella contenente la *`.start()`* chiamata deve essere eseguita solo una volta.

Per tornare alla modalità su richiesta: riavvia il notebook Jupyter o chiama quanto segue per riportare il contesto alla modalità su richiesta.

```
reservation.stop()  # unset reservation context
```

> **Note**
>
> Le prenotazioni hanno un orario di inizio e di fine programmato (vedi Creare una prenotazione). I `reservation.stop()` metodi `reservation.start()` and non iniziano o terminano una prenotazione. Si tratta di metodi per modificare tutte le attività quantistiche successive da eseguire durante la prenotazione. Questi metodi non hanno alcun effetto sull'orario di prenotazione programmato.

Passa esplicitamente la prenotazione ARN durante la creazione dell'attività

Un altro modo per creare attività durante una prenotazione consiste nell'inoltrare esplicitamente la prenotazione al ARN momento della chiamata. `device.run()`

```
task = device.run(bell, shots=100, reservation_arn="<my_reservation_arn>")
```

Questo metodo associa direttamente l'attività quantistica alla prenotazioneARN, assicurando che venga eseguita durante il periodo prenotato. Per questa opzione, aggiungi la prenotazione ARN a ogni attività che intendi eseguire durante una prenotazione. Inoltre, controlla che le attività siano state create in Qiskit oppure PennyLane stanno utilizzando la prenotazione correttaARN. A causa di queste considerazioni aggiuntive, si consigliano i due metodi precedenti.

Quando usi direttamente boto3, trasmetti la prenotazione ARN come associazione durante la creazione di un'attività.

```
import boto3

braket_client = boto3.client("braket")


kwargs["associations"] = [
    {
        "arn": "<my_reservation_arn>",
        "type": "RESERVATION_TIME_WINDOW_ARN",
    }
]

response = braket_client.create_quantum_task(**kwargs)
```

Esecuzione di lavori ibridi durante una prenotazione

Una volta che hai una funzione Python da eseguire come lavoro ibrido, puoi eseguire il lavoro ibrido in una prenotazione passando l'argomento della `reservation_arn` parola chiave. Tutte le attività all'interno del lavoro ibrido utilizzano la prenotazioneARN. È importante sottolineare che il lavoro ibrido attiva il calcolo classico *`reservation_arn`* solo dopo l'inizio della prenotazione.

> **Note**
>
> Un processo ibrido in esecuzione durante una prenotazione esegue correttamente solo attività quantistiche sul dispositivo riservato. Il tentativo di utilizzare un dispositivo Braket

on-demand diverso genererà un errore. Se devi eseguire attività sia su un simulatore su richiesta che sul dispositivo riservato all'interno dello stesso lavoro ibrido, utilizza invece. `DirectReservation`

Il codice seguente per eseguire un processo ibrido durante una prenotazione.

```
from braket.aws import AwsDevice
from braket.devices import Devices
from braket.jobs import get_job_device_arn, hybrid_job

@hybrid_job(device=Devices.IonQ.Aria1, reservation_arn="<my_reservation_arn>")
def example_hybrid_job():
    # declare AwsDevice within the hybrid job
    device = AwsDevice(get_job_device_arn())
    bell = Circuit().h(0).cnot(0, 1)

    task = device.run(bell, shots=10)
```

Per i lavori ibridi che utilizzano uno script Python (vedi Create your first Hybrid Job), puoi eseguirli nella prenotazione passando l'argomento della parola chiave `reservation_arn`

```
from braket.aws import AwsQuantumJob
from braket.devices import Devices

job = AwsQuantumJob.create(
    Devices.IonQ.Aria1,
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    reservation_arn="<my_reservation_arn>"
)
```

Cosa succede alla fine della prenotazione

Al termine della prenotazione, non avrai più un accesso dedicato al dispositivo. Tutti i carichi di lavoro rimanenti in coda con questa prenotazione vengono automaticamente annullati.

> **Note**
>
> Qualsiasi lavoro che era in corso al termine della `RUNNING` prenotazione viene annullato. Ti consigliamo di utilizzare i checkpoint per salvare e riavviare i lavori a tuo piacimento.

Una prenotazione in corso, ad esempio dopo l'inizio della prenotazione e prima della fine della prenotazione, non può essere estesa perché ogni prenotazione rappresenta l'accesso indipendente a un dispositivo dedicato. Ad esempio, due back-to-back prenotazioni sono considerate separate e tutte le attività in sospeso dalla prima prenotazione vengono automaticamente annullate. Non riprendono nella seconda prenotazione.

> **Note**
>
> Le prenotazioni rappresentano l'accesso al dispositivo dedicato per i tuoi AWS conto. Anche se il dispositivo rimane inattivo, nessun altro cliente può utilizzarlo. Pertanto, ti verrà addebitata la durata del tempo prenotato, indipendentemente dal tempo utilizzato.

Annulla o riprogramma una prenotazione esistente

Puoi cancellare la tua prenotazione non meno di 48 ore prima dell'orario di inizio previsto per la prenotazione. Per annullare, rispondi all'e-mail di conferma della prenotazione che hai ricevuto con la richiesta di cancellazione.

Per riprogrammare, devi cancellare la prenotazione esistente e quindi crearne una nuova.

Tecniche di mitigazione degli errori

La mitigazione quantistica degli errori è un insieme di tecniche volte a ridurre gli effetti degli errori nei computer quantistici.

I dispositivi quantistici sono soggetti a rumori ambientali che riducono la qualità dei calcoli eseguiti. Sebbene il calcolo quantistico tollerante ai guasti prometta una soluzione a questo problema, gli attuali dispositivi quantistici sono limitati dal numero di qubit e dai tassi di errore relativamente elevati. Per ovviare a questo problema nel breve termine, i ricercatori stanno studiando metodi per migliorare l'accuratezza dei rumorosi calcoli quantistici. Questo approccio, noto come mitigazione degli errori

quantistici, prevede l'utilizzo di varie tecniche per estrarre il segnale migliore da dati di misurazione rumorosi.

In questa sezione:

- Tecniche di mitigazione degli errori su IonQ Aria

Tecniche di mitigazione degli errori su IonQ Aria

La mitigazione degli errori implica l'esecuzione di più circuiti fisici e la combinazione delle relative misurazioni per ottenere un risultato migliore. Il IonQ Aria il dispositivo presenta un metodo di mitigazione degli errori chiamato debiasing.

Debiasing mappa un circuito in più varianti che agiscono su diverse permutazioni di qubit o con diverse decomposizioni dei gate. Ciò riduce l'effetto di errori sistematici come le sovratratazioni del gate o un singolo qubit difettoso utilizzando diverse implementazioni di un circuito che altrimenti potrebbero alterare i risultati di misurazione. Ciò comporta costi aggiuntivi per la calibrazione di più qubit e gate.

Per ulteriori informazioni sul debiasing, vedere Miglioramento delle prestazioni dei computer quantistici attraverso la simmetrizzazione.

> **Note**
>
> L'uso del debiasing richiede un minimo di 2500 scatti.

È possibile eseguire un'operazione quantistica con il debiasing su un IoQ Aria dispositivo che utilizza il seguente codice:

```
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.error_mitigation import Debias

device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")
circuit = Circuit().h(0).cnot(0, 1)

task = device.run(circuit, shots=2500, device_parameters={"errorMitigation": Debias()})

result = task.result()
```

```
print(result.measurement_counts)
>>> {"00": 1245, "01": 5, "10": 10 "11": 1240} # result from debiasing
```

Una volta completata l'attività quantistica, è possibile visualizzare le probabilità di misurazione e tutti i tipi di risultato dell'operazione quantistica. Le probabilità di misurazione e i conteggi di tutte le varianti vengono aggregati in un'unica distribuzione. Qualsiasi tipo di risultato specificato nel circuito, ad esempio i valori attesi, viene calcolato utilizzando i conteggi delle misurazioni aggregate.

Filtro di nitidezza

È inoltre possibile accedere alle probabilità di misurazione calcolate con una diversa strategia di post-elaborazione chiamata nitidezza. La nitidezza confronta i risultati di ciascuna variante ed elimina le immagini incoerenti, favorendo il risultato di misurazione più probabile tra le varianti. Per ulteriori informazioni, vedete Migliorare le prestazioni dei computer quantistici attraverso la simmetrizzazione.

È importante sottolineare che la nitidezza presuppone che la forma della distribuzione di output sia scarsa, con pochi stati ad alta probabilità e molti stati a probabilità zero. Se questa ipotesi non è valida, può distorcere la distribuzione delle probabilità.

Puoi accedere alle probabilità da una distribuzione più nitida nel `additional_metadata` campo su Braket `GateModelTaskResult` Python. SDK Nota che la nitidezza non restituisce i conteggi delle misurazioni, ma restituisce invece una distribuzione di probabilità rinormalizzata. Il seguente frammento di codice mostra come accedere alla distribuzione dopo la nitidezza.

```
print(result.additional_metadata.ionqMetadata.sharpenedProbabilities)
>>> {"00": 0.51, "11": 0.549} # sharpened probabilities
```

Risoluzione dei problemi con Amazon Braket

Utilizza le informazioni e le soluzioni per la risoluzione dei problemi in questa sezione per risolvere i problemi con Amazon Braket.

In questa sezione:

AccessDeniedException

Se ricevi un messaggio AccessDeniedExceptionquando attivi o utilizzi Braket, è probabile che tu stia tentando di abilitare o utilizzare Braket in una regione in cui il tuo ruolo limitato non ha accesso.

In questi casi, devi contattare il tuo interno AWS amministratore per capire quale delle seguenti condizioni si applica:

- Se esistono restrizioni di ruolo che impediscono l'accesso a una regione.
- Se il ruolo che stai tentando di utilizzare è autorizzato, usa Braket.

Se il tuo ruolo non ha accesso a una determinata regione quando usi Braket, non potrai utilizzare i dispositivi in quella particolare regione.

Si è verificato un errore (ValidationException) durante la chiamata dell'operazione CreateQuantumTask

Se ricevi un errore simile a: `An error occurred (ValidationException) when calling the CreateQuantumTask operation: Caller doesn’t have access to amazon-`

`braket-…` Verifica di fare riferimento a una s3_folder esistente. Braket non crea automaticamente nuovi bucket e prefissi Amazon S3 per te.

Se stai accedendo a API direttamente e riceve un errore simile a: `Failed to create quantum task: Caller doesn't have access to s3://MY_BUCKET` Verifica di non includere `s3://` nel percorso del bucket Amazon S3.

Una SDK funzionalità non funziona

La tua versione di Python deve essere 3.9 o superiore. Per Amazon Braket Hybrid Jobs, consigliamo Python 3.10.

Verifica che i tuoi schemi SDK e lo siano. up-to-date Per aggiornarlo SDK dal notebook o dall'editor Python, esegui il seguente comando:

```
pip install amazon-braket-sdk --upgrade --upgrade-strategy eager
```

Per aggiornare gli schemi, esegui il seguente comando:

```
pip install amazon-braket-schemas --upgrade
```

Se accedi ad Amazon Braket dal tuo cliente, verifica AWSLa regione è impostata su una regione supportata da Amazon Braket.

Il processo ibrido fallisce a causa di ServiceQuotaExceededException

Un job ibrido che esegue attività quantistiche con i simulatori Amazon Braket può non essere creato se si supera il limite di attività quantistiche simultanee per il dispositivo di simulazione a cui si sta puntando. Per ulteriori informazioni sui limiti del servizio, consulta l'argomento Quotas.

Se esegui attività simultanee su un dispositivo di simulazione in più lavori ibridi dal tuo account, potresti riscontrare questo errore.

Per visualizzare il numero di attività quantistiche simultanee rispetto a uno specifico dispositivo di simulazione, usa il `search-quantum-tasks` API, come illustrato nel seguente esempio di codice.

```
DEVICE_ARN=arn:aws:braket:::device/quantum-simulator/amazon/sv1
```

```
task_list=""
for status_value in "CREATED" "QUEUED" "RUNNING" "CANCELLING"; do
    tasks=$(aws braket search-quantum-tasks --filters
 name=status,operator=EQUAL,values=${status_value}
 name=deviceArn,operator=EQUAL,values=$DEVICE_ARN --max-results 100 --query
 'quantumTasks[*].quantumTaskArn' --output text)
    task_list="$task_list $tasks"
done;
echo "$task_list" | tr -s ' \t' '[\n*]' | sort | uniq
```

Puoi anche visualizzare le attività quantistiche create su un dispositivo utilizzando i CloudWatch parametri di Amazon: Braket > By Device.

Per evitare di incorrere in questi errori:

1. Richiedete un aumento della quota di servizio per il numero di attività quantistiche simultanee per il dispositivo di simulazione. Questo è applicabile solo a SV1 dispositivo.
2. Gestisci `ServiceQuotaExceeded` le eccezioni nel codice e riprova.

I componenti hanno smesso di funzionare nell'istanza del notebook

Se alcuni componenti del notebook smettono di funzionare, prova quanto segue:

1. Scarica tutti i notebook che hai creato o modificato su un'unità locale.
2. Interrompi l'istanza del tuo notebook.
3. Elimina l'istanza del tuo notebook.
4. Crea una nuova istanza del notebook con un nome diverso.
5. Carica i taccuini nella nuova istanza.

Risoluzione dei problemi Open QASM

Questa sezione fornisce suggerimenti per la risoluzione dei problemi che potrebbero essere utili in caso di errori utilizzando Open 3.0. QASM

In questa sezione:

- Miscelazione fisica qubits con virtuale qubits error
- Richiesta dei tipi di risultati e delle misurazioni qubits nello stesso errore del programma
- Classico e qubit limite di registro superato (errore)
- Casella non preceduta da un errore pragmatico letterale
- Errore con porte native mancanti nelle caselle Verbatim
- Le scatole Verbatim sono mancanti qubits error
- Nel pragma letterale manca l'errore «braket»
- Singolo qubits non può essere un errore indicizzato
- Il fisico qubits in due qubit errore di mancata connessione del cancello
- Avviso di supporto al simulatore locale

Errore di inclusione dell'istruzione

Braket attualmente non dispone di un file di libreria gate standard da includere nei programmi Open. QASM Ad esempio, l'esempio seguente genera un errore del parser.

```
OPENQASM 3;
include "standardlib.inc";
```

Questo codice genera il messaggio di errore: `No terminal matches '"' in the current parser context, at line 2 col 17.`

Non contiguo qubits error

Utilizzo di elementi non contigui qubits sui dispositivi che sono `requiresContiguousQubitIndices` impostati sulla funzionalità `true` del dispositivo generano un errore.

Quando si eseguono attività quantistiche su simulatori e IonQ, il seguente programma attiva l'errore.

```
OPENQASM 3;

qubit[4] q;

h q[0];
cnot q[0], q[2];
```

```
cnot q[0], q[3];
```

Questo codice genera il messaggio di errore: `Device requires contiguous qubits. Qubit register q has unused qubits q[1], q[4].`

Miscelazione fisica qubits con virtuale qubits error

Miscelazione fisica qubits con virtuale qubits nello stesso programma non è consentito e genera un errore. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit[2] q;
cnot q[0], $1;
```

Questo codice genera il messaggio di errore: `[line 4] mixes physical qubits and qubits registers.`

Richiesta dei tipi di risultati e delle misurazioni qubits nello stesso errore del programma

Richiedere tipi di risultati e altro qubits vengono misurati in modo esplicito nello stesso programma generano un errore. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit[2] q;

h q[0];
cnot q[0], q[1];
measure q;

#pragma braket result expectation x(q[0]) @ z(q[1])
```

Questo codice genera il messaggio di errore: `Qubits should not be explicitly measured when result types are requested.`

Classico e qubit limite di registro superato (errore)

Solo un registro classico e uno qubit sono consentiti i registri. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit[2] q0;
qubit[2] q1;
```

Questo codice genera il messaggio di errore: `[line 4] cannot declare a qubit register. Only 1 qubit register is supported.`

Casella non preceduta da un errore pragmatico letterale

Tutte le caselle devono essere precedute da un pragma letterale. Il codice seguente genera l'errore.

```
box{
rx(0.5) $0;
}
```

Questo codice genera il messaggio di errore: `In verbatim boxes, native gates are required. x is not a device native gate.`

Errore con porte native mancanti nelle caselle Verbatim

Le scatole Verbatim devono avere porte native e fisiche qubits. Il codice seguente genera l'errore native gates.

```
#pragma braket verbatim
box{
x $0;
}
```

Questo codice genera il messaggio di errore: `In verbatim boxes, native gates are required. x is not a device native gate.`

Le scatole Verbatim sono mancanti qubits error

Le scatole Verbatim devono essere fisiche qubits. Il codice seguente genera il file fisico mancante qubits errore.

```
qubit[2] q;
```

```
#pragma braket verbatim
box{
rx(0.1) q[0];
}
```

Questo codice genera il messaggio di errore: `Physical qubits are required in verbatim box.`

Nel pragma letterale manca l'errore «braket»

Devi includere «parentesi» nel pragma letterale. Il codice seguente genera l'errore.

```
#pragma braket verbatim          // Correct
#pragma verbatim                 // wrong
```

Questo codice genera il messaggio di errore: `You must include “braket” in the verbatim pragma`

Singolo qubits non può essere un errore indicizzato

Singolo qubits non può essere indicizzato. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit q;
h q[0];
```

Questo codice genera l'errore: `[line 4] single qubit cannot be indexed.`

Tuttavia, singolo qubit gli array possono essere indicizzati come segue:

```
OPENQASM 3;

qubit[1] q;
h q[0];   // This is valid
```

Il fisico qubits in due qubit errore di mancata connessione del cancello

Da usare fisico qubits, verifica innanzitutto che il dispositivo utilizzi dispositivi fisici qubits controllando `device.properties.action[DeviceActionType.OPENQASM].supportPhysicalQubits` e

quindi verificando il grafico di connettività selezionando `device.properties.paradigm.connectivity.connectivityGraph` o`device.properties.paradigm.connectivity.fullyConnected`.

```
OPENQASM 3;

cnot $0, $14;
```

Questo codice genera il messaggio di errore: `[line 3] has disconnected qubits 0 and 14`

Avviso di supporto al simulatore locale

`LocalSimulator`Supporta funzionalità avanzate in Open QASM che potrebbero non essere disponibili nei simulatori QPUs o su richiesta. Se il programma contiene funzionalità linguistiche specifiche solo per`LocalSimulator`, come illustrato nell'esempio seguente, verrà visualizzato un avviso.

```
qasm_string = """
qubit[2] q;

h q[0];
ctrl @ x q[0], q[1];
"""
qasm_program = Program(source=qasm_string)
```

Questo codice genera l'avviso: `Questo programma utilizza le funzionalità del QASM linguaggio Open supportate solo in. LocalSimulator Alcune di queste funzionalità potrebbero non essere supportate nei simulatori QPUs o su richiesta.

Per ulteriori informazioni sulle QASM funzionalità Open supportate, fai clic qui.

Sicurezza in Amazon Braket

Questo capitolo ti aiuta a capire come applicare il modello di responsabilità condivisa quando usi Amazon Braket. Ti mostra come configurare Amazon Braket per soddisfare i tuoi obiettivi di sicurezza e conformità. Imparerai anche a usarne altri Servizi AWS che ti aiutano a monitorare e proteggere le tue risorse Amazon Braket.

Sicurezza nel cloud presso AWS è la massima priorità. Come un AWS cliente, trarrai vantaggio da un'architettura di data center e rete progettata per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza. L'utente è responsabile di altri fattori, tra cui la sensibilità dei dati, i requisiti aziendali e le leggi e i regolamenti applicabili.

In questa sezione:

Responsabilità condivisa per la sicurezza

La sicurezza è una responsabilità condivisa tra AWS e tu. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud — AWS è responsabile della protezione dell'infrastruttura in esecuzione Servizi AWS nel Cloud AWS. AWS offre inoltre servizi che è possibile utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito del AWS Programmi di conformità. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon Braket, consulta AWS Servizi rientranti nell'ambito del programma di conformità.

- Sicurezza nel cloud: è tua responsabilità mantenere il controllo sui contenuti ospitati su questo sito AWS infrastruttura. Questo contenuto include le attività di configurazione e gestione della sicurezza per Servizi AWS che usi.

Protezione dei dati

Il AWS modello di responsabilità condivisa modello si applica alla protezione dei dati in Amazon Braket. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutte le Cloud AWS. L'utente è responsabile del mantenimento del controllo sui contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile delle attività di configurazione e gestione della sicurezza per Servizi AWS che usi. Per ulteriori informazioni sulla privacy dei dati, consulta la sezione Privacy dei dati FAQ. Per informazioni sulla protezione dei dati in Europa, consultare il AWS Modello di responsabilità condivisa e post sul GDPR blog sul AWS Blog sulla sicurezza.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS credenziali e configura singoli utenti con AWS IAM Identity Center oppure AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Usa l'autenticazione a più fattori (MFA) con ogni account.
- UsaSSL/TLSper comunicare con AWS risorse. Richiediamo TLS 1.2 e consigliamo TLS 1.3.
- Configurazione API e registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi di acquisizione AWS attività, vedi Lavorare con i CloudTrail sentieri in AWS CloudTrail Guida per l'utente.
- Utilizzo AWS soluzioni di crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se sono necessari FIPS 140-3 moduli crittografici convalidati per l'accesso AWS tramite un'interfaccia a riga di comando o unAPI, utilizza un endpoint. FIPS Per ulteriori informazioni sugli FIPS endpoint disponibili, vedere Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Amazon Braket o altro Servizi AWS utilizzando la console, API AWS

CLI, oppure AWS SDKs. I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Se fornisci un URL a un server esterno, ti consigliamo vivamente di non includere le informazioni sulle credenziali URL per convalidare la tua richiesta a quel server.

Conservazione dei dati

Dopo 90 giorni, Amazon Braket rimuove automaticamente tutte le attività quantistiche IDs e gli altri metadati associati alle tue attività quantistiche. Come risultato di questa politica di conservazione dei dati, queste attività e risultati non sono più recuperabili tramite ricerca dalla console Amazon Braket, sebbene rimangano archiviati nel bucket S3.

Se hai bisogno di accedere alle attività e ai risultati quantistici storici archiviati nel tuo bucket S3 per più di 90 giorni, devi tenere un registro separato dell'ID dell'attività e degli altri metadati associati a tali dati. Assicurati di salvare le informazioni prima di 90 giorni. Puoi utilizzare le informazioni salvate per recuperare i dati storici.

Gestione dell'accesso ad Amazon Braket

Questo capitolo descrive le autorizzazioni necessarie per l'esecuzione Amazon Braket, o per limitare l'accesso di utenti e ruoli specifici. Puoi concedere (o negare) le autorizzazioni richieste a qualsiasi utente o ruolo del tuo account. A tale scopo, allega la policy Amazon Braket appropriata a quell'utente o ruolo nel tuo account, come descritto nelle sezioni seguenti.

Come prerequisito, devi abilitare Amazon Braket. Per abilitare Braket, assicurati di accedere come utente o ruolo con (1) autorizzazioni di amministratore o (2) che disponga della AmazonBraketFullAccesspolicy e delle autorizzazioni per creare bucket Amazon Simple Storage Service (Amazon S3).

In questa sezione:

- Risorse Amazon Braket
- Taccuini e ruoli
- Informazioni sulla AmazonBraketFullAccess politica
- Informazioni sulla AmazonBraketJobsExecutionPolicy politica
- Limita l'accesso degli utenti a determinati dispositivi
- Amazon Braket si aggiorna a AWS policy gestite

- Limita l'accesso degli utenti a determinate istanze del notebook
- Limita l'accesso degli utenti a determinati bucket S3

Risorse Amazon Braket

Braket crea un tipo di risorsa: la risorsa quantum-task. L'Amazon Resource Name (ARN) per questo tipo di risorsa è il seguente:

- Nome della risorsa: AWS::Service: :Braket
- ARNRegex: arn: $ {Partition} :braket: $ {Region} :$ {Account} :quantum-task/$ {} RandomId

Taccuini e ruoli

Puoi usare il tipo di risorsa notebook in Braket. Un notebook è una SageMaker risorsa Amazon che Braket è in grado di condividere. Per utilizzare un notebook con Braket, devi specificare un IAM ruolo con un nome che inizi con. `AmazonBraketServiceSageMakerNotebook`

Per creare un taccuino, è necessario utilizzare un ruolo con autorizzazioni di amministratore o a cui sia associata la seguente politica in linea.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": "iam:CreateRole",
           "Resource": "arn:aws:iam::*:role/service-role/
AmazonBraketServiceSageMakerNotebookRole*"
       },
       {
           "Effect": "Allow",
           "Action": "iam:CreatePolicy",
           "Resource": [
               "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookAccess*",
               "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookRole*"
           ]
       },
       {
```

```
            "Effect": "Allow",
            "Action": "iam:AttachRolePolicy",
            "Resource": "arn:aws:iam::*:role/service-role/
AmazonBraketServiceSageMakerNotebookRole*",
            "Condition": {
                "StringLike": {
                    "iam:PolicyARN": [
                        "arn:aws:iam::aws:policy/AmazonBraketFullAccess",
                        "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookAccess*",
                        "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookRole*"
                    ]
                }
            }
        }
    ]
}
```

Per creare il ruolo, segui i passaggi indicati nella pagina Crea un taccuino o chiedi all'amministratore di crearlo per te. Assicurati che la AmazonBraketFullAccesspolicy sia allegata.

Dopo aver creato il ruolo, puoi riutilizzarlo per tutti i notebook che lancerai in futuro.

Informazioni sulla AmazonBraketFullAccess politica

La AmazonBraketFullAccesspolicy concede le autorizzazioni per le operazioni di Amazon Braket, incluse le autorizzazioni per le seguenti attività:

- Scarica contenitori da Amazon Elastic Container Registry: per leggere e scaricare immagini di contenitori utilizzate per Amazon Funzione Braket Hybrid Jobs. I contenitori devono essere conformi al formato «arn:aws:ecr: :repository/amazon-braket».
- Mantieni AWS CloudTrail log: per tutte le azioni di descrizione, acquisizione ed elenco, oltre all'avvio e all'interruzione delle query, al test dei filtri delle metriche e al filtraggio degli eventi di registro. Il AWS CloudTrail il file di registro contiene un record di tutti gli Amazon Braket API attività che si verifica nel tuo account.
- Utilizza i ruoli per controllare le risorse: per creare un ruolo collegato al servizio nel tuo account. Il ruolo collegato al servizio ha accesso a AWS risorse per tuo conto. Può essere utilizzato solo dal servizio Amazon Braket. Inoltre, per passare i IAM ruoli ad Amazon Braket `CreateJob` API e per creare un ruolo e allegare una policy specifica AmazonBraketFullAccess al ruolo.

- Crea gruppi di log, eventi di log e gruppi di log di query per gestire i file di log di utilizzo per il tuo account: per creare, archiviare e visualizzare informazioni di registrazione sull'utilizzo di Amazon Braket nel tuo account. Interroga le metriche sui gruppi di log dei lavori ibridi. Comprende il percorso Braket corretto e consenti l'inserimento dei dati di registro. Inserisci i dati metrici. CloudWatch
- Crea e archivia dati nei bucket Amazon S3 ed elenca tutti i bucket — Per creare bucket S3, elenca i bucket S3 nel tuo account e inserisci oggetti e ottieni oggetti da qualsiasi bucket del tuo account il cui nome inizia con amazon-braket-. Queste autorizzazioni sono necessarie per consentire a Braket di inserire i file contenenti i risultati di attività quantistiche elaborate nel bucket e di recuperarli dal bucket.
- Passa IAM ruoli: per trasferire i ruoli a IAM `CreateJob` API.
- Amazon SageMaker Notebook: per creare e gestire SageMaker istanze di notebook riconducibili alla risorsa «arn:aws:sagemaker: ::notebook-instance/amazon-braket-».
- Convalida delle quote di servizio: per creare SageMaker notebook e lavori Amazon Braket Hybrid, il numero di risorse non può superare le quote del tuo account.

Contenuto della politica

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock",
                "s3:PutBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::amazon-braket-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "servicequotas:GetServiceQuota",
                "cloudwatch:GetMetricData"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": "arn:aws:ecr:*:*:repository/amazon-braket*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:Describe*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/braket*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "sagemaker:ListNotebookInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedNotebookInstanceUrl",
                "sagemaker:CreateNotebookInstance",
                "sagemaker:DeleteNotebookInstance",
                "sagemaker:DescribeNotebookInstance",
                "sagemaker:StartNotebookInstance",
                "sagemaker:StopNotebookInstance",
                "sagemaker:UpdateNotebookInstance",
                "sagemaker:ListTags",
                "sagemaker:AddTags",
                "sagemaker:DeleteTags"
            ],
            "Resource": "arn:aws:sagemaker:*:*:notebook-instance/amazon-braket-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:DescribeNotebookInstanceLifecycleConfig",
                "sagemaker:CreateNotebookInstanceLifecycleConfig",
                "sagemaker:DeleteNotebookInstanceLifecycleConfig",
                "sagemaker:ListNotebookInstanceLifecycleConfigs",
                "sagemaker:UpdateNotebookInstanceLifecycleConfig"
            ],
            "Resource": "arn:aws:sagemaker:*:*:notebook-instance-lifecycle-config/
amazon-braket-*"
        },
        {
            "Effect": "Allow",
            "Action": "braket:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/braket.amazonaws.com/
AWSServiceRoleForAmazonBraket*",
```

```
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "braket.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/service-role/
AmazonBraketServiceSageMakerNotebookRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "sagemaker.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/service-role/
AmazonBraketJobsExecutionRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "braket.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetQueryResults"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/braket*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "/aws/braket"
                }
            }
        }
    ]
}
```

Informazioni sulla AmazonBraketJobsExecutionPolicy politica

La AmazonBraketJobsExecutionPolicypolicy concede le autorizzazioni per i ruoli di esecuzione utilizzati in Amazon Braket Hybrid Jobs come segue:

- Scarica contenitori da Amazon Elastic Container Registry - Autorizzazioni per leggere e scaricare immagini di container utilizzate per la funzionalità Amazon Braket Hybrid Jobs. I contenitori devono essere conformi al formato «arn:aws:ecr: *:*:repository/amazon-braket*».
- Crea gruppi di log ed eventi di log e interroga gruppi di log per gestire i file di log di utilizzo per il tuo account: crea, archivia e visualizza le informazioni di registrazione sull'utilizzo di Amazon Braket nel tuo account. Esegui query sui gruppi di log dei lavori ibridi. Comprende il percorso Braket corretto e consenti l'inserimento dei dati di registro. Inserisci i dati metrici. CloudWatch
- Archivia i dati in bucket Amazon S3: elenca i bucket S3 nel tuo account, inserisci oggetti e ottieni oggetti da qualsiasi bucket del tuo account che inizia con amazon-braket- nel nome. Queste autorizzazioni sono necessarie per consentire a Braket di inserire i file contenenti i risultati di attività quantistiche elaborate nel bucket e di recuperarli dal bucket.

- Passa IAM ruoli: trasferimento dei ruoli al IAM CreateJob API. I ruoli devono essere conformi al formato arn:aws:iam: :*:role/service-role/AmazonBraketJobsExecutionRole*.

```
"Version": "2012-10-17",
"Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetObject",
   "s3:PutObject",
   "s3:ListBucket",
   "s3:CreateBucket",
   "s3:PutBucketPublicAccessBlock",
   "s3:PutBucketPolicy"
  ],
  "Resource": "arn:aws:s3:::amazon-braket-*"
 },
 {
  "Effect": "Allow",
  "Action": [
   "ecr:GetDownloadUrlForLayer",
   "ecr:BatchGetImage",
   "ecr:BatchCheckLayerAvailability"
  ],
  "Resource": "arn:aws:ecr:*:*:repository/amazon-braket*"
 },
 {
  "Effect": "Allow",
  "Action": [
   "ecr:GetAuthorizationToken"
  ],
  "Resource": "*"
 },
 {
  "Effect": "Allow",
  "Action": [
   "braket:CancelJob",
   "braket:CancelQuantumTask",
   "braket:CreateJob",
   "braket:CreateQuantumTask",
   "braket:GetDevice",
   "braket:GetJob",
```

```
    "braket:GetQuantumTask",
    "braket:SearchDevices",
    "braket:SearchJobs",
    "braket:SearchQuantumTasks",
    "braket:ListTagsForResource",
    "braket:TagResource",
    "braket:UntagResource"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "iam:PassRole"
   ],
   "Resource": "arn:aws:iam::*:role/service-role/AmazonBraketJobsExecutionRole*",
   "Condition": {
    "StringLike": {
     "iam:PassedToService": [
      "braket.amazonaws.com"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iam:ListRoles"
   ],
   "Resource": "arn:aws:iam::*:role/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "logs:GetQueryResults"
   ],
   "Resource": [
    "arn:aws:logs:*:*:log-group:*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "logs:PutLogEvents",
```

```
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:GetLogEvents",
    "logs:DescribeLogStreams",
    "logs:StartQuery",
    "logs:StopQuery"
   ],
   "Resource": "arn:aws:logs:*:*:log-group:/aws/braket*"
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": "/aws/braket"
    }
   }
  }
 ]
}
```

Limita l'accesso degli utenti a determinati dispositivi

Per limitare l'accesso di determinati utenti a determinati dispositivi Braket, puoi aggiungere una politica di negazione delle autorizzazioni a uno specifico IAM ruolo.

Le seguenti azioni possono essere limitate con tali autorizzazioni:

- `CreateQuantumTask`- per negare la creazione di attività quantistiche su dispositivi specifici.
- `CreateJob`- negare la creazione di posti di lavoro ibridi su dispositivi specifici.
- `GetDevice`- negare di ottenere dettagli su dispositivi specifici.

L'esempio seguente limita l'accesso a tutti QPUs per Account AWS `123456789012`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Effect": "Deny",
    "Action": [
```

```
      "braket:CreateQuantumTask",
      "braket:CreateJob",
      "braket:GetDevice"
    ],
    "Resource": [
    "arn:aws:braket:*:*:device/qpu/*"
    ]
   }
  ]
}
```

> **Note**
>
> Escludi l'`braket:GetDevice`Action dalla policy per consentire l'accesso in lettura di un utente alle proprietà del dispositivo, come la disponibilità del dispositivo, i dati di calibrazione e i prezzi, tramite la console Braket.

Per adattare questo codice, sostituisci il Amazon Numero di risorsa (ARN) del dispositivo con restrizioni per la stringa mostrata nell'esempio precedente. Questa stringa fornisce il valore Resource. In Braket, un dispositivo rappresenta un QPU simulatore OR che è possibile chiamare per eseguire attività quantistiche. I dispositivi disponibili sono elencati nella pagina Dispositivi. Esistono due schemi utilizzati per specificare l'accesso a questi dispositivi:

- `arn:aws:braket:<region>:<account id>:device/qpu/<provider>/<device_id>`
- `arn:aws:braket:<region>:<account id>:device/quantum-simulator/<provider>/<device_id>`

Di seguito sono riportati alcuni esempi di vari tipi di accesso ai dispositivi

- Per selezionare QPUs tutte le aree geografiche: `arn:aws:braket:*:*:device/qpu/*`
- Per selezionare tutto QPUs nella regione us-west-2: ONLY `arn:aws:braket:us-west-2:123456789012:device/qpu/*`
- Equivalentemente, per selezionare tutto QPUs nella ONLY regione us-west-2 (poiché i dispositivi sono una risorsa di servizio, non una risorsa per i clienti): `arn:aws:braket:us-west-2:* :device/qpu/*`
- Per limitare l'accesso a tutti i dispositivi di simulazione su richiesta: `arn:aws:braket:* :123456789012:device/quantum-simulator/*`

- Per limitare l'accesso ai dispositivi di un determinato provider (ad esempio, a Rigetti QPU dispositivi): `arn:aws:braket:* :123456789012:device/qpu/rigetti/*`
- Per limitare l'accesso a TN1 dispositivo: `arn:aws:braket:* :123456789012:device/quantum-simulator/amazon/tn1`

Amazon Braket si aggiorna a AWS policy gestite

La tabella seguente fornisce dettagli sugli aggiornamenti di AWS politiche gestite per Braket da quando questo servizio ha iniziato a tenere traccia di queste modifiche.

Modifica	Descrizione	Data
AmazonBraketFullAccess- Politica di accesso completo per Braket	Sono state aggiunte le azioni servicequotas: GetServiceQuota e cloudwatch: da includere nella GetMetricData policy. AmazonBraketFullAccess .	24 marzo 2023
AmazonBraketFullAccess- Politica di accesso completo per Braket	Braket adjusted iam: PassRole autorizzazioni AmazonBraketFullAccess per includere il percorso. `service-role/`	29 novembre 2021
AmazonBraketJobsExecutionPolicy- Politica di esecuzione dei lavori ibridi per Amazon Braket Hybrid Jobs	Braket ha aggiornato il ruolo di esecuzione dei lavori ARN ibridi per includere il `service-role/` percorso.	29 novembre 2021
Braket ha iniziato a tenere traccia delle modifiche	Braket ha iniziato a tracciare le modifiche per il suo AWS politiche gestite.	29 novembre 2021

Limita l'accesso degli utenti a determinate istanze del notebook

Per limitare l'accesso di determinati utenti a istanze specifiche di Braket Notebook, puoi aggiungere una politica di negazione delle autorizzazioni a un ruolo, utente o gruppo specifico.

L'esempio seguente utilizza variabili di policy per limitare in modo efficiente le autorizzazioni all'avvio, all'arresto e all'accesso a istanze specifiche del notebook in Account AWS `123456789012`, che viene denominato in base all'utente che deve avere accesso (ad esempio, l'utente `Alice` avrebbe accesso a un'istanza del notebook denominata`amazon-braket-Alice`).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "sagemaker:CreateNotebookInstance",
        "sagemaker:DeleteNotebookInstance",
        "sagemaker:UpdateNotebookInstance",
        "sagemaker:CreateNotebookInstanceLifecycleConfig",
        "sagemaker:DeleteNotebookInstanceLifecycleConfig",
        "sagemaker:UpdateNotebookInstanceLifecycleConfig"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "sagemaker:DescribeNotebookInstance",
        "sagemaker:StartNotebookInstance",
        "sagemaker:StopNotebookInstance",
      ],
      "NotResource": [
        "arn:aws:sagemaker:*:123456789012:notebook-instance/amazon-braket-
${aws:username}"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "sagemaker:CreatePresignedNotebookInstanceUrl"
      ],
      "NotResource": [
```

```
        "arn:aws:sagemaker:*:123456789012:notebook-instance/amazon-braket-
${aws:username}*"
      ]
    }
  ]
}
```

Limita l'accesso degli utenti a determinati bucket S3

Per limitare l'accesso di determinati utenti a specifici bucket Amazon S3, puoi aggiungere una politica di rifiuto a un ruolo, utente o gruppo specifico.

L'esempio seguente limita le autorizzazioni per recuperare e collocare oggetti in uno specifico S3 bucket (`arn:aws:s3:::amazon-braket-us-east-1-123456789012-Alice`) e limita anche l'elenco di tali oggetti.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "s3:ListBucket"
      ],
      "NotResource": [
        "arn:aws:s3:::amazon-braket-us-east-1-123456789012-Alice"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "s3:GetObject"
      ],
      "NotResource": [
        "arn:aws:s3:::amazon-braket-us-east-1-123456789012-Alice/*"
      ]
    }
  ]
}
```

Per limitare l'accesso al bucket per una determinata istanza di notebook, puoi aggiungere la politica precedente al ruolo di esecuzione del notebook.

Ruolo collegato al servizio Amazon Braket

Quando abiliti Amazon Braket, nel tuo account viene creato un ruolo collegato al servizio.

Un ruolo collegato al servizio è un tipo di IAM ruolo unico che, in questo caso, è collegato direttamente ad Amazon Braket. Il ruolo collegato al servizio Amazon Braket è predefinito per includere tutte le autorizzazioni richieste da Braket quando chiama altri Servizi AWS per tuo conto.

Un ruolo collegato al servizio semplifica la configurazione di Amazon Braket perché non è necessario aggiungere manualmente le autorizzazioni necessarie. Amazon Braket definisce le autorizzazioni dei suoi ruoli collegati ai servizi. A meno che non modifichi queste definizioni, solo Amazon Braket può assumerne i ruoli. Le autorizzazioni definite includono la politica di fiducia e la politica di autorizzazione. La politica delle autorizzazioni non può essere associata a nessun'altra entità. IAM

Il ruolo collegato ai servizi impostato da Amazon Braket fa parte del AWS Identity and Access Management (IAM) funzionalità di ruoli collegati ai servizi. Per informazioni su altri Servizi AWS che supportano ruoli collegati ai servizi, vedi AWS Servizi compatibili IAM e cerca i servizi con Sì nella colonna Service-Linked Role. Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

In questa sezione:

- Autorizzazioni di ruolo collegate ai servizi per Amazon Braket

Autorizzazioni di ruolo collegate ai servizi per Amazon Braket

Amazon Braket utilizza il ruolo `AWSServiceRoleForAmazonBraket` collegato al servizio che affida l'assunzione del ruolo all'entità braket.amazonaws.com.

È necessario configurare le autorizzazioni per consentire a un'IAMentità (come un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio. Per ulteriori informazioni, vedere Autorizzazioni dei ruoli collegati ai servizi.

Per impostazione predefinita, al ruolo collegato ai servizi in Amazon Braket vengono concesse le seguenti autorizzazioni:

- Amazon S3: autorizzazioni per elencare i bucket nel tuo account e inserire oggetti e ottenere oggetti da qualsiasi bucket del tuo account con un nome che inizia con amazon-braket-.
- Amazon CloudWatch Logs: autorizzazioni per elencare e creare gruppi di log, creare i flussi di log associati e inserire eventi nel gruppo di log creato per Amazon Braket.

La seguente politica è allegata al ruolo collegato al servizio: **AWSServiceRoleForAmazonBraket**

```
{"Version": "2012-10-17",
    "Statement": [
        {"Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::amazon-braket*"
        },
        {"Effect": "Allow",
            "Action": [
                "logs:Describe*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/braket/*"
        },
        {"Effect": "Allow",
            "Action": "braket:*",
            "Resource": "*"
        },
        {"Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/braket.amazonaws.com/
AWSServiceRoleForAmazonBraket*",
            "Condition": {"StringEquals": {"iam:AWSServiceName": "braket.amazonaws.com"
                }
            }
        }
    ]
}
```

Convalida della conformità per Amazon Braket

I revisori di terze parti valutano regolarmente la sicurezza e la conformità di Amazon Braket e la nostra integrazione con fornitori di hardware di terze parti. Per un up-to-date elenco di informazioni sulla conformità per Braket, vedere Servizi AWSScope by compliance program. Per informazioni generali, vedere AWSconformità.

Puoi scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, vedere Scaricamento dei report in AWS Artifact.

> **Note**
>
> AWSi rapporti di conformità non coprono le QPU di fornitori di hardware di terze parti che possono scegliere di sottoporsi ai propri audit indipendenti.

La tua responsabilità in materia di conformità quando usi Amazon Braket è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità della tua azienda e dalle leggi e dai regolamenti applicabili. AWSfornisce le seguenti risorse per contribuire alla conformità:

- Guide Quick Start per la sicurezza e conformità: queste guide all'implementazione illustrano considerazioni relative all'architettura e forniscono fasi per l'implementazione di ambienti di base incentrati sulla sicurezza e sulla conformità su AWS.
- Risorse per la conformità di AWS: questa raccolta di workbook e guide potrebbe essere utile al tuo settore e alla tua posizione.

Sicurezza dell'infrastruttura in Amazon Braket

In quanto servizio gestito, Amazon Braket è protetto da AWS procedure di sicurezza di rete globali descritte nel AWS: white paper sulla panoramica dei processi di sicurezza.

Per accedere ad Amazon Braket tramite la rete, effettui chiamate a published AWS APIs. I client devono supportare Transport Layer Security (TLS) 1.2 o versione successiva. I client devono inoltre supportare suite di crittografia con Perfect Forward Secrecy (PFS) come Ephemeral Diffie-Hellman () o Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale. IAM Oppure puoi usare il AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per firmare le richieste.

Sicurezza dei fornitori di hardware Amazon Braket

QPUssu Amazon Braket sono ospitati da fornitori di hardware di terze parti. Quando esegui un'attività quantistica su un dispositivoQPU, Amazon Braket utilizza il ARN dispositivo come identificatore per inviare il circuito al luogo specificato per l'QPUelaborazione.

Se utilizzi Amazon Braket per accedere all'hardware di calcolo quantistico gestito da uno dei fornitori di hardware di terze parti, il tuo circuito e i dati associati vengono elaborati da fornitori di hardware esterni a strutture gestite da AWS. Informazioni sulla posizione fisica e AWS La regione in cui ciascuno QPU è disponibile è disponibile nella sezione Dettagli del dispositivo della console Amazon Braket.

I tuoi contenuti sono resi anonimi. Solo il contenuto necessario per elaborare il circuito viene inviato a terzi. Account AWS le informazioni non vengono trasmesse a terzi.

Tutti i dati sono crittografati sia a riposo che in transito. I dati vengono decrittografati solo per l'elaborazione. I fornitori terzi di Amazon Braket non sono autorizzati a archiviare o utilizzare i tuoi contenuti per scopi diversi dall'elaborazione del tuo circuito. Una volta completato il circuito, i risultati vengono restituiti ad Amazon Braket e archiviati nel bucket S3.

La sicurezza dei fornitori di hardware quantistico terzi di Amazon Braket viene verificata periodicamente per garantire il rispetto degli standard di sicurezza della rete, controllo degli accessi, protezione dei dati e sicurezza fisica.

VPCEndpoint Amazon per Amazon Braket

Puoi stabilire una connessione privata tra il tuo VPC e Amazon Braket creando un VPC endpoint di interfaccia. Gli endpoint dell'interfaccia sono alimentati da AWS PrivateLink, una tecnologia che consente l'accesso a Braket APIs senza un gateway Internet, un NAT dispositivo, una VPN connessione o AWS Direct Connect connessione. Le istanze del tuo VPC non hanno bisogno di indirizzi IP pubblici per comunicare con APIs Braket.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle tue sottoreti.

Con PrivateLink, il traffico tra te VPC e Braket non esce dal Amazon rete, che aumenta la sicurezza dei dati condivisi con le applicazioni basate sul cloud, poiché riduce l'esposizione dei dati alla rete Internet pubblica. Per ulteriori informazioni, consulta Interface VPC endpoints (AWS PrivateLink) nella Amazon VPC User Guide.

In questa sezione:

- Considerazioni sugli endpoint Amazon Braket VPC
- Configura Braket e PrivateLink
- Ulteriori informazioni sulla creazione di un endpoint
- Controlla l'accesso con le policy VPC degli endpoint di Amazon

Considerazioni sugli endpoint Amazon Braket VPC

Prima di configurare un VPC endpoint di interfaccia per Braket, assicurati di esaminare le proprietà e le limitazioni degli endpoint dell'interfaccia nella Amazon VPC User Guide.

Braket supporta l'esecuzione di chiamate a tutte le sue API azioni dal tuo. VPC

Per impostazione predefinita, l'accesso completo a Braket è consentito tramite l'VPCendpoint. È possibile controllare l'accesso se si specificano le politiche VPC degli endpoint. Per ulteriori informazioni, consulta Controllare l'accesso ai servizi con VPC endpoint nella Amazon VPC User Guide.

Configura Braket e PrivateLink

Per utilizzare AWS PrivateLink con Amazon Braket, devi creare un endpoint Amazon Virtual Private Cloud VPC (Amazon) come interfaccia e quindi connetterti all'endpoint tramite Amazon Braket API servizio.

Di seguito sono riportate le fasi generali di questo processo, spiegate in dettaglio nelle sezioni successive.

- Configura e avvia un Amazon VPC per ospitare il tuo AWS risorse. Se ne hai già unoVPC, puoi saltare questo passaggio.
- Crea un VPC endpoint Amazon per Braket
- Connect ed esegui le attività quantistiche di Braket tramite il tuo endpoint

Passaggio 1: avvia un Amazon VPC se necessario

Ricorda che puoi saltare questo passaggio se il tuo account ha già un VPC account operativo.

A VPC controlla le impostazioni di rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. In sostanza, stai lanciando il tuo AWS risorse in una rete virtuale personalizzata. Per ulteriori informazioniVPCs, consulta la Amazon VPC User Guide.

Apri la VPCconsole Amazon e creane una nuova VPC con sottoreti, gruppi di sicurezza e gateway di rete.

Passaggio 2: crea un endpoint di interfaccia VPC per Braket

Puoi creare un VPC endpoint per il servizio Braket utilizzando la VPC console Amazon o il AWS Command Line Interface (AWS CLI). Per ulteriori informazioni, consulta Creazione di un endpoint di interfaccia nella Amazon VPC User Guide.

Per creare un VPC endpoint nella console, apri la VPCconsole Amazon, apri la pagina Endpoints e procedi con la creazione del nuovo endpoint. Prendi nota dell'ID dell'endpoint per riferimento successivo. È obbligatorio come parte del `–endpoint-url` flag quando si effettuano determinate chiamate al Braket API.

Crea l'VPCendpoint per Braket utilizzando il seguente nome di servizio:

- `com.amazonaws.substitute_your_region.braket`

Nota: se abiliti la modalità privata DNS per l'endpoint, puoi creare API richiede a Braket utilizzando il DNS nome predefinito per la regione, ad esempio. `braket.us-east-1.amazonaws.com`

Per ulteriori informazioni, consulta Accedere a un servizio tramite un endpoint di interfaccia nella Amazon VPC User Guide.

Passaggio 3: Connect ed esegui le attività quantistiche di Braket tramite il tuo endpoint

Dopo aver creato un VPC endpoint, è possibile eseguire CLI comandi che includono il `endpoint-url` parametro per specificare gli endpoint dell'interfaccia verso API o runtime, come nell'esempio seguente:

```
aws braket search-quantum-tasks --endpoint-url
 VPC_Endpoint_ID.braket.substituteYourRegionHere.vpce.amazonaws.com
```

Se abiliti DNS i nomi host privati per il tuo VPC endpoint, non è necessario specificare l'endpoint come a URL nei comandi. CLI Invece, il metodo Amazon Braket API DNShostname, utilizzato da CLI e Braket per impostazione predefinita, viene SDK risolto nel tuo endpoint. VPC Ha la forma mostrata nell'esempio seguente:

```
https://braket.substituteYourRegionHere.amazonaws.com
```

Il post sul blog intitolato Accesso diretto ai SageMaker notebook Amazon da Amazon VPC utilizzando un AWS PrivateLink endpoint fornisce un esempio di come configurare un endpoint per stabilire connessioni sicure ai SageMaker notebook, che sono simili a Amazon Notebook Braket.

Se stai seguendo i passaggi descritti nel post del blog, ricordati di sostituire il nome Amazon Staffa per Amazon SageMaker. Per il nome del servizio, inserisci `com.amazonaws.us-east-1.braket` o sostituisci il nome corretto Regione AWS nome in quella stringa, se la tua regione non è us-east-1.

Ulteriori informazioni sulla creazione di un endpoint

- Per informazioni su come creare una VPC con sottoreti private, consulta Creare una con sottoreti private. VPC
- Per informazioni sulla creazione e configurazione di un endpoint utilizzando la VPC console Amazon o il AWS CLI, consulta Creazione di un endpoint di interfaccia nella Amazon VPC User Guide.
- Per informazioni sulla creazione e configurazione di un endpoint utilizzando AWS CloudFormation, vedi il AWS::EC2:: VPCEndpoint risorsa in AWS CloudFormation Guida per l'utente.

Controlla l'accesso con le policy VPC degli endpoint di Amazon

Per controllare l'accesso alla connettività ad Amazon Braket, puoi collegare un AWS Identity and Access Management (IAM) politica degli endpoint per il tuo VPC endpoint Amazon. Questa policy specifica le informazioni riportate di seguito:

- Il principale (utente o ruolo) che può eseguire azioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire azioni.

Per ulteriori informazioni, consulta Controllare l'accesso ai servizi con VPC endpoint nella Amazon VPC User Guide.

Esempio: policy VPC sugli endpoint per le azioni Braket

L'esempio seguente mostra una policy sugli endpoint per Braket. Se collegata a un endpoint, questa politica consente l'accesso alle azioni Braket elencate per tutti i principali su tutte le risorse.

```
{
 "Statement":[
 {
   "Principal":"*",
   "Effect":"Allow",
   "Action":[
     “braket:action-1",
     “braket:action-2",
     “braket:action-3”
     ],
   "Resource":"*"
   }
  ]
}
```

È possibile creare IAM regole complesse allegando più policy per gli endpoint. Per ulteriori informazioni ed esempi, consulta:

- Policy degli endpoint di Amazon Virtual Private Cloud per Step Functions
- Creazione di IAM autorizzazioni granulari per utenti non amministratori
- Controllo dell'accesso ai servizi con endpoint VPC

Registrazione di log e monitoraggio

Dopo aver inviato un'attività quantistica, è possibile tenerne traccia dello stato tramite Amazon Staffa SDK e console. Una volta completata l'attività quantistica, Braket salva i risultati nella posizione Amazon S3 specificata. Il completamento può richiedere del tempo, specialmente per QPU i dispositivi, a seconda della lunghezza della coda. I tipi di stato includono:

- `CREATED`— Amazon Braket ha ricevuto il tuo compito quantistico.
- `QUEUED`— Amazon Braket ha elaborato il tuo task quantistico ed è ora in attesa di essere eseguito sul dispositivo.
- `RUNNING`— La tua attività quantistica è in esecuzione su un QPU simulatore o su richiesta.
- `COMPLETED`— L'esecuzione dell'attività quantistica è terminata sul simulatore o su richiesta. QPU
- `FAILED`— Il tuo compito quantistico ha tentato di essere eseguito ed è fallito. A seconda del motivo per cui il task quantistico non è riuscito, provate a inviarlo nuovamente.
- `CANCELLED`— Hai annullato l'operazione quantistica. Il task quantistico non è stato eseguito.

In questa sezione:

Monitoraggio delle attività quantistiche da Amazon Braket SDK

Il comando `device.run(…)` definisce un'attività quantistica con un ID di attività quantistica univoco. È possibile interrogare e tenere traccia dello stato con `task.state()` come mostrato nell'esempio seguente.

Nota: `task = device.run()` è un'operazione asincrona, il che significa che potete continuare a lavorare mentre il sistema elabora il vostro compito quantistico in background.

Recupera un risultato

Quando chiami`task.result()`, SDK inizia il sondaggio Amazon Fai una staffa per vedere se il compito quantistico è completo. SDKUtilizza i parametri di polling che hai definito. `.run()` Una volta completata l'attività quantistica, SDK recupera il risultato dal bucket S3 e lo restituisce come oggetto. `QuantumTaskResult`

```
# create a circuit, specify the device and run the circuit
circ = Circuit().rx(0, 0.15).ry(1, 0.2).cnot(0,2)
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
task = device.run(circ, s3_location, shots=1000)

# get ID and status of submitted task
task_id = task.id
status = task.state()
print('ID of task:', task_id)
print('Status of task:', status)
# wait for job to complete
while status != 'COMPLETED':
    status = task.state()
    print('Status:', status)
```

```
ID of task:
arn:aws:braket:us-west-2:123412341234:quantum-task/b68ae94b-1547-4d1d-aa92-1500b82c300d
Status of task: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: RUNNING
Status: RUNNING
Status: COMPLETED
```

Annulla un'attività quantistica

Per annullare un'operazione quantistica, chiamate il `cancel()` metodo, come illustrato nell'esempio seguente.

```
# cancel quantum task
```

```
task.cancel()
status = task.state()
print('Status of task:', status)
```

```
Status of task: CANCELLING
```

Controllate i metadati

È possibile controllare i metadati dell'operazione quantistica completata, come mostrato nell'esempio seguente.

```
# get the metadata of the quantum task
metadata = task.metadata()
# example of metadata
shots = metadata['shots']
date = metadata['ResponseMetadata']['HTTPHeaders']['date']
# print example metadata
print("{} shots taken on {}.".format(shots, date))

# print name of the s3 bucket where the result is saved
results_bucket = metadata['outputS3Bucket']
print('Bucket where results are stored:', results_bucket)
# print the s3 object key (folder name)
results_object_key = metadata['outputS3Directory']
print('S3 object key:', results_object_key)

# the entire look-up string of the saved result data
look_up = 's3://'+results_bucket+'/'+results_object_key
print('S3 URI:', look_up)
```

```
1000 shots taken on Wed, 05 Aug 2020 14:44:22 GMT.
Bucket where results are stored: amazon-braket-123412341234
S3 object key: simulation-output/b68ae94b-1547-4d1d-aa92-1500b82c300d
S3 URI: s3://amazon-braket-123412341234/simulation-output/b68ae94b-1547-4d1d-
aa92-1500b82c300d
```

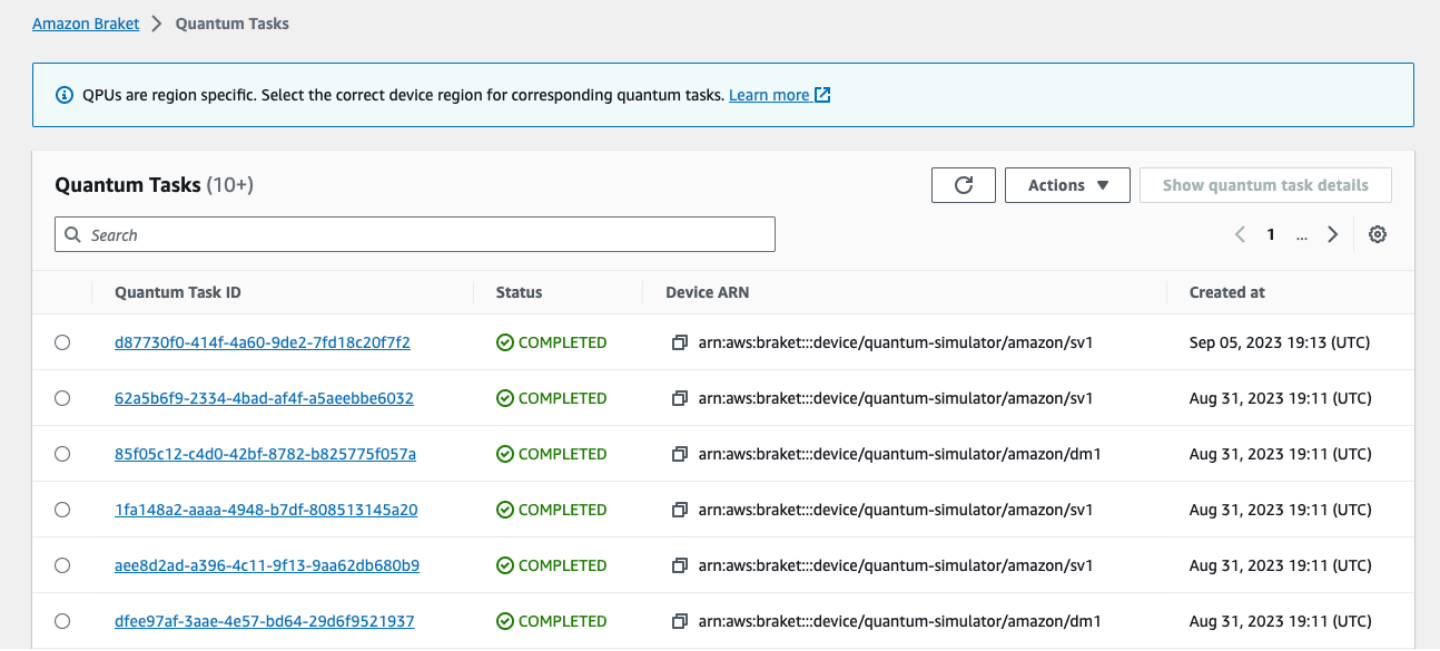
Recuperate un'attività o un risultato quantistico

Se il kernel muore dopo aver inviato il task quantistico o se chiudi il notebook o il computer, puoi ricostruire l'`task`oggetto con il suo ID univoco ARN (quantum task ID). Quindi puoi chiamare `task.result()` per ottenere il risultato dal bucket S3 in cui è memorizzato.

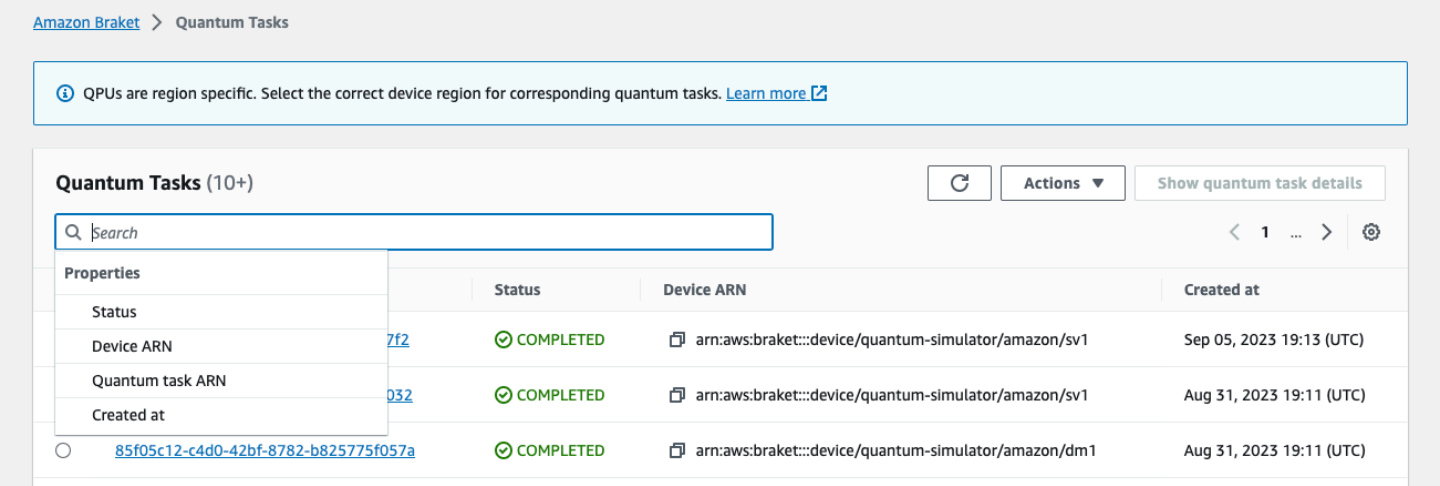
```
from braket.aws import AwsSession, AwsQuantumTask

# restore task with unique arn
task_load = AwsQuantumTask(arn=task_id)
# retrieve the result of the task
result = task_load.result()
```

Monitoraggio delle attività quantistiche tramite la console Amazon Braket

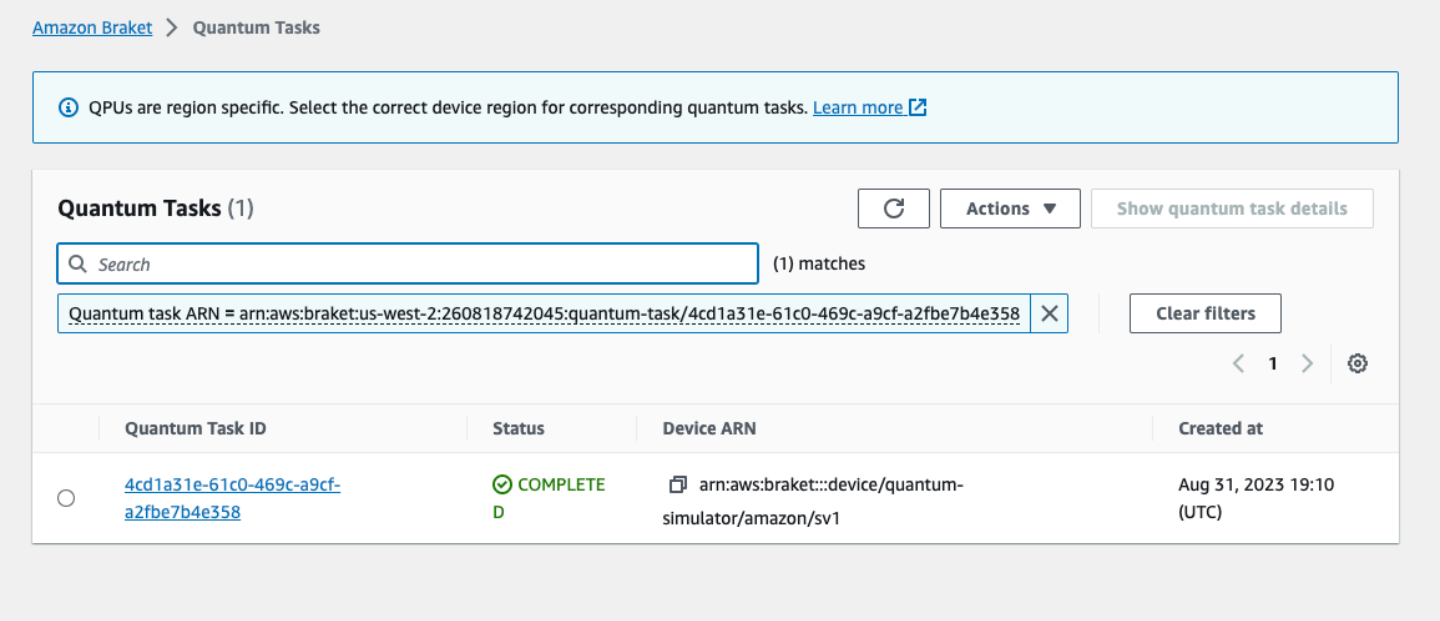
Amazon Braket offre un modo pratico per monitorare l'attività quantistica tramite la console Amazon Braket. Tutte le attività quantistiche inviate sono elencate nel campo Attività quantistiche, come mostrato nella figura seguente. Questo servizio è specifico della regione, il che significa che è possibile visualizzare solo le attività quantistiche create nello specifico Regione AWS.



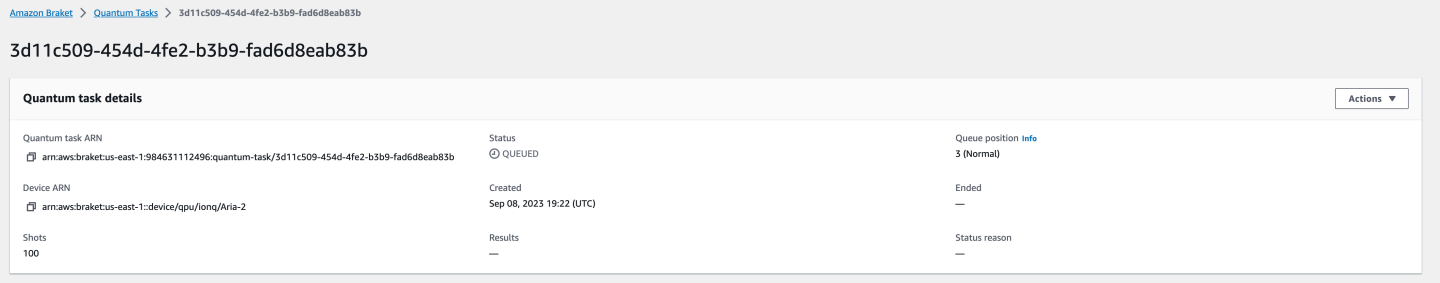
Puoi cercare attività quantistiche particolari tramite la barra di navigazione. La ricerca può essere basata su Quantum Task ARN (ID), stato, dispositivo e ora di creazione. Le opzioni vengono visualizzate automaticamente quando si seleziona la barra di navigazione, come illustrato nell'esempio seguente.



L'immagine seguente mostra un esempio di ricerca di un'attività quantistica in base al relativo ID univoco di attività quantistica, che può essere ottenuto chiamando. `task.id`



Inoltre, come mostrato nella figura seguente, lo stato di un'attività quantistica può essere monitorato mentre si trova in uno stato. QUEUED Facendo clic sull'ID dell'attività quantistica viene visualizzata la pagina dei dettagli. Questa pagina mostra la posizione dinamica della coda per il task quantistico rispetto al dispositivo su cui verrà elaborato.



Le attività quantistiche inviate come parte di un processo ibrido avranno la priorità quando sono in coda. Le attività quantistiche inviate al di fuori di un lavoro ibrido avranno la normale priorità di messa in coda.

I clienti che desiderano interrogare Braket SDK possono ottenere le posizioni relative alle attività quantistiche e alle posizioni ibride nella coda dei lavori in modo programmatico. Per ulteriori informazioni, consulta la pagina Quando verrà eseguita la mia attività.

Etichettare le risorse di Amazon Braket

Un tag è un'etichetta di attributo personalizzata che assegni tu o AWS assegna a un AWS risorsa. Un tag è un metadato che fornisce maggiori informazioni sulla tua risorsa. Ciascun tag è formato da una chiave e da un valore, Tutti questi sono noti come coppie chiave-valore. Per i tag assegnati da te, puoi definire la chiave e il valore.

Nel Amazon Con la console Braket, è possibile accedere a un'attività quantistica o a un taccuino e visualizzare l'elenco dei tag ad esso associati. È possibile aggiungere un tag, rimuovere un tag o modificare un tag. È possibile etichettare un task quantistico o un taccuino al momento della creazione e quindi gestire i tag associati tramite la console, AWS CLI, oppure API.

Maggiori informazioni su AWS e tag

- Per informazioni generali sull'etichettatura, incluse le convenzioni di denominazione e utilizzo, consulta Tagging AWS Risorse in AWS Riferimento generale.
- Per informazioni sulle restrizioni relative all'etichettatura, consulta Limiti e requisiti di denominazione dei tag nella AWS Riferimento generale.
- Per le migliori pratiche e le strategie di etichettatura, consulta Migliori pratiche di etichettatura e AWS Strategie di etichettatura.
- Per un elenco di servizi che supportano l'utilizzo dei tag, consulta il Resource Groups Tagging API Reference.

Le seguenti sezioni forniscono informazioni più specifiche sui tag per Amazon Braket.

In questa sezione:

Utilizzo dei tag

I tag possono organizzare le tue risorse in categorie che ti sono utili. Ad esempio, puoi assegnare un tag «Dipartimento» per specificare il dipartimento che possiede questa risorsa.

Ogni tag è costituito da due parti:

- Una chiave di tag (ad esempio CostCenter, Ambiente o Progetto). Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole.
- Un campo opzionale noto come valore di tag (ad esempio, 111122223333 o Production). Non specificare il valore del tag equivale a utilizzare una stringa vuota. Analogamente alle chiavi dei tag, i valori dei tag prevedono una distinzione tra lettere maiuscole e minuscole.

I tag ti aiutano a fare le seguenti cose:

- Identifica e organizza i tuoi AWS risorse. Molti Servizi AWS supportano l'etichettatura, in modo da poter assegnare lo stesso tag a risorse di servizi diversi per indicare che le risorse sono correlate.
- Tieni traccia dei tuoi AWS costi. Questi tag vengono attivati su AWS Billing and Cost Management cruscotto. AWS utilizza i tag per classificare i costi e fornirti un rapporto mensile sull'allocazione dei costi. Per ulteriori informazioni, consulta Utilizzare i tag di allocazione dei costi nel AWS Billing and Cost Management Guida per l'utente.
- Controlla l'accesso al tuo AWS risorse. Per ulteriori informazioni, consulta Controllo dell'accesso tramite tag.

Risorse supportate per l'etichettatura in Amazon Braket

Il seguente tipo di risorsa in Amazon Braket supporta l'etichettatura:

- Risorsa `quantum-task`
- Nome della risorsa: `AWS::Service::Braket`
- ARNRegex: `arn:${Partition}:braket:${Region}:${Account}:quantum-task/${RandomId}`

Nota: puoi applicare e gestire i tag per Amazon Staffa i taccuini nel Amazon Console Braket, utilizzando la console per accedere alla risorsa del notebook, sebbene i notebook siano in realtà risorse Amazon. SageMaker Per ulteriori informazioni, consulta Notebook Instance Metadata nella documentazione. SageMaker

Restrizioni di tagging

Le seguenti restrizioni di base si applicano ai tag nelle risorse Amazon Braket:

- Numero massimo di tag che puoi assegnare a una risorsa: 50
- lunghezza massima della chiave: 128 caratteri Unicode;
- lunghezza massima del valore: 256 caratteri Unicode;
- Caratteri validi per chiave e valore: `a-z, A-Z, 0-9, space` e questi caratteri: `_ . : / = + -` e `@`
- Per chiavi e valori viene fatta distinzione tra maiuscole e minuscole.
- Non utilizzare `aws` come prefisso per le chiavi; è riservato a AWS usare.

Gestione dei tag in Amazon Braket

I tag vengono impostati come proprietà su una risorsa. Puoi visualizzare, aggiungere, modificare, elencare ed eliminare i tag tramite la console Amazon Braket, Amazon Braket API, oppure AWS CLI. Per ulteriori informazioni, consulta la pagina di riferimento di Amazon Braket. API

In questa sezione:

- Aggiungere tag
- Visualizzazione dei tag
- Modifica dei tag

- Rimozione dei tag

Aggiungere tag

Puoi aggiungere tag alle risorse taggabili nei seguenti orari:

- Quando crei la risorsa: utilizza la console o includi il `Tags` parametro con l'`Create`operazione nel AWS API.
- Dopo aver creato la risorsa: utilizzate la console per accedere alla risorsa Quantum Task o Notebook oppure chiamate l'`TagResource`operazione nel AWS API.

Per aggiungere tag a una risorsa al momento della creazione, è inoltre necessaria l'autorizzazione a creare una risorsa del tipo specificato.

Visualizzazione dei tag

Puoi visualizzare i tag su qualsiasi risorsa etichettabile in Amazon Braket utilizzando la console per accedere alla risorsa task o notebook oppure chiamando il AWS `ListTagsForResource` API operazione.

È possibile utilizzare quanto segue AWS API comando per visualizzare i tag su una risorsa:

- AWS API: `ListTagsForResource`

Modifica dei tag

È possibile modificare i tag utilizzando la console per accedere all'attività quantistica o alla risorsa notebook oppure è possibile utilizzare il comando seguente per modificare il valore di un tag allegato a una risorsa etichettabile. Quando specificate una chiave di tag già esistente, il valore di quella chiave viene sovrascritto:

- AWS API: `TagResource`

Rimozione dei tag

È possibile rimuovere i tag da una risorsa specificando le chiavi da rimuovere, utilizzando la console per accedere alla risorsa quantum task o notebook o quando si chiama l'operazione. `UntagResource`

- AWS API: `UntagResource`

Esempio di CLI etichettatura in Amazon Braket

Se stai lavorando con AWS CLI, ecco un comando di esempio che mostra come creare un tag che si applica a un task quantistico creato per SV1 con le impostazioni dei parametri di Rigetti QPU. Notate che il tag è specificato alla fine del comando di esempio. In questo caso, a Key viene assegnato il valore **`state`** e a Value viene assegnato il valore`Washington`.

```
aws braket create-quantum-task --action /
"{\"braketSchemaHeader\": {\"name\": \"braket.ir.jaqcd.program\", /
    \"version\": \"1\"}, /
    \"instructions\": [{\"angle\": 0.15, \"target\": 0, \"type\": \"rz\"}], /
    \"results\": null, /
    \"basis_rotation_instructions\": null}" /
  --device-arn "arn:aws:braket:::device/quantum-simulator/amazon/sv1" /
  --output-s3-bucket  "my-example-braket-bucket-name" /
  --output-s3-key-prefix "my-example-username"  /
  --shots 100   /
  --device-parameters /
  "{\"braketSchemaHeader\": /
     {\"name\": \"braket.device_schema.rigetti.rigetti_device_parameters\", /
      \"version\": \"1\"}, \"paradigmParameters\": /
       {\"braketSchemaHeader\": /
         {\"name\": \"braket.device_schema.gate_model_parameters\", /
          \"version\": \"1\"}, /
          \"qubitCount\": 2}}" /
          --tags {\"state\":\”Washington\"}
```

Etichettatura con Amazon Braket API

- Se utilizzi Amazon Braket API per impostare i tag su una risorsa, chiama il `TagResource`API.

```
aws braket tag-resource --resource-arn $YOUR_TASK_ARN --tags {\"city\":
\"Seattle\"}
```

- Per rimuovere i tag da una risorsa, chiamate il `UntagResource`API.

```
aws braket list-tags-for-resource --resource-arn $YOUR_TASK_ARN
```

- Per elencare tutti i tag associati a una particolare risorsa, chiamate il `ListTagsForResource`API.

```
aws braket tag-resource --resource-arn $YOUR_TASK_ARN --tag-keys "[\"city
\",\"state\"]"
```

Monitoraggio delle attività quantistiche con EventBridge

Amazon EventBridge monitora gli eventi di modifica dello stato nelle attività quantistiche di Amazon Braket. Gli eventi di Amazon Braket vengono consegnati a EventBridge, quasi in tempo reale. Puoi scrivere semplici regole che indichino quali eventi ti interessano, includendo operazioni automatizzate da eseguire quando si verifica un evento previsto da una regola. Le azioni automatiche che possono essere attivate includono:

- Invocare un AWS Lambda funzione
- Attivazione di un AWS Step Functions macchina a stati
- Notifica di un argomento su Amazon SNS

EventBridge monitora questi eventi di modifica dello stato di Amazon Braket:

- Lo stato delle attività quantistiche cambia

Amazon Braket garantisce la fornitura di eventi di modifica dello stato delle attività quantistiche. Questi eventi vengono forniti almeno una volta, ma potrebbero non funzionare correttamente.

Per ulteriori informazioni, vedere Eventi e modelli di eventi in EventBridge.

In questa sezione:

Monitora lo stato delle attività quantistiche con EventBridge

Con EventBridge, puoi creare regole che definiscono le azioni da intraprendere quando Amazon Braket invia una notifica di una modifica dello stato relativa a un'attività quantistica di Braket. Ad esempio, puoi creare una regola che ti invii un messaggio e-mail ogni volta che lo stato di un'attività quantistica cambia.

1. Effettua il login a AWS utilizzando un account che dispone delle autorizzazioni necessarie all'uso EventBridge e Amazon Staffa.
2. Apri la EventBridge console Amazon all'indirizzo https://console.aws.amazon.com/events/.
3. Utilizzando i seguenti valori, crea una EventBridge regola:
   - Per Rule type (Tipo di regola), scegli Rule with an event pattern (Regola con un modello di eventi).
   - In Event source (Origine eventi), scegli Other (Altro).
   - Nella sezione Schema di eventi, scegliete Modelli personalizzati (JSONeditor), quindi incollate il seguente modello di evento nell'area di testo:

     ```
     {
       "source": [
         "aws.braket"
       ],
       "detail-type": [
         "Braket Task State Change"
       ]
     }
     ```

     Per acquisire tutti gli eventi da Amazon Braket, escludi la `detail-type` sezione come mostrato nel codice seguente:

     ```
     {
       "source": [
         "aws.braket"
       ]
     }
     ```

   - Per i tipi di Target, scegli Servizio AWS, e per Seleziona un obiettivo, scegli un obiettivo come un SNS argomento Amazon o AWS Lambda funzione. Il bersaglio viene attivato quando viene ricevuto un evento di cambiamento dello stato dell'attività quantistica da Amazon Staffa.

     Ad esempio, utilizza un argomento Amazon Simple Notification Service (SNS) per inviare un'e-mail o un messaggio di testo quando si verifica un evento. Per farlo, crea prima un SNS argomento Amazon utilizzando la SNS console Amazon. Per ulteriori informazioni, consulta Usare Amazon SNS per le notifiche agli utenti.

Per informazioni dettagliate sulla creazione di regole, consulta Creazione di EventBridge regole Amazon che reagiscono agli eventi.

Esempio di evento Amazon Braket EventBridge

Per informazioni sui campi per un evento Amazon Braket Quantum Task Status Change, consulta Eventi e modelli di eventi in. EventBridge

I seguenti attributi vengono visualizzati nel campo JSON «dettaglio».

- **`quantumTaskArn`**(str): L'attività quantistica per la quale è stato generato questo evento.
- **`status`**(Opzionale [str]): Lo stato a cui è passata l'attività quantistica.
- **`deviceArn`**(str): Il dispositivo specificato dall'utente per il quale è stata creata questa operazione quantistica.
- shots (int): Il numero di shots richiesto dall'utente.
- **`outputS3Bucket`**(str): Il bucket di output specificato dall'utente.
- **`outputS3Directory`**(str): Il prefisso della chiave di output specificato dall'utente.
- **`createdAt`**(str): Il tempo di creazione dell'attività quantistica come stringa ISO -8601.
- **`endedAt`**(Opzionale [str]): L'ora in cui l'attività quantistica ha raggiunto uno stato terminale. Questo campo è presente solo quando l'attività quantistica è passata a uno stato terminale.

Il JSON codice seguente mostra un esempio di Amazon Evento Braket Quantum Task Status Change.

```
{
    "version":"0",
    "id":"6101452d-8caf-062b-6dbc-ceb5421334c5",
    "detail-type":"Braket Task State Change",
    "source":"aws.braket",
    "account":"012345678901",
    "time":"2021-10-28T01:17:45Z",
    "region":"us-east-1",
    "resources":[
        "arn:aws:braket:us-east-1:012345678901:quantum-task/834b21ed-77a7-4b36-a90c-
c776afc9a71e"
    ],
    "detail":{
        "quantumTaskArn":"arn:aws:braket:us-east-1:012345678901:quantum-
task/834b21ed-77a7-4b36-a90c-c776afc9a71e",
```

```
        "status":"COMPLETED",
        "deviceArn":"arn:aws:braket:::device/quantum-simulator/amazon/sv1",
        "shots":"100",
        "outputS3Bucket":"amazon-braket-0260a8bc871e",
        "outputS3Directory":"sns-testing/834b21ed-77a7-4b36-a90c-c776afc9a71e",
        "createdAt":"2021-10-28T01:17:42.898Z",
        "eventName":"MODIFY",
        "endedAt":"2021-10-28T01:17:44.735Z"
      }
}
```


Monitoraggio delle metriche con CloudWatch

Puoi monitorare Amazon Braket utilizzando Amazon CloudWatch, che raccoglie dati grezzi e li elabora in metriche leggibili quasi in tempo reale. Visualizza le informazioni storiche generate fino a 15 mesi fa o le metriche di ricerca aggiornate nelle ultime 2 settimane nella CloudWatch console Amazon per avere una prospettiva migliore sulle prestazioni di Amazon Braket. Per ulteriori informazioni, consulta Utilizzo CloudWatch delle metriche.

> **Note**
>
> Puoi visualizzare i flussi di CloudWatch log per i notebook Amazon Braket accedendo alla pagina dei dettagli del notebook sulla console Amazon. SageMaker Ulteriori impostazioni del notebook Amazon Braket sono disponibili tramite la SageMaker console.

In questa sezione:

- Metriche e dimensioni di Amazon Braket

Metriche e dimensioni di Amazon Braket

Le metriche sono il concetto fondamentale in. CloudWatch Una metrica rappresenta un insieme di punti dati ordinati nel tempo su cui vengono pubblicati. CloudWatch Ogni metrica è caratterizzata da un insieme di dimensioni. Per ulteriori informazioni sulle dimensioni delle metriche in CloudWatch, consulta CloudWatch Dimensioni.

Amazon Braket invia i seguenti dati metrici, specifici di Amazon Braket, ai parametri Amazon: CloudWatch

Quantum Task Metrics

Le metriche sono disponibili se esistono attività quantistiche. Sono visualizzate sotto AWS/Braket/By Device nella console. CloudWatch

Parametro	Descrizione
Conteggio	Numero di attività quantistiche.
Latenza	Questa metrica viene emessa quando viene completata un'attività quantistica. Rappresenta il tempo totale dall'inizializzazione al completamento dell'attività quantistica.

Dimensioni per Quantum Task Metrics

Le metriche delle attività quantistiche sono pubblicate con una dimensione basata sul **`deviceArn`** parametro, che ha la forma arn:aws:braket: ::device/xxx.

Registrazione delle attività quantistiche con CloudTrail

Amazon Braket è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un Servizio AWS in Amazon Braket. CloudTrail cattura tutto API chiama Amazon Braket come eventi. Le chiamate acquisite includono chiamate dalla console Amazon Braket e chiamate in codice verso le operazioni Amazon Braket. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon Braket. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta effettuata ad Amazon Braket, l'indirizzo IP da cui è stata effettuata, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta il AWS CloudTrail Guida per l'utente.

In questa sezione:

- Informazioni su Amazon Braket in CloudTrail
- Comprendere le voci dei file di log di Amazon Braket

Informazioni su Amazon Braket in CloudTrail

CloudTrail è abilitato sul tuo Account AWS quando crei l'account. Quando si verifica un'attività in Amazon Braket, tale attività viene registrata in un CloudTrail evento insieme ad altri Servizio AWS eventi nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nella Account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo Account AWS, inclusi gli eventi per Amazon Braket, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutti Regioni AWS. Il percorso registra gli eventi di tutte le regioni del AWS esegue il partizionamento e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurarne altri Servizi AWS per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail log. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione di Amazon SNS Notifications per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutti Amazon Le azioni di Braket vengono registrate da. CloudTrail Ad esempio, le chiamate alle `GetDevice` azioni `GetQuantumTask` o generano voci nei file di CloudTrail registro.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS.

Per ulteriori informazioni, consulta l'CloudTrail userIdentity elemento.

Comprendere le voci dei file di log di Amazon Braket

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia stack ordinata del pubblico API chiamate, quindi non appaiono in un ordine specifico.

L'esempio seguente è una voce di registro per l'`GetQuantumTask`azione, che ottiene i dettagli di un'attività quantistica.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "foobar",
    "arn": "foobar",
    "accountId": "foobar",
    "accessKeyId": "foobar",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "foobar",
        "arn": "foobar",
        "accountId": "foobar",
        "userName": "foobar"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-08-07T00:56:57Z"
      }
    }
  },
  "eventTime": "2020-08-07T01:00:08Z",
  "eventSource": "braket.amazonaws.com",
  "eventName": "GetQuantumTask",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "foobar",
  "userAgent": "aws-cli/1.18.110 Python/3.6.10
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.17.33",
```

```
  "requestParameters": {
    "quantumTaskArn": "foobar"
  },
  "responseElements": null,
  "requestID": "20e8000c-29b8-4137-9cbc-af77d1dd12f7",
  "eventID": "4a2fdb22-a73d-414a-b30f-c0797c088f7c",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "foobar"
}
```

Di seguito viene mostrata una voce di registro relativa all'`GetDevice`azione, che restituisce i dettagli di un evento del dispositivo.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "foobar",
    "arn": "foobar",
    "accountId": "foobar",
    "accessKeyId": "foobar",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "foobar",
        "arn": "foobar",
        "accountId": "foobar",
        "userName": "foobar"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-08-07T00:46:29Z"
      }
    }
  },
  "eventTime": "2020-08-07T00:46:32Z",
  "eventSource": "braket.amazonaws.com",
  "eventName": "GetDevice",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "foobar",
```

```
  "userAgent": "Boto3/1.14.33 Python/3.7.6 Linux/4.14.158-129.185.amzn2.x86_64 exec-
env/AWS_ECS_FARGATE Botocore/1.17.33",
  "errorCode": "404",
  "requestParameters": {
    "deviceArn": "foobar"
  },
  "responseElements": null,
  "requestID": "c614858b-4dcf-43bd-83c9-bcf9f17f522e",
  "eventID": "9642512a-478b-4e7b-9f34-75ba5a3408eb",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "foobar"
}
```

Registrazione avanzata

È possibile registrare l'intero processo di elaborazione delle attività utilizzando un logger. Queste tecniche di registrazione avanzate consentono di visualizzare il polling in background e di creare un record per il debug successivo.

Per utilizzare il logger, consigliamo di modificare i `poll_interval_seconds` parametri `poll_timeout_seconds` and, in modo che un'attività quantistica possa durare a lungo e lo stato dell'attività quantistica venga registrato continuamente, con i risultati salvati in un file. È possibile trasferire questo codice su uno script Python anziché su un notebook Jupyter, in modo che lo script possa essere eseguito come processo in background.

Configura il logger

Innanzitutto, configura il logger in modo che tutti i registri vengano scritti automaticamente in un file di testo, come mostrato nelle seguenti righe di esempio.

```
# import the module
import logging
from datetime import datetime

# set filename for logs
log_file = 'device_logs-'+datetime.strftime(datetime.now(), '%Y%m%d%H%M%S')+'.txt'
print('Task info will be logged in:', log_file)

# create new logger object
logger = logging.getLogger("newLogger")
```

```
# configure to log to file device_logs.txt in the appending mode
logger.addHandler(logging.FileHandler(filename=log_file, mode='a'))

# add to file all log messages with level DEBUG or above
logger.setLevel(logging.DEBUG)
```

```
Task info will be logged in: device_logs-20200803203309.txt
```

Crea ed esegui il circuito

Ora puoi creare un circuito, inviarlo a un dispositivo per farlo funzionare e vedere cosa succede, come mostrato in questo esempio.

```
# define circuit
circ_log = Circuit().rx(0, 0.15).ry(1, 0.2).rz(2, 0.25).h(3).cnot(control=0,
 target=2).zz(1, 3, 0.15).x(4)
print(circ_log)
# define backend
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
# define what info to log
logger.info(
    device.run(circ_log, s3_location,
               poll_timeout_seconds=1200, poll_interval_seconds=0.25, logger=logger,
 shots=1000)
    .result().measurement_counts
)
```

Controllate il file di registro

Puoi controllare cosa è scritto nel file inserendo il seguente comando.

```
# print logs
! cat {log_file}
```

```
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: start polling for completion
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status CREATED
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status CREATED
```

```
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status QUEUED
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status RUNNING
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status RUNNING
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status COMPLETED
Counter({'00001': 493, '00011': 493, '01001': 5, '10111': 4, '01011': 3, '10101': 2})
```

Ottieni il file ARN dal file di registro

Dall'output del file di registro restituito, come mostrato nell'esempio precedente, è possibile ottenere le ARN informazioni. Con l'ARNID, è possibile recuperare il risultato dell'operazione quantistica completata.

```
# parse log file for arn
with open(log_file) as openfile:
    for line in openfile:
        for part in line.split():
            if "arn:" in part:
                arn = part
                break
# remove final semicolon in logs
arn = arn[:-1]

# with this arn you can restore again task from unique arn
task_load = AwsQuantumTask(arn=arn, aws_session=AwsSession())

# get results of task
result = task_load.result()
```

Quote Amazon Braket

La tabella seguente elenca le quote di servizio per Amazon Braket. Le quote di servizio, note anche come limiti, sono il numero massimo di risorse o operazioni di servizio per i Account AWS.

Alcune quote possono essere aumentate. Per ulteriori informazioni, consulta Servizio AWS quote.

- Le quote burst rate non possono essere aumentate.
- L'aumento massimo del tasso per le quote regolabili (ad eccezione del burst rate, che non può essere modificato) è 2 volte il limite di tasso predefinito specificato. Ad esempio, una quota predefinita di 60 può essere regolata fino a un massimo di 120.
- La quota regolabile per i concorrenti SV1 (DM1) le attività quantistiche consentono un massimo di 60 per Regione AWS.
- Il numero massimo consentito di istanze di calcolo per un processo ibrido è 5 e le quote sono regolabili.

Risorsa	Descrizione	Limiti	Regolabile
Tasso di API richieste	Il numero massimo di richieste al secondo che puoi inviare in questo account nella regione corrente.	140	Sì
Frequenza di scoppio di API richieste	Il numero massimo di richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	600	No
Frequenza delle richieste `CreateQuantumTask`	Il numero massimo di `CreateQuantumTask` richieste che puoi inviare al	20	Sì

Risorsa	Descrizione	Limiti	Regolabile
	secondo in questo account per regione.		
Frequenza di burst delle richieste `CreateQuantumTask`	Il numero massimo di `CreateQuantumTask` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	40	No
Frequenza delle richieste `SearchQuantumTasks`	Il numero massimo di `SearchQuantumTasks` richieste che puoi inviare al secondo in questo account per regione.	5	Sì
Frequenza di burst delle richieste `SearchQuantumTasks`	Il numero massimo di `SearchQuantumTasks` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No

Risorsa	Descrizione	Limiti	Regolabile
Frequenza delle richieste `GetQuantumTask`	Il numero massimo di `GetQuantumTask` richieste che puoi inviare al secondo in questo account per regione.	100	Sì
Frequenza di burst delle richieste `GetQuantumTask`	Il numero massimo di `GetQuantumTask` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	500	No
Frequenza delle richieste `CancelQuantumTask`	Il numero massimo di `CancelQuantumTask` richieste che puoi inviare al secondo in questo account per regione.	2	Sì
Frequenza di burst delle richieste `CancelQuantumTask`	Il numero massimo di `CancelQuantumTask` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	20	No

Risorsa	Descrizione	Limiti	Regolabile
Frequenza delle richieste `GetDevice`	Il numero massimo di `GetDevice` richieste che puoi inviare al secondo in questo account per regione.	5	Sì
Frequenza di burst delle richieste `GetDevice`	Il numero massimo di `GetDevice` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No
Frequenza delle richieste `SearchDevices`	Il numero massimo di `SearchDevices` richieste che puoi inviare al secondo in questo account per regione.	5	Sì
Frequenza di burst delle richieste `SearchDevices`	Il numero massimo di `SearchDevices` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No

Risorsa	Descrizione	Limiti	Regolabile
Frequenza delle richieste `CreateJob`	Il numero massimo di `CreateJob` richieste che puoi inviare al secondo in questo account per regione.	1	Sì
Frequenza di burst delle richieste `CreateJob`	Il numero massimo di `CreateJob` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	5	No
Frequenza delle richieste `SearchJob`	Il numero massimo di `SearchJob` richieste che puoi inviare al secondo in questo account per regione.	5	Sì
Frequenza di burst delle richieste `SearchJob`	Il numero massimo di `SearchJob` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No

Risorsa	Descrizione	Limiti	Regolabile
Frequenza delle richieste `GetJob`	Il numero massimo di `GetJob` richieste che puoi inviare al secondo in questo account per regione.	5	Sì
Frequenza di burst delle richieste `GetJob`	Il numero massimo di `GetJob` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	25	No
Frequenza delle richieste `CancelJob`	Il numero massimo di `CancelJob` richieste che puoi inviare al secondo in questo account per regione.	2	Sì
Frequenza di burst delle richieste `CancelJob`	Il numero massimo di `CancelJob` richieste aggiuntiv e al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	5	No

Risorsa	Descrizione	Limiti	Regolabile
Numero di simultanee SV1 compiti quantistici	Il numero massimo di attività quantistiche simultanee in esecuzione sul simulatore vettoriale di stato (SV1) nella regione corrente.	100 us-est-1, 50 Stati Uniti - Ovest-1, 100 Stati Uniti-West-2, 50 eu-west-2	No
Numero di concorrenti DM1 compiti quantistici	Il numero massimo di attività quantistiche simultanee in esecuzione sul simulatore di matrice di densità (DM1) nella regione corrente.	100 us-est-1, 50 Stati Uniti - Ovest-1, 100 Stati Uniti-West-2, 50 eu-west-2	No
Numero di concorrenti TN1 compiti quantistici	Il numero massimo di attività quantistiche simultanee in esecuzione sul simulatore di rete tensoriale (TN1) nella regione corrente.	10 Stati Uniti - Est-1, 10 Stati Uniti - Ovest - 2, 5 eu-west-2,	Sì
Numero di lavori ibridi simultanei	Il numero massimo di lavori ibridi simultanei nella regione corrente.	5	Sì
Limite di runtime dei lavori ibridi	La quantità massima di tempo in giorni durante la quale un processo ibrido può essere eseguito.	5	No

Di seguito sono riportate le quote predefinite delle istanze di calcolo classiche per Hybrid Jobs. Per aumentare queste quote, contatta AWS Support. Inoltre, le regioni disponibili sono specificate per ogni istanza.

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c4.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c4.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No
Numero massimo di istanze di ml.c4.2xlarge per	Il numero massimo di istanze di tipo ml.c4.2xlarge consentit	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
lavori ibridi	o per tutti i lavori ibridi Amazon Braket in questo account e regione.							
Numero massimo di istanze di ml.c4.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c4.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c4.8xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c4.8xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	1	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.9xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.9xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	1	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.18xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.18xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.9xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.9xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.18xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.18xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.8xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.8xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.12xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.12xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.16xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.16xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	2	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.10› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.10› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.16› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.16› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.large per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.12› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.12› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.24› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.24› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p2.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p2.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p2.8xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p2.8xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p2.16xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p2.16xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p3.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p3.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p4d.24xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p4d.24xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p3dn.24xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p3dn.24xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p3.8xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p3.8xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	Sì	No

Risorsa	Descrizione	Limiti	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p3.16xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p3.16xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	Sì	No

Richiesta di aggiornamenti dei limiti

Se ricevi un' ServiceQuotaExceeded eccezione per un tipo di istanza e non disponi di un numero sufficiente di istanze disponibili, puoi richiedere un aumento del limite dalla pagina Service Quotas del AWS console e cerca Amazon Braket sotto AWS Servizi.

> **Note**
>
> Se il tuo lavoro ibrido non è in grado di fornire la capacità di calcolo ML richiesta, utilizza un'altra regione. Inoltre, se non vedi un'istanza nella tabella, non è disponibile per Hybrid Jobs.

Quote e limiti aggiuntivi

- L'azione dell'attività quantistica di Amazon Braket ha una dimensione limitata a 3 MB.
- Il numero massimo di scatti per attività consentito SV1, DM1e Rigetti i dispositivi sono 50.000.
- Il numero massimo di scatti per attività consentito TN1 è 1000.
- In IonQdei dispositivi Aria-1 e Aria-2, il massimo è 5.000 scatti per operazione. I limiti di gate sono fino a 950 porte da 1 qubit e 650 porte da 2 qubit per attività su richiesta. Questi limiti sono regolabili tramite le quote di servizio.
- In IonQdel dispositivo Forte il massimo è 10.000 scatti per operazione.
- In QuEradel dispositivo Aquila, il massimo è 1.000 colpi per operazione.
- In IQMdel dispositivo Garnet, il massimo è 20.000 scatti per operazione.
- In TN1 e il QPU dispositivi, gli scatti per operazione devono essere > 0.

Cronologia dei documenti

La tabella seguente descrive la documentazione per questa versione di Amazon Braket.

- API versione: 28 aprile 2022
- Più recente API Aggiornamento di riferimento: 25 settembre 2023
- Ultimo aggiornamento della documentazione: 29 agosto 2024

Modifica	Descrizione	Data
IonQ Harmony ritiro del dispositivo	Supporto rimosso per IonQ Harmony dispositivo.	29 agosto 2024
Nuovo dispositivo Rigetti Ankaa-2	È stato aggiunto il supporto per Rigetti Ankaa-2 dispositivo. Un dispositivo da 84 quibit che utilizza una tecnologia multi-chip scalabile.	26 agosto 2024
Riorganizzazione della guida per gli sviluppatori	La nuova guida per sviluppatori riprende il percorso esistente del cliente Build, Test, Run e guida gli utenti lungo questo percorso con Amazon Braket.	23 agosto 2024
OQC Lucy ritiro del dispositivo	Supporto rimosso per OQC Lucy dispositivo.	28 giugno 2024
Nuovo dispositivo IQM Garnet e regione Europe North 1	È stato aggiunto il supporto per il dispositivo IQMGarnet. Un dispositivo da 20 qubit con topologia a reticolo quadrato. Expanded Braket ha supportato le regioni dell'Euro	22 maggio 2024

	pa settentrionale 1 (Stoccolma).	
Rilasciata la desintonizzazione locale	Le funzionalità sperimentali ora includono la funzionalità di detoning locale di's Aquila. QuEra QPU	11 aprile 2024
Rilasciato Notebook Inactivity Manager	Quando crei un'istanza del notebook, abilita il gestore dell'inattività e imposta un periodo di inattività per ripristinare automaticamente l'istanza del notebook Braket.	27 marzo 2024
Rielaborazione del sommario	Ha riorganizzato il sommario di Amazon Braket per conformarsi al AWS definisce i requisiti e migliora il flusso di contenuti per l'esperienza del cliente.	12 dicembre 2023
Braket rilasciato direttamente	È stato aggiunto il supporto per le funzionalità di Braket Direct, tra cui: • Lavorare con le prenotazioni • Ottenere la consulenza di un esperto • Esplora le funzionalità sperimentali	27 novembre 2023
Aggiornato Crea un'istanza di notebook Amazon Braket	È stata aggiornata la documentazione per aggiungere informazioni per creare un'istanza notebook per clienti Amazon Braket nuovi ed esistenti.	27 novembre 2023

Aggiornato Porta il tuo contenitore () BYOC	È stata aggiornata la documentazione per aggiungere informazioni su quandoBYOC, sulla ricetta e sull'esecuzione di BYOC Braket Hybrid Jobs sul contenitore.	18 ottobre 2023
Rilasciato Hybrid Job Decorator	Esecuzione del codice locale come processo ibridoPagina aggiunta. Contiene esempi: • Crea un lavoro ibrido dal codice Python locale • Installa pacchetti Python e codice sorgente aggiuntivi • Salva e carica i dati in un'istanza di lavoro ibrida • Le migliori pratiche per i decoratori di lavori ibridi	16 ottobre 2023
Aggiunta la visibilità della coda	È stata aggiornata la documentazione della Developer's Guide per includerla queue depth e queue position. È stata aggiornata la API documentazione per rifletter e le nuove API modifiche relative alla visibilità delle code.	25 settembre 2023

Standardizza la denominazione nella documentazione	È stata aggiornata la documentazione per modificare qualsiasi istanza di «job» in «hybrid job» e «task» in «quantum task»	11 settembre 2023
Nuovo dispositivo IonQ Aria 2	È stato aggiunto il supporto per IonQ Aria 2 dispositivo	8 settembre 2023
Native Gates aggiornati	È stata aggiornata la documentazione per aggiungere informazioni sull'accesso programmatico ai gate nativi di Rigetti.	16 agosto 2023
Xanadu partenza	Aggiornata la documentazione per rimuovere tutto Xanadu dispositivi	2 giugno 2023
Nuovo dispositivo IonQ Aria	È stato aggiunto il supporto per IonQ Aria dispositivo	16 maggio 2023
Ritirato Rigetti dispositivo	Supporto interrotto per Rigetti Aspen-M-2	2 maggio 2023
Informazioni aggiornate AmazonBraketFullAccesssulla politica	È stato aggiornato lo script che definisce il contenuto della AmazonBraketFullAccesspolicy per includere le GetMetricData azioni servicequotas: GetServiceQuota e cloudwatch:, nonché informazioni sulle limitazioni rispetto alle quote.	19 aprile 2023

Lancio di Guided Journeys	È stata modificata la documentazione per rifletter e il metodo più aggiornato e semplificato per l'onboarding di Braket.	5 aprile 2023
Nuovo dispositivo Rigetti Aspen-M-3	È stato aggiunto il supporto per Rigetti Aspen-M-3 dispositi vo	17 gennaio 2023
Nuova funzione di gradiente aggiuntivo	Sono state aggiunte informazi oni sulla funzione di sfumatura aggiuntiva offerta da SV1	7 dicembre 2022
Nuova funzionalità di libreria di algoritmi	Sono state aggiunte informazi oni sulla libreria di algoritmi Braket, che fornisce un catalogo di algoritmi quantistici predefiniti	28 novembre 2022
D-Wave partenza	È stata aggiornata la documentazione per consentir e la rimozione di tutti D-Wave dispositivi	17 novembre 2022
Nuovo dispositivo QuEra Aquila	È stato aggiunto il supporto per QuEra Aquila dispositivo	31 ottobre 2022
Supporto per Braket Pulse	È stato aggiunto il supporto per Braket Pulse, che consente di utilizzare il controllo degli impulsi su Rigetti e OQC dispositivi	20 ottobre 2022
Support per porte native IonQ	È stato aggiunto il supporto per il set di porte nativo offerto dal dispositivo IonQ	13 settembre 2022

Nuove quote di istanze	Sono state aggiornate le quote predefinite delle istanze di calcolo classiche associate a Hybrid Jobs	22 agosto 2022
Nuova dashboard di servizio	Schermate della console aggiornate per includere la dashboard del servizio	17 agosto 2022
Nuovo dispositivo Rigetti Aspen-M-2	È stato aggiunto il supporto per Rigetti Aspen-M-2 dispositivo	12 agosto 2022
Nuove QASM funzionalità Open	Aggiunto il supporto QASM delle funzionalità Open per i simulatori locali (braket_sv e braket_dm)	4 agosto 2022
Nuove procedure di tracciamento dei costi	È stato aggiunto come ottenere stime dei costi massimi quasi in tempo reale per simulatori e carichi di lavoro hardware	18 luglio 2022
Novità Xanadu Borealis dispositivo	È stato aggiunto il supporto per Xanadu Borealis dispositivo	2 giugno 2022
Nuove procedure di semplificazione dell'onboarding	Sono state aggiunte informazioni su come funzionano le nuove e semplificate procedure di onboarding	16 maggio 2022
Nuovo dispositivo D-Wave Advantage_system6.1	È stato aggiunto il supporto per D-Wave Advantage _system6.1 dispositivo	12 maggio 2022

Support per simulatori integrati	È stato aggiunto come eseguire simulazioni integrate con lavori ibridi e come utilizzare il simulatore di fulmini PennyLane	4 maggio 2022
AmazonBraketFullAccess - Politica di accesso completo per Amazon Braket	Aggiunto s3: ListAllMyBuckets autorizzazioni per consentir e agli utenti di visualizzare e ispezionare i bucket creati e utilizzati per Amazon Braket	31 marzo 2022
Support per Open QASM	Aggiunto il supporto Open QASM 3.0 per dispositivi e simulatori quantistici basati su gate	7 marzo 2022
Nuovo fornitore di hardware quantistico, Oxford Quantum Circuits e nuova regione, eu-west-2	Aggiunto il supporto per OQC e eu-west-2	28 febbraio 2022
Novità Rigetti dispositivo	È stato aggiunto il supporto per Rigetti Aspen M-1	15 febbraio 2022
Nuovi limiti di risorse	È stato aumentato il numero massimo di connessioni simultanee DM1 e SV1 attività da 55 a 100	5 gennaio 2022
Novità Rigetti dispositivo	È stato aggiunto il supporto per Rigetti Aspen-11	20 dicembre 2021
Ritirato Rigetti dispositivo	Supporto interrotto per Rigetti Aspen-10 dispositivo	20 dicembre 2021

Nuovo tipo di risultato	Tipo di risultato a matrice a densità ridotta supportato dal simulatore di matrice a densità locale e DM1 dispositivi	20 dicembre 2021
Descrizione aggiornata della politica	Amazon Braket ha aggiornato il ruolo ARN per includere il `servicerole/` percorso. Per informazioni sugli aggiornamenti delle policy, consulta gli aggiornamenti di Amazon Braket su AWStabella delle politiche gestite.	29 novembre 2021
Offerte di lavoro Amazon Braket	Guida per l'utente per Amazon Braket Hybrid Jobs e API aggiunto	29 novembre 2021
Novità Rigetti dispositivo	È stato aggiunto il supporto per Rigetti Aspen-10	20 novembre 2021
Ritirato D-Wave dispositivo	Supporto interrotto per D-Wave QPU, Advantage _system1	4 novembre 2021
Novità D-Wave dispositivo	È stato aggiunto il supporto per un ulteriore D-Wave QPU, Advantage_system4	5 ottobre 2021
Nuovi simulatori di rumore	È stato aggiunto il supporto per un simulatore di matrice di densità (DM1), che può simulare circuiti fino a 17 qubits e un simulatore di rumore locale braket_dm	25 maggio 2021

PennyLane supporto	È stato aggiunto il supporto per PennyLane Amazon Braket	8 dicembre 2020
Nuovo simulatore	È stato aggiunto il supporto per un simulatore di rete Tensor (TN1), che consente circuiti più grandi	8 dicembre 2020
Suddivisione in batch delle attività	Braket supporta il raggruppamento delle attività dei clienti	24 novembre 2020
Manuale qubit assegnazione	Braket supporta il manuale qubit allocazione sul Rigetti dispositivo	24 novembre 2020
Quote modificabili	Braket supporta quote regolabili in modalità self-service per le risorse operative	30 ottobre 2020
Support per PrivateLink	Puoi configurare VPC endpoint privati per i tuoi lavori in Braket	30 ottobre 2020
Supporto per i tag	Supporti Braket APItag basati per la risorsa quantum-task	30 ottobre 2020
Novità D-Wave dispositivo	È stato aggiunto il supporto per un ulteriore D-Wave QPU, Advantage_system1	29 settembre 2020
Rilascio iniziale	Versione iniziale della documentazione di Amazon Braket	12 agosto 2020

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.