Archiviazione dei dati in Amazon RDS for MySQL, Amazon RDS per MariaDB e Aurora MySQL-Compatible

AWS Linee guida prescrittive

AWS Linee guida prescrittive: Archiviazione dei dati in Amazon RDS for MySQL, Amazon RDS per MariaDB e Aurora MySQL-Compatible

Table of Contents

Archiviazione dei dati in Amazon RDS for MySQL, Amazon RDS for MariaDB e Aurora MySQL compatibile

Amazon Web Services di Shyam Sunder Rakhecha, Abhishek Karmakar, Oliver Francis e Saumya Singh ()AWS

Aprile 2025 (storia del documento)

La necessità di archiviare i dati storici può derivare da diversi casi d'uso. L'applicazione potrebbe essere stata progettata senza funzionalità di archiviazione e la crescita dell'azienda nel tempo potrebbe comportare grandi quantità di dati storici. Ciò porta inevitabilmente a un peggioramento delle prestazioni. È inoltre possibile conservare i dati storici a causa dei requisiti di conformità all'interno dell'organizzazione.

Questa guida spiega come archiviare i dati storici in Amazon Simple Storage Service (Amazon S3) con un impatto minimo sull'applicazione e recuperare le informazioni archiviate quando ne hai bisogno.

Panoramica di

Questa guida illustra diversi approcci per l'archiviazione dei dati storici da tabelle di grandi dimensioni in Amazon Relational Database Service (Amazon RDS) per MySQL, Amazon RDS per MariaDB e Amazon Aurora MySQL Compatible Edition sul cloud Amazon Web Services (AWS). In questa guida, imparerai come archiviare sia i dati delle tabelle partizionate che i dati non partizionati e che risiedono in tabelle di grandi dimensioni. È possibile implementare gli approcci presentati nella guida per ridurre le dimensioni dei dati in tempo reale conservando al contempo importanti dati storici per ulteriori analisi.

L'archiviazione regolare dei dati delle tabelle comporta un set più ridotto di dati in tempo reale nelle tabelle, il che porta a letture e scritture più veloci e migliora le prestazioni dell'applicazione. L'archiviazione regolare dei dati rientra nei pilastri dell'eccellenza operativa e dell'efficienza delle prestazioni del Well-Architected Framework. Quando sposti dati più vecchi su Amazon Simple Storage Service (Amazon S3) e pulisci i dati archiviati nell'istanza Amazon RDS o nel cluster Aurora compatibile con MySQL, puoi risparmiare sui costi di storage. Ciò si adatta al pilastro dell'ottimizzazione dei costi e ti aiuta a evitare costi inutili su AWS.

Obiettivi aziendali specifici

Questa guida si concentra sui seguenti risultati aziendali:

- Esperienza utente migliorata
- Requisiti di conformità dei dati soddisfatti
- Costi di storage ridotti
- Dati organizzati

Esperienza utente migliorata

I database che conservano dati storici possono avere prestazioni lente a causa di tabelle e indici di grandi dimensioni. Quando si archiviano i dati storici, si riducono le tabelle e gli indici. Ciò ha un impatto positivo diretto sulle operazioni API rivolte ai clienti che interagiscono con il database.

Requisiti di conformità dei dati soddisfatti

Settori come i servizi finanziari, le organizzazioni del settore pubblico e l'assistenza sanitaria hanno requisiti di archiviazione rigorosi. Archiviando i dati delle applicazioni che risiedono nel tuo database compatibile con Amazon RDS for MySQL, Amazon RDS for MariaDB o Aurora MySQL in Amazon S3, puoi soddisfare i requisiti per la conformità regolamentata, tra cui:

- Payment Card Industry Data Security Standard (PCI DSS)
- Health Insurance Portability and Accountability Act (HIPAA) e Health Information Technology for Economic and Clinical Health (HITECH)
- Federal Risk and Authorization Management Program (FedRAMP)
- Regolamento generale sulla protezione dei dati (GDPR)
- Standard federali per l'elaborazione delle informazioni (FIPS) 140-2
- Istituto nazionale di standard e tecnologia (NIST) 800—171

Costi di storage ridotti

La conservazione dei dati in Amazon RDS aumenta i costi di storage e richiede IOPS più elevati. Se si confronta il costo di storage per GB al mese per Amazon RDS for GP2 MySQL Multi-AZ con quello

di Amazon S3 Glacier nella regione AWS, il costo di storage di S3 Glacier è circa 57 volte inferiore `us-east-1` a quello di Amazon RDS.

Dati organizzati

È consigliabile conservare nel database i dati informativi a cui l'applicazione accederà frequentemente. Tuttavia, le applicazioni generano una grande quantità di dati che non è richiesta molto spesso o diventa obsoleta. Questi record possono essere archiviati e conservati sul posto, il che è conveniente e non influisce sulle prestazioni delle applicazioni.

Archiviazione dei dati in tabelle partizionate

MySQL supporta il partizionamento per il motore di archiviazione InnoDB ed è possibile utilizzare questa funzionalità per partizionare tabelle di grandi dimensioni. Le partizioni all'interno della tabella vengono archiviate come tabelle fisiche separate, sebbene l'SQL che opera sulla tabella partizionata legga l'intera tabella. In questo modo è possibile rimuovere le partizioni non necessarie dalla tabella senza eseguire row-by-row eliminazioni, in modo da poter archiviare le righe cronologiche del database.

Considerate il seguente codice di esempio. `TABLE orders`esiste all'interno dello `orderprocessing` schema. I suoi dati storici sono presenti nella partizione `phistorica` l, che contiene dati relativi al 2021 e precedenti. Nella stessa tabella, gli hot data a livello di applicazione sono presenti nelle partizioni live per ogni mese del 2022. Per archiviare i dati nella partizione`phistorical`, è possibile creare un archivio `table orders_2021_and_older` con la stessa struttura nello schema. `archive` È quindi possibile utilizzare MySQL EXCHANGE PARTITION per spostare la `phistorical` partizione in quella tabella. Si noti che la tabella di archivio non è partizionata. Dopo l'archiviazione, puoi verificare i tuoi dati e spostarli su Amazon S3.

```
CREATE TABLE orders (
      orderid bigint NOT NULL AUTO_INCREMENT,
      customerid bigint DEFAULT NULL,
      ............
      ............
      order_date date NOT NULL,
      PRIMARY KEY (`orderid`,`order_date`))
    PARTITION BY RANGE (TO_DAYS(order_date)) (
          PARTITION pstart   VALUES LESS THAN (0),
          PARTITION phistorical VALUES LESS THAN (TO_DAYS('2022-01-01')),
          PARTITION p2022JAN VALUES LESS THAN (TO_DAYS('2022-02-01')),
          PARTITION p2022FEB VALUES LESS THAN (TO_DAYS('2022-03-01')),
          PARTITION p2022MAR VALUES LESS THAN (TO_DAYS('2022-04-01')),
          PARTITION p2022APR VALUES LESS THAN (TO_DAYS('2022-05-01')),
          PARTITION p2022MAY VALUES LESS THAN (TO_DAYS('2022-06-01')),
          PARTITION p2022JUN VALUES LESS THAN (TO_DAYS('2022-07-01')),
          PARTITION p2022JUL VALUES LESS THAN (TO_DAYS('2022-08-01')),
          PARTITION p2022AUG VALUES LESS THAN (TO_DAYS('2022-09-01')),
          PARTITION p2022SEP VALUES LESS THAN (TO_DAYS('2022-10-01')),
          PARTITION p2022OCT VALUES LESS THAN (TO_DAYS('2022-11-01')),
          PARTITION p2022NOV VALUES LESS THAN (TO_DAYS('2022-12-01')),
          PARTITION p2022DEC VALUES LESS THAN (TO_DAYS('2023-01-01')),
```

```
          PARTITION pfuture       VALUES LESS THAN MAXVALUE
      );
CREATE TABLE orders_2021_and_older (
        orderid bigint NOT NULL AUTO_INCREMENT,
        customerid bigint DEFAULT NULL,
        ............
        ............
        order_date date NOT NULL,
        PRIMARY KEY (`orderid`,`order_date`));
mysql> alter table orderprocessing.orders exchange partition phistorical with table
 archive.orders_2021_and_older;
  Query OK, 0 rows affected (0.33 sec)
```

Quando utilizzi la `EXCHANGE PARTITION` funzionalità per archiviare dati storici, ti consigliamo le seguenti best practice:

- Create uno schema separato per l'archiviazione dei dati di archivio nell'applicazione. Questo schema conterrà tabelle di archivio che conterranno i dati archiviati. Una tabella di archivio nello schema di archiviazione dovrebbe avere la stessa struttura della tabella live, compresi gli indici e la chiave primaria. Tuttavia, la tabella di archiviazione di destinazione non può essere una tabella partizionata. Lo scambio di partizioni tra due tabelle partizionate non è consentito in MySQL.
- Segui una convenzione di denominazione per la tabella di archivio che ti aiuti a identificare i dati storici in essa contenuti. Ciò è utile quando si eseguono attività di controllo o lavori di progettazione che trasferiscono questi dati su Amazon S3.
- Esegui l'istruzione DDL (`EXCHANGE PARTITION`Data Definition Language) in una finestra di inattività quando non c'è traffico in entrata verso il tuo writer Aurora compatibile con MySQL, Amazon RDS for MySQL o Amazon RDS for MariaDB.

  Potrebbe essere possibile eseguirlo durante le finestre a traffico limitato nell'applicazione o nel microservizio. `EXCHANGE PARTITION` Tuttavia, non dovrebbero esserci scritture e nessuna o pochissime selezioni sulla tabella partizionata. Le query di selezione esistenti a esecuzione prolungata possono causare l'attesa della `EXCHANGE PARTITION` DDL, causando un conflitto di risorse nel database. Script di progettazione che verificano che tutte queste condizioni siano soddisfatte prima dell'esecuzione sul sistema. `EXCHANGE PARTITION`

Se la progettazione dell'applicazione è in grado di supportare dati partizionati e al momento disponi di una tabella non partizionata, valuta la possibilità di spostare i dati in tabelle partizionate per supportare l'archiviazione dei dati. Per ulteriori informazioni, consulta la documentazione di MySQL.

Archiviazione dei dati da tabelle non partizionate

Nelle tabelle del database in cui il partizionamento non è possibile, è possibile utilizzare lo strumento Percona Toolkit pt-archiver per archiviare i dati della tabella in un'altra tabella nel database MySQL.

Lo strumento pt-archiver viene utilizzato per archiviare i record da tabelle di grandi dimensioni in altre tabelle o file. È uno read/write strumento, il che significa che elimina i dati dalla tabella di origine dopo averli archiviati, quindi non è necessario gestire l'eliminazione dei dati di origine separatamente. Lo scopo principale di questo script è archiviare i vecchi dati dalla tabella senza influire sul carico delle query OLTP (online transaction processing) esistente (vedi Appendice I) e inserire i dati in un'altra tabella sullo stesso server o su un server diverso.

Puoi scaricare Percona Toolkit e installarlo sul tuo computer locale o sull'istanza Amazon Elastic Compute Cloud (Amazon EC2) da cui ti connetti al database. Per eseguire lo strumento pt-archiver, usa la seguente sintassi.

```
pt-archiver --source h=<HOST>,D=<DATABASE>,t=<TABLE>,u=<USER>,p=<PASSWORD> --dest
 h=<HOST>,D=<DATABASE>,t=<TABLE> --where ""1=1"" --statistics
```

Sostituisci`HOST`, `DATABASE`, `TABLE`, e `USER` con i dettagli e le credenziali del database di origine e destinazione.

Puoi anche usare AWS Batch per creare e pianificare questo lavoro per le tue tabelle.

Quando usi lo strumento pt-archiver per archiviare i dati della tabella, considera quanto segue:

- Avere una chiave primaria nella tabella sorgente migliorerà le prestazioni di questo strumento. Se la tabella non ha una chiave primaria, è possibile creare un indice su una colonna unica, che aiuterà pt-archiver a esaminare tutte le righe della tabella e ad archiviarle.
- Per impostazione predefinita, pt-archiver elimina i dati dopo aver archiviato la tabella. Prima di eseguirla sul server di produzione, assicuratevi di testare i vostri lavori di archiviazione con. `--dry-run` In alternativa, è possibile utilizzare l'`--no-delete`opzione.
- Lo strumento pt-archiver regola la velocità di archiviazione in base al carico sul sistema (vedi Appendice II). Con carichi più elevati, ci si può aspettare prestazioni di archiviazione più lente.

Dopo aver eseguito pt-archiver, i dati archiviati dovrebbero trovarsi nella tabella corrispondente nello schema di archivio. Da lì, puoi spostarlo su Amazon S3.

Spostamento dei dati delle tabelle archiviate su Amazon S3

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) è l'obiettivo naturale per i dati archiviati. Offre una durabilità del 99,76% ed è meno costoso dello storage di database.

Inoltre, Amazon S3 dispone di classi di storage integrate il cui prezzo è basato sul modello di recupero. Hai la possibilità di trasferire gli oggetti S3 scaricati in un livello di storage più economico in base alla frequenza di recupero dei dati. Per ulteriori informazioni sulle classi di storage e sui prezzi, consulta la documentazione di Amazon S3.

Per le applicazioni che utilizzano flotte di istanze MySQL, l'offload in Amazon S3 significherebbe risparmiare sui dati che soddisfano i seguenti criteri:

- Deve essere archiviato dal database per aumentare l'efficienza
- Non è necessario immediatamente o è necessario con parsimonia per qualsiasi processo aziendale
- Deve essere conservato a lungo termine a causa dei requisiti di audit

È possibile archiviare i dati MySQL nei seguenti modi:

- Esporta dati da un cluster Amazon Aurora attivo
- Esporta i dati utilizzando `SELECT INTO OUTFILE S3`
- Esporta dati utilizzando AWS Glue

Esportazione di dati da un cluster Aurora DB live

Puoi esportare dati da un cluster Amazon Aurora DB live in un bucket S3. Il processo di esportazione viene eseguito in background e non ha ripercussioni sulle prestazioni dell'istanza del cluster di database attivo.

Per impostazione predefinita, vengono esportati tutti i dati nel cluster database. Tuttavia, è possibile scegliere di esportare set specifici di database, schemi o tabelle. Aurora clona il cluster DB, estrae i dati dal clone e archivia i dati in un bucket S3. I dati vengono archiviati in un formato Apache Parquet compresso e coerente. I singoli file Parquet hanno in genere una dimensione compresa tra 1 e 10 MB.

Per ulteriori informazioni, consulta la documentazione di Aurora.

Esporta i dati utilizzando SELECT INTO OUTFILE S3

Per copiare i dati dal MySQL-Compatible DB Aurora direttamente in Amazon S3, puoi utilizzare l'istruzione. `SELECT INTO OUTFILE S3` Questa istruzione SQL può essere eseguita sulla tabella e il numero richiesto di righe può essere scaricato come file con valori separati da virgole (CSV) con una dimensione massima di 6 GB. Quando viene superata la soglia dei 6 GB, vengono creati più file.csv.

Affinché l'esportazione funzioni, configura quanto segue:

- Ruoli e policy di AWS Identity and Access Management (IAM)
- Autorizzazioni del database concesse all'utente per eseguire il comando
- La posizione Amazon S3 per l'offload

Per ulteriori informazioni, consulta la documentazione di Amazon Aurora.

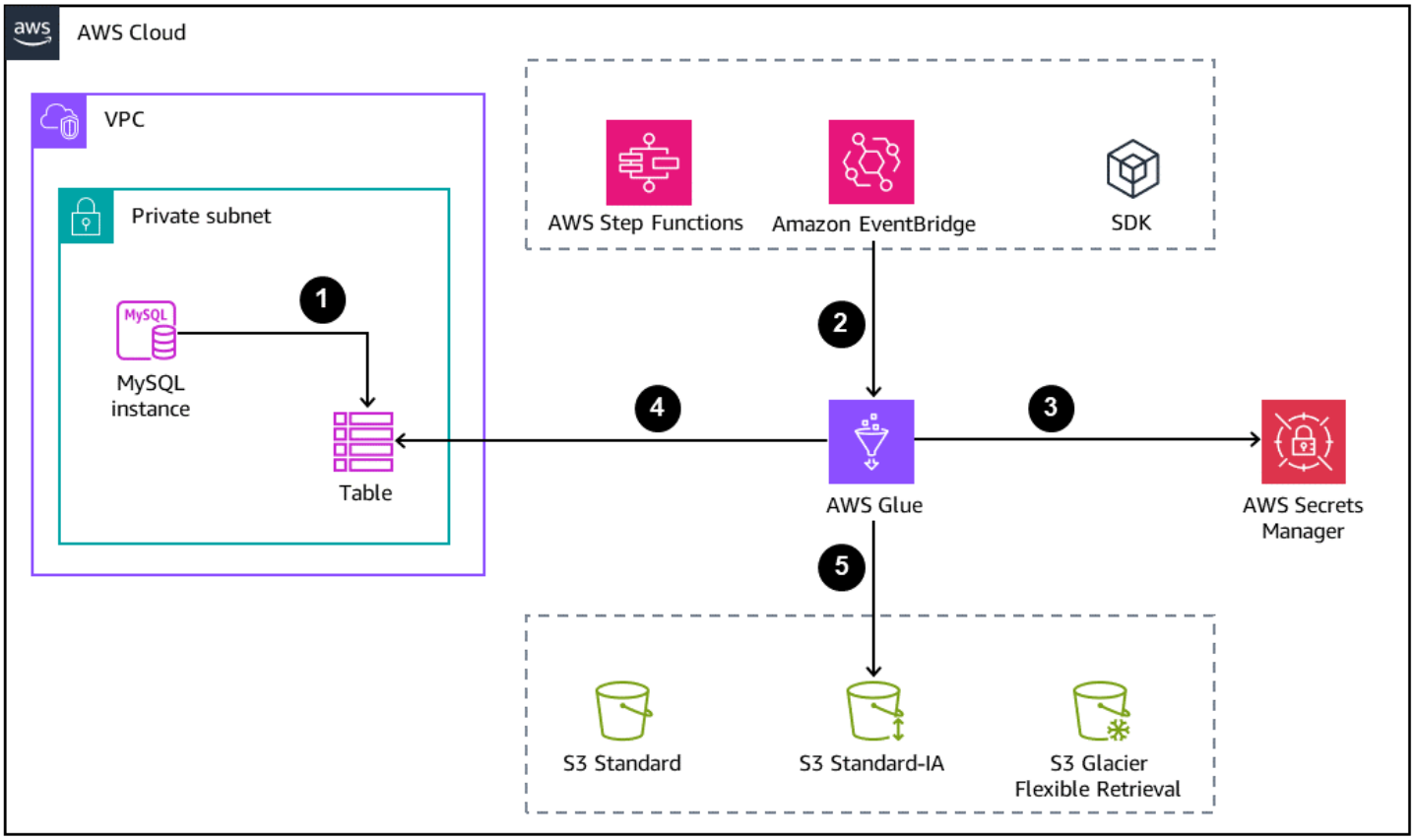
Nota: per trasferire gli oggetti S3 in livelli di storage convenienti, consigliamo di configurare le regole del ciclo di vita di Amazon S3 nel bucket S3 in cui vengono esportati i dati.

Esporta dati utilizzando AWS Glue

Puoi archiviare i dati MySQL in Amazon S3 utilizzando AWS Glue, un servizio di analisi serverless per scenari di big data. AWS Glue è basato su Apache Spark, un framework di cluster-computing distribuito ampiamente utilizzato che supporta molte fonti di database.

L'offload dei dati archiviati dal database ad Amazon S3 può essere eseguito con poche righe di codice in un job AWS Glue. Il principale vantaggio offerto da AWS Glue è la scalabilità orizzontale e un modello pay-as-you-go, che offre efficienza operativa e ottimizzazione dei costi.

Il diagramma seguente mostra un'architettura di base per l'archiviazione dei database.



1. Il database MySQL crea l'archivio o la tabella di backup da scaricare in Amazon S3.
2. Un job AWS Glue viene avviato con uno dei seguenti approcci:
   - In modo sincrono come fase all'interno di una macchina a stati AWS Step Functions
   - In modo asincrono tramite un evento Amazon EventBridge
   - Tramite una richiesta manuale tramite l'interfaccia a riga di comando di AWS o un SDK AWS
3. Le credenziali DB vengono recuperate da AWS Secrets Manager.
4. Il job AWS Glue utilizza una connessione Java Database Connectivity (JDBC) per accedere al database e leggere la tabella.
5. AWS Glue scrive i dati in Amazon S3 in formato Parquet, un formato di dati aperto, colonnare e poco ingombrante.

Configurazione di AWS Glue Job

Per funzionare come previsto, il job AWS Glue richiede i seguenti componenti e configurazioni:

- Connessioni AWS Glue: si tratta di un oggetto AWS Glue Data Catalog che colleghi al job per accedere al database. Un job può avere più connessioni per effettuare chiamate a più database. Le connessioni contengono le credenziali del database archiviate in modo sicuro.
- GlueContext— Questo è un wrapper personalizzato sulla GlueContext classe SparkContextLa classe fornisce operazioni API di ordine superiore per interagire con Amazon S3 e le fonti del database. Consente l'integrazione con Data Catalog. Inoltre elimina la necessità di affidarsi ai driver per la connessione al database, che viene gestita all'interno della connessione Glue. Inoltre, la GlueContext classe fornisce modi per gestire le operazioni dell'API Amazon S3, cosa non possibile con la classe originale SparkContext .
- Politiche e ruoli IAM: poiché AWS Glue interagisce con altri servizi AWS, è necessario configurare ruoli appropriati con il minimo privilegio richiesto. I servizi che richiedono le autorizzazioni appropriate per interagire con AWS Glue includono:
  - Simple Storage Service (Amazon S3)
  - AWS Secrets Manager
  - AWS Key Management Service (AWS KMS)

Best practice

- Per leggere intere tabelle con un numero elevato di righe da scaricare, consigliamo di utilizzare l'endpoint di replica di lettura per aumentare la velocità di lettura senza compromettere le prestazioni dell'istanza di scrittura principale.
- Per raggiungere l'efficienza nel numero di nodi utilizzati per l'elaborazione del lavoro, attiva la scalabilità automatica in AWS Glue 3.0.
- Se il bucket S3 fa parte dell'architettura del data lake, consigliamo di scaricare i dati organizzandoli in partizioni fisiche. Lo schema di partizione deve essere basato sui modelli di accesso. Il partizionamento basato sui valori di data è una delle pratiche più consigliate.
- Il salvataggio dei dati in formati aperti come Parquet o Optimized Row Columnar (ORC) aiuta a renderli disponibili ad altri servizi di analisi come Amazon Athena e Amazon Redshift.
- Per ottimizzare la lettura dei dati scaricati da altri servizi distribuiti, è necessario controllare il numero di file di output. È quasi sempre utile disporre di un numero inferiore di file di grandi dimensioni anziché di un numero elevato di file di piccole dimensioni. Spark dispone di file di configurazione e metodi integrati per controllare la generazione di file di parti.

- I dati archiviati per definizione sono set di dati a cui si accede spesso. Per raggiungere l'efficienza in termini di costi per lo storage, la classe Amazon S3 dovrebbe essere trasferita a livelli meno costosi. Ciò può essere fatto utilizzando due approcci:
  - Transizione sincrona del livello durante l'offload: se sai in anticipo che la transizione dei dati scaricati deve essere effettuata come parte del processo, puoi utilizzare il meccanismo GlueContext transition_s3_path all'interno dello stesso job AWS Glue che scrive i dati in Amazon S3.
  - Transizione asincrona con S3 Lifecycle: configura le regole del ciclo di vita di S3 con parametri appropriati per la transizione e la scadenza della classe di storage Amazon S3. Una volta configurato nel bucket, persisterà per sempre.
- Crea e configura una sottorete con un intervallo di indirizzi IP sufficiente all'interno del cloud privato virtuale (VPC) in cui viene distribuito il database. Ciò eviterà errori di lavoro di AWS Glue causati da un numero insufficiente di indirizzi di rete quando è configurato un numero elevato di unità di elaborazione dati (DPU).

Accesso ai dati archiviati su Amazon S3

Amazon S3 fornisce una serie di strumenti per leggere il contenuto dei dati. Tuttavia, a seconda della classe di storage, potrebbero essere necessarie alcune fasi di preelaborazione. Questa sezione include quanto segue:

- Lettura di oggetti S3 archiviati con la classe di storage Standard utilizzando AWS Glue
- Lettura di un oggetto S3 archiviato con le classi di storage S3 Glacier utilizzando S3 Batch Operations
- Best practice

Lettura di oggetti S3 archiviati con la classe di storage Standard

Usare AWS Glue

I dati trasferiti da MySQL ad Amazon S3 mantengono la stessa rigidità strutturale e coerenza tipiche di un sistema di gestione di database relazionali (RDBMS).

AWS Glue Crawler esegue la scansione degli oggetti S3, deduce i tipi di dati e crea i metadati delle tabelle come una tabella DDL esterna. Quando configuri il crawler job, usa Amazon S3 come origine e specifica la posizione del prefisso S3 in cui vengono creati tutti i file di dati. Nella configurazione, includi quanto segue:

- Opzioni di esecuzione del crawler
- Preferenza opzionale per il prefisso della tabella
- Database di destinazione per la creazione della tabella
- Ruoli IAM con autorizzazioni richieste

Dopo aver richiamato il job, analizzerà i dati per dedurre lo schema e conservarlo in AWS Glue Data Catalog come tabelle AWS Glue. Le tabelle AWS Glue sono essenzialmente tabelle esterne che possono essere interrogate con istruzioni SQL come una normale tabella di database utilizzando servizi di analisi come Amazon Athena, AmazonRedshift Spectrum e Apache Hive su Amazon EMR. Per ulteriori informazioni sul crawler, consulta la documentazione di AWS Glue.

Per i file.csv con un'intestazione di colonna specificata, i nomi delle colonne della tabella risultante rifletteranno gli stessi nomi di campo. Il tipo di dati viene dedotto in base ai valori dell'oggetto dati.

Per i file Parquet, lo schema viene mantenuto all'interno dei dati stessi e la tabella risultante rifletterà gli stessi nomi di campo e lo stesso tipo di dati.

In alternativa, puoi eseguire manualmente una DDL all'interno di Athena per creare la definizione della tabella con i nomi di colonna e il tipo di dati richiesti. In questo modo viene creata la definizione della tabella all'interno di Data Catalog. Per ulteriori informazioni sulla creazione di tabelle Athena, consulta la documentazione di Amazon Athena.

Nota: se la riga di intestazione non è presente nel file CSV, il crawler crea il nome del campo come generico c_0, c_1, c_2,...

Utilizzo di Amazon S3 Select

Puoi usare Amazon S3 Select per leggere gli oggetti S3 a livello di codice utilizzando espressioni SQL. L'operazione API può essere richiamata utilizzando il `select-object-content` comando AWS CLI o utilizzando un SDK come Boto3 e richiamando l'operazione da Python. `select_object_content`

Le operazioni API supportano le istruzioni SQL come parametri e possono leggere solo file di tipo JSON e Parquet. Gli output possono essere reindirizzati come file di output.

Queste operazioni vengono richiamate per ogni oggetto S3. Per più file, esegui le operazioni in modo ricorsivo.

Per ulteriori informazioni sull'esecuzione delle operazioni utilizzando l'interfaccia a riga di comando di AWS, consulta la documentazione dell'interfaccia a riga di comando di AWS. Per ulteriori informazioni sull'esecuzione di S3 Select utilizzando Python SDK Boto3, consulta la documentazione di Boto3.

Lettura di oggetti S3 archiviati con le classi di storage S3 Glacier

Le classi Amazon S3 Glacier sono classi di storage speciali con prezzi convenienti ma tempi di recupero elevati. A differenza degli oggetti S3 Standard, gli oggetti S3 Glacier non possono essere letti come tabelle AWS Glue. Per rendere disponibili i dati per le query analitiche o i report, devi prima ripristinare gli oggetti S3 Glacier. Il ripristino è un processo asincrono che avviene nel tempo e prevede un periodo di conservazione. Dopo il ripristino, gli oggetti possono essere copiati in una posizione diversa come oggetti S3 Standard. Dopo il periodo di conservazione, gli oggetti ripristinati tornano ad Amazon S3 Glacier.

Utilizzo di S3 Batch Operations

S3 Batch Operations consente operazioni batch su larga scala su Amazon S3 nell'ordine di miliardi di oggetti contenenti exabyte di dati. Amazon S3 tiene traccia dei progressi, invia notifiche e archivia un report dettagliato sul completamento di tutte le azioni, offrendo un'esperienza completamente gestita, verificabile e serverless.

S3 Batch Operations supporta l'operazione di ripristino, che avvia il ripristino degli oggetti S3 per i seguenti livelli di storage:

- Oggetti archiviati nelle classi di archiviazione S3 Glacier Flexible Retrieval o S3 Glacier Deep Archive
- Oggetti archiviati tramite la classe di storage S3 Intelligent-Tiering nei livelli di accesso di archiviazione o archiviazione profonda

L'operazione batch può essere richiamata sia a livello di codice che sulla console Amazon S3. Per l'input, richiede un file manifesto in formato.csv che contenga l'elenco degli oggetti da ripristinare.

Puoi utilizzare un report di inventario Amazon S3 come input per il lavoro in batch. Il rapporto di inventario è configurato per un bucket e può essere limitato agli oggetti con prefissi specifici. È un report automatizzato e viene generato settimanalmente o giornalmente in formato CSV, ORC o Parquet.

Per ulteriori informazioni sulla configurazione di un rapporto di inventario, consulta la documentazione di Amazon S3. Per informazioni sull'utilizzo di Boto3 per creare un job S3 Batch Operations, consulta la documentazione di Boto3.

Best practice

Consigliamo le seguenti best practice per accedere ai dati archiviati:

- Per set di dati di archiviazione di grandi dimensioni, consigliamo di creare tabelle AWS Glue sopra i dati in modo che possano essere letti utilizzando motori di query come Athena e Amazon Redshift. Sia Athena che Amazon Redshift offrono la scalabilità orizzontale delle prestazioni delle query. Utilizzano anche un pay-per-query modello che è conveniente in uno scenario di interrogazione una tantum. Inoltre, Amazon Redshift è dotato di motori Advanced Query Accelerator (AQUA), che velocizzano le prestazioni di lettura senza costi aggiuntivi.
- I dati archiviati scaricati regolarmente in Amazon S3 non devono essere archiviati come heap dump. Invece, dovrebbe essere salvato come nuova partizione. Una partizione di data separerà i

dati in dimensioni di data (ad esempio,`year=<value>/month=<value>/day=<value>`). Ciò è estremamente utile in due situazioni:

- Se le tabelle AWS Glue vengono create dai crawler di AWS Glue, queste partizioni fungono da pseudo colonne. Ciò migliora le prestazioni di lettura limitando i dati scansionati alle partizioni nell'intervallo di query.
- Questo aiuta in un'operazione di ripristino di S3 Glacier quando si ripristina solo un sottoinsieme dell'oggetto come S3 Standard.

- I crawler di AWS Glue mostrano un grande valore quando i dati archiviati salvati in Amazon S3 vengono partizionati fisicamente. Ogni volta che i dati vengono scaricati come nuova partizione di prefisso, il crawler analizza solo la nuova partizione e aggiorna i metadati per quella partizione. Se lo schema della tabella cambia, tali modifiche verranno acquisite nei metadati a livello di partizione.

Pulizia della tabella di archivio

La fase finale del processo di archiviazione consiste nella pulizia delle tabelle nello schema di archiviazione. Puoi farlo dopo aver confermato che i tuoi dati di archivio sono archiviati in modo sicuro in Amazon S3. Per evitare qualsiasi impatto sull'applicazione, consigliamo di eliminare le tabelle degli schemi di archiviazione durante un periodo di inattività pianificato o una finestra di manutenzione o durante una finestra di traffico molto basso nell'applicazione. Queste tabelle non vengono interrogate attivamente dall'applicazione e non dovrebbero essere motivo di allarme per il loro impatto sulle transazioni in corso. Tuttavia, è consigliabile eseguirla DDLs durante un periodo di inattività.

Dopo aver liberato lo storage per tabelle di schemi di archiviazione di grandi dimensioni, Amazon Aurora utilizza il ridimensionamento dinamico per aiutarti a risparmiare sui costi di storage.

Risorse

Documentazione

- Amazon Aurora
- AWS Glue
- Amazon S3
- Interrogazione dell'inventario Amazon S3 con Amazon Athena
- Configurazione dell'inventario Amazon S3
- Esecuzione di operazioni in batch su larga scala su oggetti Amazon S3
- Documentazione Boto3
- Documentazione sul partizionamento MySQL

Prezzi

- Prezzi di Amazon RDS per MySQL Multi-AZ GP2
- Prezzi di Amazon S3 Glacier

Strumenti

- Toolkit Percona
- pt-archiver
- sysbench

Post di blog

- S3 Select e S3 Glacier Select: recupero di sottoinsiemi di oggetti
- Archivia ed elimina i dati per Amazon RDS for PostgreSQL e Amazon Aurora con compatibilità PostgreSQL utilizzando pg_partman e Amazon S3

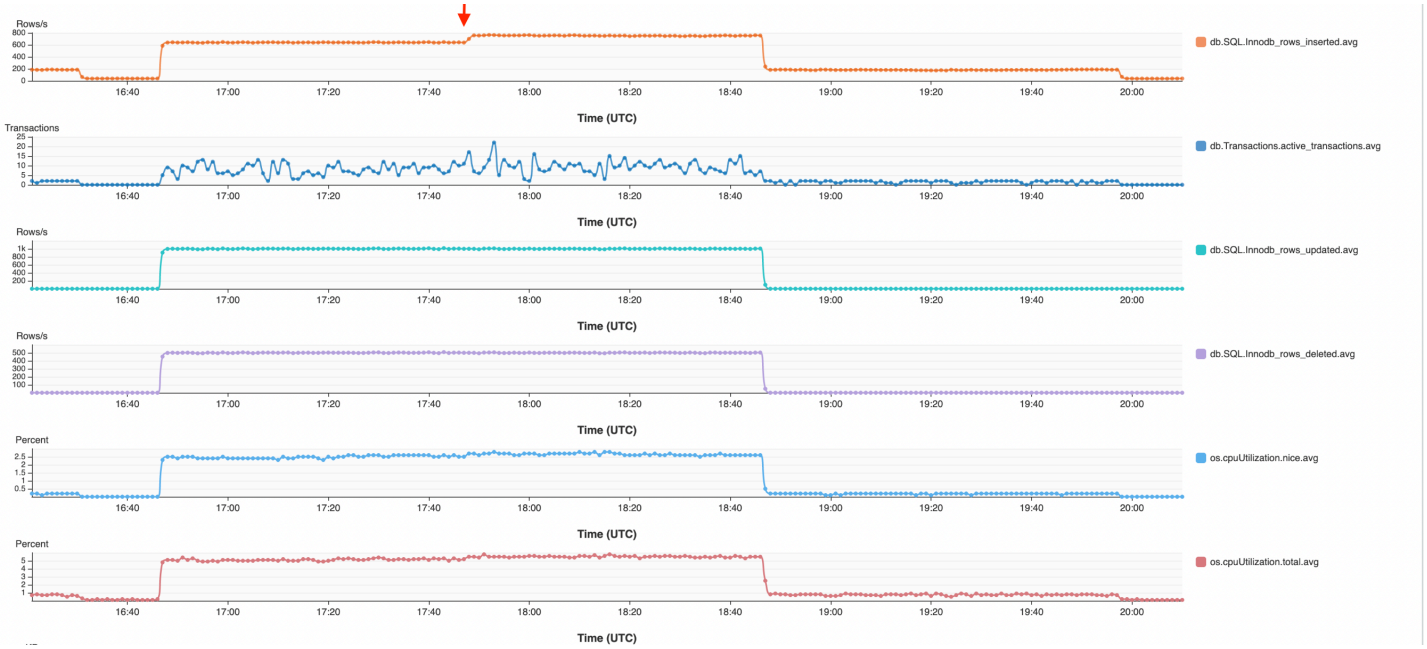
Appendice I

Tutti i test vengono eseguiti su un'istanza Amazon RDS for MySQL in esecuzione sulla classe di istanza. `db.r6g.8xlarge`

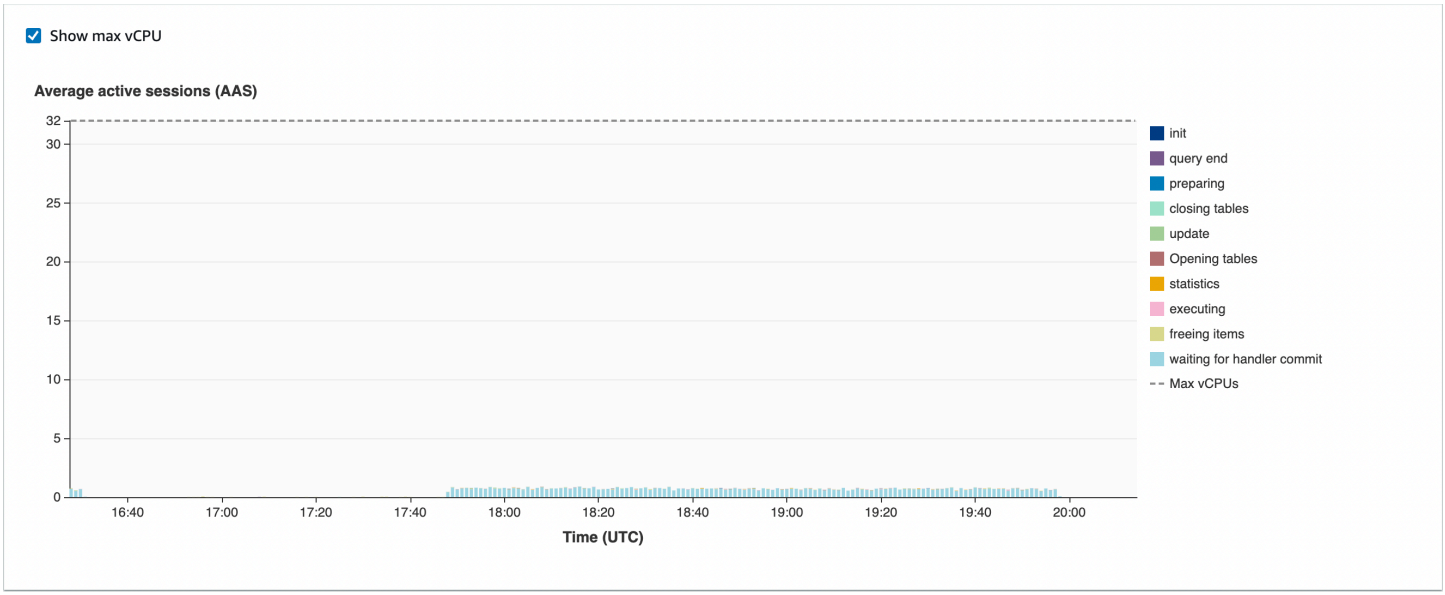
I seguenti comandi sysbench sono stati utilizzati per preparare ed eseguire il carico sul database.

```
sysbench oltp_read_write --db-driver=mysql --mysql-db=<DATABASE> --mysql-user=<USER> --
mysql-password=<PASSWORD> --mysql-host=<ENDPOINT> --tables=500 --table-size=2000000 --
threads=500 prepare
sysbench oltp_read_write --db-driver=mysql --mysql-db=employees --mysql-user=admin --
mysql-password=qwertyuiop --mysql-host=mysql8.cbbhujzeoxed.us-east-1.rds.amazonaws.com
 --tables=500 --rate=500 --time=7200 run
```

Nel grafico seguente, era in esecuzione un carico di lavoro OLTP e il processo pt-archiver è iniziato dove è contrassegnata la freccia.



Non vi è alcun cambiamento significativo nell'utilizzo della CPU con pt-archiver in esecuzione in parallel, il che deduce che pt-archiver non influisce sulle query OLTP durante l'esecuzione.


Show max vCPU
Average active sessions (AAS)
32
30
25
20
15
10
5
0
16:40
17:00
17:20
17:40
18:00
18:20
18:40
19:00
19:20
19:40
20:00
Time (UTC)
init
query end
preparing
closing tables
update
Opening tables
statistics
executing
freeing items
waiting for handler commit
Max vCPUs

Appendice II

Questa sezione fornisce i risultati di benchmarking degli strumenti pt-archiver in diversi scenari. Lo strumento sysbench viene utilizzato in questo test per caricare il database. Tutti i test vengono eseguiti sull'istanza Amazon RDS for MySQL in esecuzione sulla classe di istanza. `db.r6g.8xlarge`

I seguenti comandi sysbench sono stati utilizzati per preparare ed eseguire il carico sul database:

```
sysbench oltp_read_write --db-driver=mysql --mysql-db=<DATABASE> --mysql-user=<USER> --
mysql-password=<PASSWORD> --mysql-host=<ENDPOINT> --tables=1000 --table-size=2000000 --
threads=500 prepare
sysbench oltp_read_write --db-driver=mysql --mysql-db=<DATABASE> --mysql-user=<USER>
 --mysql-password=<PASSWORD> --mysql-host=<ENDPOINT> --tables=1000 --rate=500 --
threads=500 run
```

Archiviazione di una tabella che non ha una chiave primaria e un solo indice (nessun carico sul database)

```
Started at 2022-11-07T05:29:12, ended at 2022-11-07T06:03:31
Action     Count           Time        Pct
commit    600050           1715.3582   83.31
select    300025           166.5470    8.09
inserting   300024         165.4025    8.03
other       0               11.6644    0.57
```

Sono stati necessari circa 34 minuti per archiviare 300.024 righe. Questa tabella conteneva 2 milioni di righe, ma lo strumento archiviava solo le righe con dati univoci per la colonna indicizzata.

Archiviazione di una tabella con una chiave primaria (nessun carico sul database)

```
Started at 2022-11-16T08:53:49, ended at 2022-11-16T12:38:18
Action         Count        Time            Pct
commit       4000000        11065.9534      82.16
select       2000000        1278.1854        9.49
inserting    1999999        1050.4961        7.80
other              0        74.1519          0.55
```

Sono state necessarie circa 3 ore, 44 minuti e 29 secondi per archiviare 1.999.999 righe.

Il grafico seguente mostra che pt-archiver consuma pochissime CPU e risorse quando viene eseguito da solo senza alcun carico sul sistema.

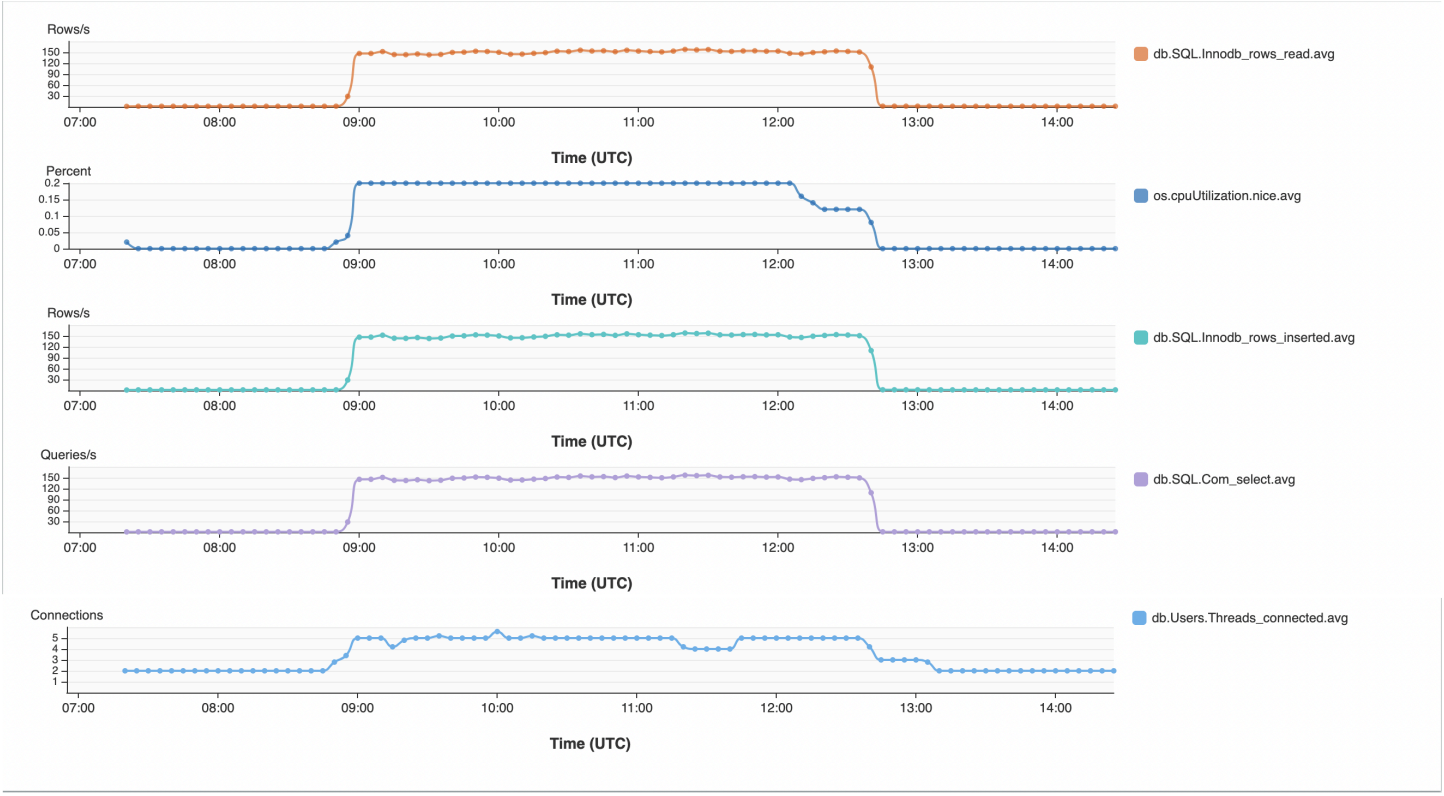
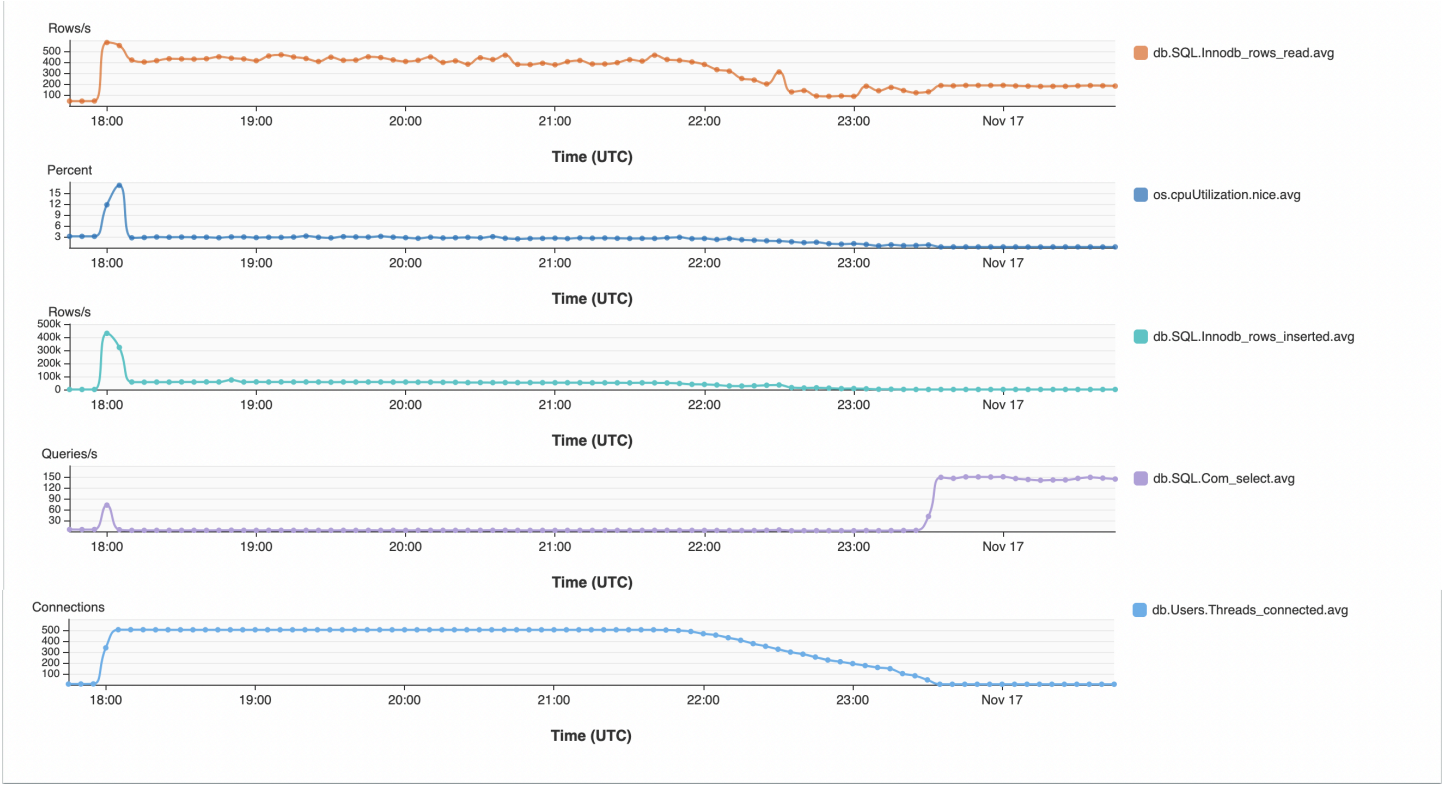


Tabella di archiviazione con una chiave primaria (con carico sul database)

```
Started at 2022-11-16T17:37:07, ended at 2022-11-17T03:20:43
Action          Count          Time           Pct
commit        4000000      19688.8362      56.23
inserting     1999999      13933.4418      39.79
select        2000000       1305.1770       3.73
other               0         89.1787       0.25
```

Sono state necessarie circa 9 ore, 43 minuti e 36 secondi per archiviare 1999999 righe.

Il grafico seguente mostra che durante il test, l'utilizzo della CPU è stato fino al 15% a causa del carico applicato da sysbench. Una volta completato il caricamento, pt-archiver ha continuato a funzionare consumando una quantità minima di CPU come previsto per completare l'archiviazione.



Come risulta evidente dai grafici, pt-archiver non archivia in modo aggressivo quando c'è un carico sul database.

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Aggiorna	Sono state aggiunte informazioni sull'esportazione di dati da un cluster Aurora DB live ad Amazon S3.	11 aprile 2025
Aggiorna	Informazioni rimosse su alcune opzioni di archiviazione.	21 ottobre 2024
Pubblicazione iniziale	—	8 giugno 2023

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.