aws

Guia do Desenvolvedor

AMIs de deep learning da AWS

AMIs de deep learning da AWS: Guia do Desenvolvedor

Table of Contents

O que é AMIs de deep learning da AWS?

AMIs de deep learning da AWS (DLAMI) fornece imagens de máquina personalizadas que você pode usar para aprendizado profundo na nuvem. Eles DLAMIs estão disponíveis na maioria Regiões da AWS para uma variedade de tipos de instância do Amazon Elastic Compute Cloud (AmazonEC2), desde uma instância CPU pequena até as mais recentes multiinstâncias de alta potência. GPU Eles DLAMIs vêm pré-configurados com NVIDIACUDAe NVIDIA com as versões mais recentes das estruturas de aprendizado profundo mais populares. DNN

Sobre este Guia

O conteúdo do pode ajudá-lo a lançar e usar DLAMIs o. O guia aborda vários casos de uso comuns de aprendizado profundo, tanto para treinamento quanto para inferência. Também aborda como escolher a opção certa AMI para sua finalidade e o tipo de instância que você pode preferir.

Além disso, DLAMIs incluem vários tutoriais fornecidos por suas estruturas suportadas. Este guia pode mostrar como ativar cada estrutura e encontrar os tutoriais apropriados para começar. Ele também tem tutoriais sobre treinamento distribuído, depuração, uso AWS Inferência e AWS Trainium e outros conceitos-chave. Para obter instruções sobre como configurar um servidor de notebook Jupyter para executar os tutoriais em seu navegador, consulte. Configurando um servidor Jupyter Notebook em uma instância DLAMI

Pré-requisitos

Para executar com sucesso oDLAMIs, recomendamos que você esteja familiarizado com as ferramentas de linha de comando e o Python básico.

Exemplo de casos de uso do DLAMI

A seguir estão exemplos de alguns casos de uso comuns para AMIs de deep learning da AWS (DLAMI).

Aprender sobre aprendizado profundo — DLAMI é uma ótima opção para aprender ou ensinar estruturas de aprendizado de máquina e aprendizado profundo. Eles DLAMIs eliminam a dor de cabeça de solucionar problemas nas instalações de cada estrutura e fazer com que elas funcionem no mesmo computador. Eles DLAMIs incluem um notebook Jupyter e facilitam a execução dos

tutoriais que as estruturas fornecem para pessoas novas em aprendizado de máquina e aprendizado profundo.

Desenvolvimento de aplicativos — Se você é um desenvolvedor de aplicativos interessado em usar o aprendizado profundo para fazer com que seus aplicativos utilizem os últimos avanços em IA, esse DLAMI é o banco de testes perfeito para você. Cada estrutura é fornecida com tutoriais sobre como começar a usar o deep learning, e muitas têm vários modelos que facilitam testar o aprendizado profundo sem precisar criar as redes neurais você mesmo ou fazer qualquer treinamento de modelo. Alguns exemplos mostram como criar um aplicativo de detecção de imagem em apenas alguns minutos, ou como criar um aplicativo de reconhecimento de voz para seu próprio chatbot.

Aprendizado de máquina e análise de dados — Se você é um cientista de dados ou está interessado em processar seus dados com aprendizado profundo, descobrirá que muitas das estruturas têm suporte para R e Spark. Você encontrará tutoriais sobre como fazer regressões simples, até a criação de sistemas de processamento de dados escaláveis para sistemas de personalização e previsões.

Pesquisa — Se você é um pesquisador que deseja experimentar uma nova estrutura, testar um novo modelo ou treinar novos modelos, então DLAMI e AWS os recursos de escala podem aliviar a dor de instalações tediosas e do gerenciamento de vários nós de treinamento.

> **Note**
>
> Embora sua escolha inicial seja atualizar seu tipo de instância para uma instância maior com mais GPUs (até 8), você também pode escalar horizontalmente criando um cluster de DLAMI instâncias. Confira Informações relacionadas sobre DLAMI para obter mais informações sobre criações de clusters.

Atributos do DLAMI

As características do AMIs de deep learning da AWS (DLAMI) incluem estruturas de aprendizado profundo pré-instaladas, GPU software, servidores de modelos e ferramentas de visualização de modelos.

Estruturas pré-instaladas

Atualmente, existem dois tipos principais de, DLAMI com outras variações relacionadas ao sistema operacional (SO) e às versões do software:

- Aprendizado profundo AMI com Conda— Frameworks instalados separadamente usando `conda` pacotes e ambientes Python separados.
- AMI de deep learning base— Nenhuma estrutura instalada; somente NVIDIACUDAe outras dependências.

O Deep Learning AMI with Conda usa `conda` ambientes para isolar cada estrutura, para que você possa alternar entre elas à vontade e não se preocupar com o conflito de dependências. O Deep Learning AMI with Conda oferece suporte às seguintes estruturas:

- PyTorch
- TensorFlow 2

> **Note**
>
> DLAMInão oferece mais suporte às seguintes estruturas de aprendizado profundo: Apache, MXNet Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Chainer e Keras.

Software pré-instalado GPU

Mesmo se você usar uma instância CPU -only, o DLAMIs will have NVIDIACUDAe o NVIDIADNNcu. O software instalado é o mesmo, independentemente do tipo de instância. Lembre-se de que ferramentas GPU específicas funcionam somente em uma instância que tenha pelo menos umaGPU. Para obter mais informações sobre os tipos de instância, consulteEscolha de um tipo de DLAMI instância.

Para obter mais informações sobreCUDA, consulteInstalações do CUDA e associações da estrutura.

Distribuição e visualização de modelos

O Deep Learning AMI with Conda vem pré-instalado com servidores de modelos para TensorFlow, bem como TensorBoard para visualizações de modelos. Para obter mais informações, consulte TensorFlow Servindo.

Começando com DLAMI

Este guia inclui dicas sobre como escolher DLAMI a instância certa para você, selecionar um tipo de instância adequado ao seu caso de uso e orçamento e Informações relacionadas sobre DLAMI que descreve configurações personalizadas que podem ser interessantes.

Se você está começando a usar AWS ou usar a AmazonEC2, comece com Aprendizado profundo AMI com Conda o. Se você está familiarizado com a Amazon EC2 e outros AWS serviços, como Amazon EMREFS, Amazon ou Amazon S3, e está interessado em integrar esses serviços para projetos que precisam de treinamento ou inferência distribuída, confira se um deles se Informações relacionadas sobre DLAMI adequa ao seu caso de uso.

Recomendamos que você confira Escolhendo um DLAMI para ter uma ideia do tipo de instância que pode ser melhor para seu aplicativo.

Próxima etapa

Escolhendo um DLAMI

Escolhendo um DLAMI

Oferecemos uma variedade de DLAMI opções. Para ajudar você a selecionar a correta DLAMI para seu caso de uso, agrupamos as imagens pelo tipo de hardware ou pela funcionalidade para a qual elas foram desenvolvidas. Nossos agrupamentos de alto nível são:

- DLAMITipo: CUDA versus base versus estrutura única versus estrutura múltipla (Conda) DLAMI
- Arquitetura de computação: Graviton baseado em x86 versus Graviton baseado em ARM64 AWS
- Tipo de processador: GPUhttps://docs.aws.amazon.com/dlami/latest/devguide/gpu versus Inferentia CPUversus Trainium
- SDK: CUDAversus AWS Neurônio
- Sistema operacional: Amazon Linux em comparação com Ubuntu

O restante dos tópicos deste guia ajudam você a obter mais informações e detalhes.

Tópicos

- Instalações do CUDA e associações da estrutura

- AMI de deep learning base
- Aprendizado profundo AMI com Conda
- Opções de arquitetura DLAMI
- Opções do sistema operacional da DLAMI

A seguir

Aprendizado profundo AMI com Conda

Instalações do CUDA e associações da estrutura

O aprendizado profundo é de última geração, e cada estrutura oferece versões "estáveis". Essas versões estáveis podem não funcionar com a implementação e os recursos mais recentes do CUDA ou do cuDNN. O caso de uso e os recursos necessários podem ajudar você a escolher uma estrutura. Se você não tiver certeza, use a AMI de aprendizado profundo mais recente com o Conda. Ele tem `pip` binários oficiais para todas as estruturas com CUDA, usando a versão mais recente suportada por cada estrutura. Se você quiser as versões mais recentes e personalizar seu ambiente de aprendizado profundo, use a AMI base de deep learning.

Consulte o nosso guia em Estabilidade e candidatos a lançamento para obter mais orientações.

Escolher uma DLAMI com o CUDA

AMI de deep learning baseTem todas as séries de versões CUDA disponíveis

Aprendizado profundo AMI com CondaTem todas as séries de versões CUDA disponíveis

> **Note**
>
> Não incluímos mais os ambientes MXNet, CNTK, Caffe, Caffe2, Theano, Chainer ou Keras Conda no. AMIs de deep learning da AWS

Para obter números de versão específicos da estrutura, consulte Notas da versão da DLAMIs

Escolha este tipo de DLAMI ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

Escolha uma das versões do CUDA e analise a lista completa das DLAMIs que têm essa versão no Apêndice, ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

A seguir

AMI de deep learning base

Related Topics

- Para obter instruções sobre como alternar entre as versões do CUDA, consulte o tutorial Usando a Base de Aprendizado Profundo AMI.

AMI de deep learning base

A AMI do deep learning base é como uma tela vazia para aprendizado profundo. Ela vem com tudo o que você precisa até o ponto da instalação de uma determinada estrutura e tem sua escolha de versões do CUDA.

Por que escolher a DLAMI base

Esse grupo de AMIs é útil para colaboradores de projeto que desejam usar um projeto de deep learning e criar o mais recente. É para alguém que deseja implementar seu próprio ambiente com a confiança de que o software NVIDIA mais recente está instalado e funcionando para que possa concentrar-se em escolher as estruturas e versões que deseja instalar.

Escolha este tipo de DLAMI ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

A seguir

DLAMI com o Conda

Tópicos relacionados

- Uso da AMI de deep learning base

Aprendizado profundo AMI com Conda

O Conda DLAMI usa ambientes `conda` virtuais, eles estão presentes em várias estruturas ou em uma estrutura única. DLAMIs Esses ambientes são configurados para manter as instalações de estruturas diferentes separadas e simplificar a alternância entre estruturas. Isso é ótimo para aprender e experimentar todas as estruturas que DLAMI ele tem a oferecer. A maioria dos usuários acha que o novo Deep Learning AMI with Conda é perfeito para eles.

Eles são atualizados frequentemente com as versões mais recentes das estruturas e têm os GPU drivers e o software mais recentes. Eles geralmente são chamados de “os” AMIs de deep learning da AWS na maioria dos documentos. Eles são DLAMIs compatíveis com os sistemas operacionais Ubuntu 20.04, Amazon Linux 2. O suporte a sistemas operacionais depende do suporte do sistema operacional upstream.

Estabilidade e candidatos a lançamento

O Conda AMIs usa binários otimizados das versões formais mais mais recentes de cada estrutura. Versões "release candidate" e recursos experimentais não devem ser esperados. As otimizações dependem do suporte da estrutura para tecnologias de aceleração como a da Intel MKLDNN, que acelera o treinamento e a inferência nos tipos de instância C5 e C4. CPU Os binários também são compilados para suportar conjuntos de instruções avançadas da Intel, incluindo, mas não se limitando aAVX, AVX -2, SSE4 .1 e SSE4 .2. Eles aceleram as operações vetoriais e de ponto flutuante nas CPU arquiteturas Intel. Além disso, por GPU exemplo, os tipos the CUDA e cu DNN são atualizados com qualquer versão suportada pela versão oficial mais recente.

O Deep Learning AMI with Conda instala automaticamente a versão mais otimizada da estrutura para sua EC2 instância Amazon após a primeira ativação da estrutura. Para mais informações, consulte Usando o aprendizado profundo AMI com o Conda.

Se você deseja realizar a instalação da origem usando opções de compilação personalizadas ou otimizadas, a de AMI de deep learning bases pode ser a melhor opção para você.

Defasagem do Python 2

A comunidade de código aberto Python encerrou oficialmente o suporte para Python 2 em 1 de janeiro de 2020. A PyTorch comunidade TensorFlow e anunciou que as versões TensorFlow 2.1 e PyTorch 1.4 são as últimas com suporte ao Python 2. As versões anteriores do DLAMI (v26, v25, etc.) que contêm ambientes Python 2 Conda continuam disponíveis. No entanto, fornecemos atualizações para os ambientes Python 2 Conda em DLAMI versões publicadas anteriormente somente se houver correções de segurança publicadas pela comunidade de código aberto para essas versões. DLAMIlançamentos com as versões mais recentes dos PyTorch frameworks TensorFlow e não contêm os ambientes Python 2 Conda.

CUDASupport

Números de CUDA versão específicos podem ser encontrados nas notas GPU DLAMI de lançamento.

A seguir

Opções de arquitetura DLAMI

Related Topics

- Para ver um tutorial sobre como usar o Deep Learning AMI com o Conda, consulte o Usando o aprendizado profundo AMI com o Conda tutorial.

Opções de arquitetura DLAMI

AMIs de deep learning da AWSOs s são oferecidos com arquiteturas Graviton2 baseadas em x86 ou AWS ARM64.

Para obter informações sobre como começar a usar o DLAMI da GPU ARM64, consulte. O ARM64 DLAMI Para obter mais detalhes sobre os tipos de instância disponíveis, consulte Escolha de um tipo de DLAMI instância.

A seguir

Opções do sistema operacional da DLAMI

Opções do sistema operacional da DLAMI

As DLAMIs são oferecidas nos seguintes sistemas operacionais.

- Amazon Linux 2
- Ubuntu 20.04
- Ubuntu 22.04

Há versões mais antigas dos sistemas operacionais disponíveis em DLAMIs descontinuadas. Para obter mais informações sobre a suspensão de uso da DLAMI, consulte Descontinuação da DLAMI

Antes de escolher uma DLAMI, avalie o tipo de instância que você precisa e identifique sua região da AWS .

A seguir

Escolha de um tipo de DLAMI instância

Escolha de um tipo de DLAMI instância

De forma mais geral, considere o seguinte ao escolher um tipo de instância para DLAMI a.

- Se você é iniciante no aprendizado profundo, uma instância com uma única GPU pode atender às suas necessidades.
- Se você se preocupa com o orçamento, pode usar instâncias CPU somente.
- Se você deseja otimizar o alto desempenho e a eficiência de custos para inferência de modelos de aprendizado profundo, pode usar instâncias com chips AWS Inferentia.
- Se você estiver procurando por uma GPU instância de alto desempenho com uma CPU arquitetura baseada em ARM64, poderá usar o tipo de instância G5g.
- Se você estiver interessado em executar um modelo pré-treinado para inferência e previsões, você pode anexar um Amazon Elastic Inference à sua instância da Amazon. EC2 O Amazon Elastic Inference oferece acesso a um acelerador com uma fração de a. GPU
- Para serviços de inferência de alto volume, uma única CPU instância com muita memória, ou um cluster dessas instâncias, pode ser uma solução melhor.
- Se estiver usando um modelo grande com muitos dados ou um tamanho de lote amplo, você precisará de uma instância maior com mais memória. Você também pode distribuir seu modelo para um cluster deGPUs. Usar uma instância com menos memória talvez seja a melhor solução se você diminuir o tamanho do lote. Isso pode afetar sua precisão e velocidade de treinamento.
- Se você estiver interessado em executar aplicativos de aprendizado de máquina usando a NVIDIA Collective Communications Library (NCCL) que exige altos níveis de comunicação entre nós em grande escala, talvez você queira usar o Elastic Fabric Adapter (EFA).

Para obter mais detalhes sobre instâncias, consulte de instância.

Os tópicos a seguir fornecem informações sobre as considerações relacionadas aos tipos de instância.

> **Important**
>
> O Deep Learning AMIs inclui drivers, software ou kits de ferramentas desenvolvidos, de propriedade ou fornecidos pela NVIDIA Corporation. Você concorda em usar esses NVIDIA drivers, software ou kits de ferramentas somente em EC2 instâncias da Amazon que incluam NVIDIA hardware.

Tópicos

Definição de preço para a DLAMI

As estruturas de aprendizado profundo incluídas no DLAMI são gratuitas e cada uma tem suas próprias licenças de código aberto. Embora o software incluído no DLAMI seja gratuito, você ainda precisa pagar pelo hardware subjacente da EC2 instância Amazon.

Alguns tipos de EC2 instância da Amazon são rotulados como gratuitos. É possível executar o DLAMI em uma dessas instâncias gratuitas. Isso significa que o uso do DLAMI é totalmente gratuito quando você usa apenas a capacidade dessa instância. Se você precisar de uma instância mais poderosa com mais CPU núcleos, mais espaço em discoRAM, mais ou uma ou maisGPUs, precisará de uma instância que não esteja na classe de instância de nível livre.

Para obter mais informações sobre opções e preços de instâncias, consulte EC2Preços da Amazon.

Disponibilidade de regiões do DLAMI

Cada região oferece suporte a uma diferente variedade de tipos de instância e, muitas vezes, um tipo de instância tem um custo um pouco diferente em regiões diferentes. DLAMIsnão estão disponíveis em todas as regiões, mas é possível copiar DLAMIs para a região de sua escolha. Consulte Copiar um AMI para obter mais informações. Observe a lista de seleções de regiões e escolha a que esteja próxima de você ou de seus clientes. Se você planeja usar mais de um DLAMI e potencialmente criar um cluster, certifique-se de usar a mesma região para todos os nós no cluster.

Para obter mais informações sobre regiões, acesse Amazon EC2 Service Endpoints.

A seguir

GPUInstâncias recomendadas

GPUInstâncias recomendadas

Recomendamos uma GPU instância para a maioria dos propósitos de aprendizado profundo. Treinar novos modelos é mais rápido em uma GPU instância do que em uma CPU instância. Você pode escalar sublinearmente quando tiver várias GPU instâncias ou se usar treinamento distribuído em várias instâncias com. GPUs

Os tipos de instância a seguir oferecem suporte aoDLAMI. Para obter informações sobre as opções de tipo de GPU instância e seus usos, consulte de EC2 instância e selecione Computação acelerada.

> **Note**
>
> O tamanho do seu modelo deve ser um fator na escolha de uma instância. Se seu modelo exceder a disponibilidade de uma instânciaRAM, escolha um tipo de instância diferente com memória suficiente para seu aplicativo.

- As instâncias Amazon EC2 P5e têm até 8 NVIDIA Tesla H200. GPUs
- As instâncias Amazon EC2 P5 têm até 8 NVIDIA Tesla H100. GPUs
- As instâncias EC2 P4 da Amazon têm até 8 NVIDIA Tesla A100. GPUs
- As instâncias EC2 P3 da Amazon têm até 8 NVIDIA Tesla V100. GPUs
- As instâncias Amazon EC2 G3 têm até 4 NVIDIA Tesla M60. GPUs
- As instâncias Amazon EC2 G4 têm até 4 NVIDIA GPUs T4.
- As instâncias Amazon EC2 G5 têm até 8 NVIDIA GPUs A10G.
- As instâncias Amazon EC2 G6 têm até 8 NVIDIA GPUs L4.
- As instâncias Amazon EC2 G6e têm até 8 NVIDIA L40S Tensor Core. GPUs
- As instâncias Amazon EC2 G5g têm processadores Graviton2 baseados em ARM64 AWS .

DLAMIas instâncias fornecem ferramentas para monitorar e otimizar seus GPU processos. Para obter mais informações sobre como monitorar seus GPU processos, consulteGPUMonitoramento e otimização.

Para ver tutoriais específicos sobre como trabalhar com instâncias G5g, consulte O ARM64 DLAMI.

A seguir

CPUInstâncias recomendadas

CPUInstâncias recomendadas

Se você está com um orçamento limitado, aprendendo sobre aprendizado profundo ou apenas deseja executar um serviço de previsão, você tem muitas opções acessíveis na CPU categoria. Algumas estruturas aproveitam as vantagens da Intel MKLDNN, que acelera o treinamento e a inferência nos tipos de instância C5 (não disponível em todas as regiões)CPU. Para obter informações sobre os tipos de CPU instância, consulte de instância e selecione Otimizado para computação.

> **Note**
>
> O tamanho do seu modelo deve ser um fator na escolha de uma instância. Se seu modelo exceder a disponibilidade de uma instânciaRAM, escolha um tipo de instância diferente com memória suficiente para seu aplicativo.

- As instâncias Amazon EC2 C5 têm até 72 IntelvCPUs. As instâncias C5 se destacam em modelagem científica, processamento em lote, análise distribuída, computação de alto desempenho (HPC) e inferência de aprendizado profundo e de máquina.

A seguir

Instâncias recomendadas do Inferentia

Instâncias recomendadas do Inferentia

AWS As instâncias de inferência são projetadas para fornecer alto desempenho e economia para cargas de trabalho de inferência de modelos de aprendizado profundo. Especificamente, os tipos de instância Inf2 usam chips AWS Inferentia e o AWS Neuron SDK, que é integrado a estruturas populares de aprendizado de máquina, como e. TensorFlow PyTorch

Os clientes podem usar instâncias Inf2 para executar aplicativos de inferência de machine learning em grande escala, como pesquisa, mecanismos de recomendação, visão computacional, reconhecimento de fala, processamento de linguagem natural, personalização e detecção de fraudes, com o menor custo na nuvem.

> **Note**
>
> O tamanho do seu modelo deve ser um fator na escolha de uma instância. Se seu modelo exceder a disponibilidade de uma instânciaRAM, escolha um tipo de instância diferente com memória suficiente para seu aplicativo.

- As instâncias Amazon EC2 Inf2 têm até 16 chips AWS Inferentia e 100 Gbps de taxa de transferência de rede.

Para obter mais informações sobre como começar a usar a AWS inferênciaDLAMIs, consulteO chip de AWS inferência com DLAMI.

A seguir

Instâncias recomendadas do Trainium

Instâncias recomendadas do Trainium

AWS As instâncias Trainium são projetadas para fornecer alto desempenho e economia para cargas de trabalho de inferência de modelos de aprendizado profundo. Especificamente, os tipos de instância Trn1 usam chips AWS Trainium e o AWS Neuron SDK, que é integrado a estruturas populares de aprendizado de máquina, como e. TensorFlow PyTorch

Os clientes podem usar instâncias Trn1 para executar aplicativos de inferência de machine learning em grande escala, como pesquisa, mecanismos de recomendação, visão computacional, reconhecimento de fala, processamento de linguagem natural, personalização e detecção de fraudes, com o menor custo na nuvem.

> **Note**
>
> O tamanho do seu modelo deve ser um fator na escolha de uma instância. Se seu modelo exceder a disponibilidade de uma instânciaRAM, escolha um tipo de instância diferente com memória suficiente para seu aplicativo.

- As instâncias Amazon EC2 Trn1 têm até 16 chips AWS Trainium e 100 Gbps de taxa de transferência de rede.

Configurando uma DLAMI instância

Depois de escolher DLAMI e escolher um tipo de instância do Amazon Elastic Compute Cloud (AmazonEC2) que você deseja usar, você estará pronto para configurar sua nova DLAMI instância.

Se você ainda não escolheu um tipo de EC2 instância DLAMI e, consulteComeçando com DLAMI.

Tópicos

Encontrando o ID de um DLAMI

Cada um DLAMI tem um identificador (ID) exclusivo. Ao iniciar uma DLAMI instância usando o EC2 console da Amazon, você pode, opcionalmente, usar o DLAMI ID para pesquisar o DLAMI que deseja usar. Quando você executa uma DLAMI instância usando o AWS Command Line Interface (AWS CLI), esse ID é obrigatório.

Você pode encontrar o ID DLAMI de sua escolha usando um AWS CLI comando para Amazon EC2 ou Parameter Store, um recurso de AWS Systems Manager. Para obter instruções sobre como instalar e configurar o AWS CLI, consulte Introdução ao AWS CLI no Guia do AWS Command Line Interface usuário.

Using Parameter Store

Para encontrar um DLAMI ID usando ssm get-parameter

No ssm get-parametercomando a seguir, para a `--name` opção, o formato do nome do parâmetro é */aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id*. Nesse formato de nome, *`architecture`* pode ser um **`x86_64`** ou outro**`arm64`**. Especifique o *`ami_type`* pegando o DLAMI nome e removendo as palavras-chave “deep”, “learning” e “ami”. AMIO nome pode ser encontrado emNotas da versão da DLAMIs.

> **Important**
>
> Para usar esse comando, o principal AWS Identity and Access Management (IAM) que você usa deve ter a `ssm:GetParameter` permissão. Para obter mais informações sobre IAM diretores, consulte a seção Recursos adicionais das IAMfunções no Guia do IAM usuário.

- ```
  aws ssm get-parameter --name /aws/service/deeplearning/ami/x86_64/base-oss-nvidia-driver-ubuntu-22.04/latest/ami-id  \
  --region us-east-1 --query "Parameter.Value" --output text
  ```

  A saída deve ser semelhante à seguinte:

  ```
  ami-09ee1a996ac214ce7
  ```

  > **Tip**
  >
  > Para algumas DLAMI estruturas atualmente suportadas, você pode encontrar exemplos de ssm get-parameter comandos mais específicos emNotas da versão da DLAMIs. Escolha o link para as notas de lançamento de sua escolha eDLAMI, em seguida, encontre sua consulta de ID nas notas de versão.

Using Amazon EC2 CLI

Para encontrar um DLAMI ID usando ec2 describe-images

No ec2 describe-imagescomando a seguir, para o valor do filtro`Name=name`, insira o DLAMI nome. Você pode especificar uma versão de lançamento para uma determinada estrutura ou obter a versão mais recente substituindo o número da versão por um ponto de interrogação (?).

- ```
  aws ec2 describe-images --region us-east-1 --owners amazon \
  --filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &amp;CreationDate))[:1].ImageId' --output text
  ```

  A saída deve ser semelhante à seguinte:

```
ami-09ee1a996ac214ce7
```

> **Tip**
>
> Para obter um exemplo de ec2 describe-images comando específico para o DLAMI de sua escolha, consulteNotas da versão da DLAMIs. Escolha o link para as notas de lançamento de sua escolha eDLAMI, em seguida, encontre sua consulta de ID nas notas de versão.

Próxima etapa

Lançamento de uma DLAMI instância

Lançamento de uma DLAMI instância

Depois de encontrar o ID do DLAMI que você quer usar para iniciar uma DLAMI instância, você estará pronto para executar a instância. Para iniciá-lo, você pode usar o EC2 console da Amazon ou o AWS Command Line Interface (AWS CLI).

> **Note**
>
> Para este passo a passo, podemos fazer referências específicas ao driver OSS Nvidia do Deep Learning Base GPU AMI (Ubuntu 22.04). Mesmo se você selecionar um diferenteDLAMI, poderá seguir este guia.

EC2 console

> **Note**
>
> Para acelerar aplicativos de computação de alto desempenho (HPC) e aprendizado de máquina, você pode iniciar sua DLAMI instância com um adaptador Elastic Fabric (EFA). Para obter instruções específicas, consulteExecutando uma AMIs de deep learning da AWS instância com EFA.

1. Abra o EC2console.
2. Anote sua atual Região da AWS na navegação superior. Se essa não for a região desejada, altere essa opção antes de continuar. Para obter mais informações, consulte dos endpoints EC2 de serviço da Amazon no Referência geral da Amazon Web Services.
3. Escolha Executar instância.
4. Insira um nome para sua instância e selecione o DLAMI que é certo para você.

   a. Encontre um existente DLAMI em Meu AMIs ou escolha Início rápido.

   b. Pesquise por DLAMI ID. Navegue pelas opções e selecione a escolhida.
5. Escolha um tipo de instância. Você pode encontrar as famílias de instâncias recomendadas para você DLAMI emNotas da versão da DLAMIs. Para obter recomendações gerais sobre tipos de DLAMI instância, consulteEscolha de um tipo de DLAMI instância.
6. Escolha Executar instância.

AWS CLI

- Para usar o AWS CLI, você deve ter o ID do DLAMI que deseja usar, o tipo de EC2 instância Região da AWS e as informações do seu token de segurança. Em seguida, você pode iniciar a instância usando o ec2 run-instances AWS CLI comando.

  Para obter instruções sobre como instalar e configurar o AWS CLI, consulte Introdução ao AWS CLI no Guia do AWS Command Line Interface usuário. Para obter mais informações, incluindo exemplos de comandos, consulte Iniciar, listar e fechar EC2 instâncias da Amazon para AWS CLI o.

Depois de iniciar sua instância usando o EC2 console da Amazon ou AWS CLI, aguarde até que a instância esteja pronta. Isso normalmente leva apenas alguns minutos. Você pode verificar o status da instância no EC2console da Amazon. Para obter mais informações, consulte Verificações de status para EC2 instâncias da Amazon no Guia EC2 do usuário da Amazon.

Próxima etapa

Conectando-se a uma DLAMI instância

Conectando-se a uma DLAMI instância

Depois de executar uma DLAMI instância e ela estar em execução, você pode se conectar a ela a partir de um cliente (Windows, macOS ou Linux) usando. SSH Para obter instruções, consulte Conecte-se à sua instância Linux usando SSH o Guia EC2 do usuário da Amazon.

Mantenha uma cópia do comando de SSH login à mão caso queira configurar um servidor Jupyter Notebook depois de fazer login. Para se conectar à página da web do Jupyter, você usa uma variação desse comando.

Próxima etapa

Configurando um servidor Jupyter Notebook em uma instância DLAMI

Configurando um servidor Jupyter Notebook em uma instância DLAMI

Com um servidor Jupyter Notebook, você pode criar e executar notebooks Jupyter a partir da sua instância. DLAMI Com os notebooks Jupyter, você pode realizar experimentos de aprendizado de máquina (ML) para treinamento e inferência enquanto usa a AWS infraestrutura e acessa pacotes incorporados no. DLAMI Para obter mais informações sobre os notebooks Jupyter, consulte The Jupyter Notebook no site de documentação do usuário do Jupyter.

Para configurar um servidor Jupyter Notebook, você deve:

- Configure o servidor Jupyter Notebook na sua DLAMI instância.
- Configure seu cliente para se conectar ao servidor Jupyter Notebook. Fornecemos instruções de configuração para Windows, macOS e clientes Linux.
- Teste a configuração fazendo login no servidor Jupyter Notebook.

Para concluir essas etapas, siga as instruções nos tópicos a seguir. Depois de configurar um servidor Jupyter Notebook, você pode executar os exemplos de tutoriais de notebook fornecidos no. DLAMIs Para obter mais informações, consulte Executar os tutoriais do notebook Jupyter.

Tópicos

- Protegendo o servidor Jupyter Notebook em uma instância DLAMI

- Iniciando o servidor Jupyter Notebook em uma instância DLAMI
- Conectando um cliente ao servidor Jupyter Notebook em uma instância DLAMI
- Fazendo login no servidor Jupyter Notebook em uma instância DLAMI

Protegendo o servidor Jupyter Notebook em uma instância DLAMI

Para manter seu servidor Jupyter Notebook seguro, recomendamos configurar uma senha e criar um SSL certificado para o servidor. Para configurar uma senhaSSL, primeiro conecte-se à sua DLAMI instância e siga estas instruções.

Para proteger o servidor Jupyter Notebook

1. O Jupyter fornece um utilitário de senha. Execute o comando a seguir e insira sua senha preferencial no prompt.

   ```
   $ jupyter notebook password
   ```

   O resultado será semelhante ao seguinte:

   ```
   Enter password:
        Verify password:
        [NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/
   jupyter_notebook_config.json
   ```

2. Crie um SSL certificado autoassinado. Siga os prompts para preencher sua localidade, conforme julgar necessário. É necessário inserir `.` se deseja deixar um prompt em branco. Suas respostas não afetarão a funcionalidade do certificado.

   ```
   $ cd ~
        $ mkdir ssl
        $ cd ssl
        $ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout mykey.key -out
    mycert.pem
   ```

> **Note**
>
> Talvez você esteja interessado em criar um SSL certificado comum assinado por terceiros e que não faça com que o navegador emita um aviso de segurança. Esse processo é muito

mais complexo. Para obter mais informações, consulte Protegendo um servidor de notebook na documentação do usuário do Jupyter Notebook.

Próxima etapa

Iniciando o servidor Jupyter Notebook em uma instância DLAMI

Iniciando o servidor Jupyter Notebook em uma instância DLAMI

Depois de proteger seu servidor Jupyter Notebook com uma senha SSL, você pode iniciar o servidor. Faça login na sua DLAMI instância e execute o comando a seguir que usa o SSL certificado que você criou anteriormente.

```
$ jupyter notebook --certfile=~/ssl/mycert.pem --keyfile ~/ssl/mykey.key
```

Com o servidor iniciado, agora você pode se conectar a ele por meio de um SSH túnel a partir do seu computador cliente. Quando o servidor for executado, você verá alguma saída do Jupyter confirmando que o servidor está em execução. Nesse ponto, ignore a chamada de que você pode acessar o servidor por meio de um host localURL, porque isso não funcionará até que você crie o túnel.

> **Note**
>
> O Jupyter resolverá a troca de ambientes para você ao alternar as estruturas usando a interface da web do Jupyter. Para obter mais informações, consulte Alternar ambientes com o Jupyter.

Próxima etapa

Conectando um cliente ao servidor Jupyter Notebook em uma instância DLAMI

Conectando um cliente ao servidor Jupyter Notebook em uma instância DLAMI

Depois de iniciar o servidor Jupyter Notebook na sua DLAMI instância, configure seu cliente Windows, macOS ou Linux para se conectar ao servidor. Ao se conectar, você pode criar e acessar

notebooks Jupyter no servidor em seu espaço de trabalho e executar seu código de aprendizado profundo no servidor.

Pré-requisitos

Certifique-se de ter o seguinte, necessário para configurar um SSH túnel:

- O DNS nome público da sua EC2 instância da Amazon. Para obter mais informações, consulte os tipos de nome de host de EC2 instância da Amazon no Guia do EC2 usuário da Amazon.
- O par de chaves do arquivo de chave privada. Para obter mais informações sobre como acessar seu par de chaves, consulte os pares de EC2 chaves da Amazon e as EC2 instâncias da Amazon no Guia EC2 do usuário da Amazon.

Conecte-se a partir de um cliente Windows, macOS ou Linux

Para se conectar à sua DLAMI instância a partir de um cliente Windows, macOS ou Linux, siga as instruções do sistema operacional do seu cliente.

Windows

Para se conectar à sua DLAMI instância a partir de um cliente Windows usando SSH

1. Use um SSH cliente para Windows, como PuTTY. Para obter instruções, consulte Conecte-se à sua instância Linux usando Pu TTY no Guia EC2 do usuário da Amazon. Para outras opções de SSH conexão, consulte Conecte-se à sua instância Linux usando SSH.
2. (Opcional) Crie um SSH túnel para um servidor Jupyter em execução. Instale o Git Bash no seu cliente Windows e siga as instruções de conexão para clientes macOS e Linux.

macOS or Linux

Para se conectar à sua DLAMI instância a partir de um cliente macOS ou Linux usando SSH

1. Abra um terminal do .
2. Execute o comando a seguir para encaminhar todas as solicitações na porta local 8888 para a porta 8888 em sua instância remota da AmazonEC2. Atualize o comando substituindo a localização da sua chave para acessar a EC2 instância da Amazon e o DNS nome público da sua EC2 instância da Amazon. Observe que, para um Amazon LinuxAMI, o nome de usuário é `ec2-user` em vez de `ubuntu`.

```
$ ssh -i ~/mykeypair.pem -N -f -L 8888:localhost:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

Esse comando abre um túnel entre seu cliente e a EC2 instância remota da Amazon que está executando o servidor Jupyter Notebook.

Próxima etapa

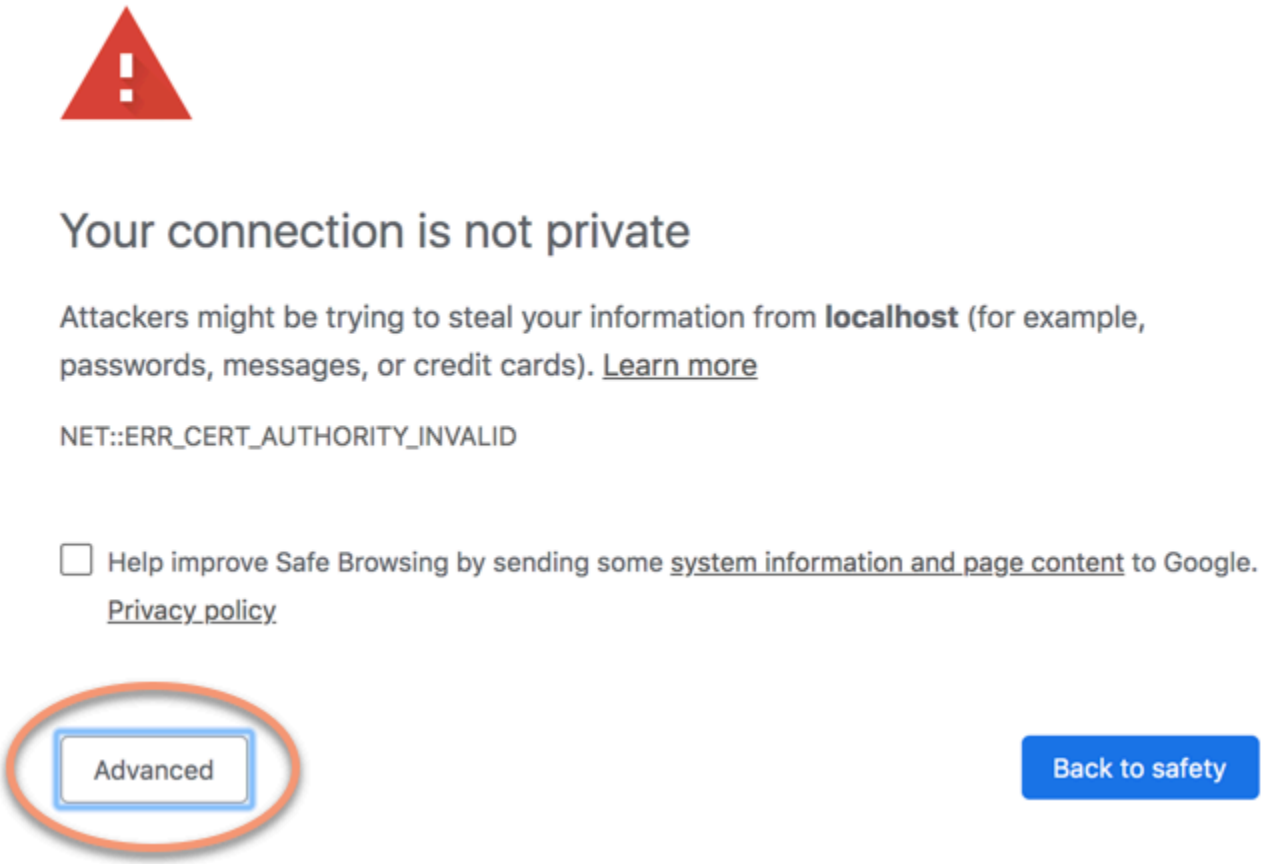
Fazendo login no servidor Jupyter Notebook em uma instância DLAMI

Fazendo login no servidor Jupyter Notebook em uma instância DLAMI

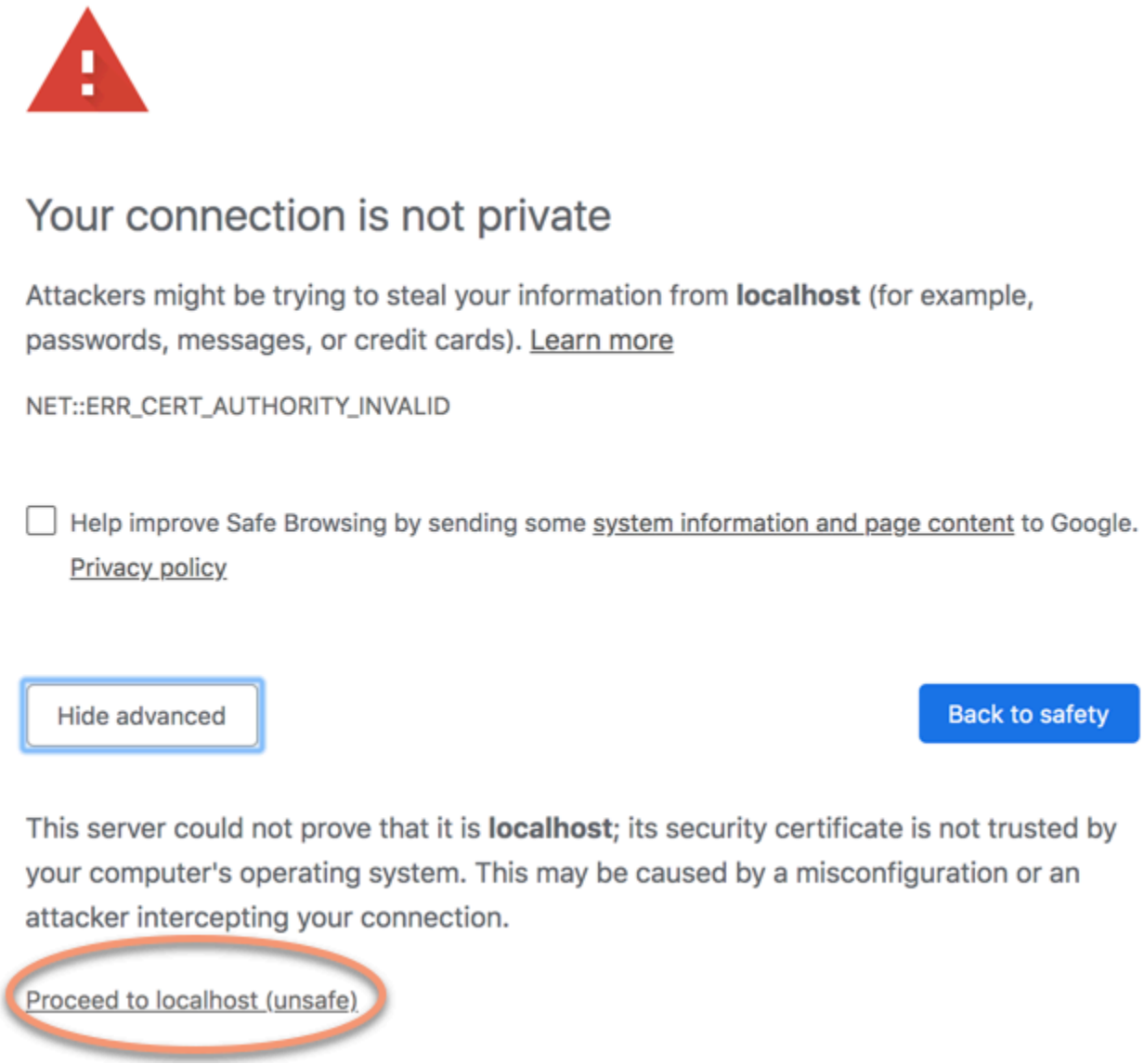
Depois de conectar seu cliente ao servidor Jupyter Notebook na sua DLAMI instância, você pode fazer login no servidor.

Para fazer login no servidor em seu navegador

1. Na barra de endereço do seu navegador, digite o seguinte URL ou clique neste link: https://localhost:8888
2. Com um SSL certificado autoassinado, seu navegador o avisará e solicitará que você evite continuar visitando o site.



Como você mesmo configurou isso, é seguro continuar. O navegador receberá um botão "avançado", "mostrar detalhes" ou algo semelhante.



Clique nele e depois clique no link "prosseguir para localhost". Se a conexão for bem-sucedida, você verá a página do servidor Jupyter Notebook. Nesse ponto, será solicitada a senha que você configurou anteriormente.

Agora você tem acesso ao servidor Jupyter Notebook que está sendo executado na DLAMI instância. Você pode criar novos cadernos ou executar os Tutoriais fornecidos.

Limpando uma DLAMI instância

Quando você não precisar mais da sua DLAMI instância, poderá interrompê-la ou encerrá-la na Amazon EC2 para evitar cobranças inesperadas.

Se você interromper uma instância, poderá mantê-la ativa e iniciá-la mais tarde, quando quiser usá-la novamente. Suas configurações, arquivos e outras informações não voláteis são armazenadas em um volume no Amazon Simple Storage Service (Amazon S3). Enquanto sua instância está parada, você incorre em cobranças do S3 pela retenção do volume, mas não incorre em cobranças por recursos computacionais. Quando você iniciar a instância novamente, ela montará esse volume de armazenamento com seus dados.

Se você encerrar uma instância, ela desaparecerá e você não poderá iniciá-la novamente. Obviamente, você não incorrerá em mais cobranças pelos recursos computacionais com uma instância encerrada. No entanto, seus dados ainda residem no Amazon S3 e você pode continuar incorrendo em cobranças do S3. Para evitar todas as cobranças adicionais relacionadas à sua instância encerrada, você também deve excluir o volume de armazenamento no Amazon S3. Para obter instruções, consulte Encerrar EC2 instâncias da Amazon no Guia do EC2 usuário da Amazon.

Para obter mais informações sobre os estados das EC2 instâncias da Amazon, como `stopped` `eterminated`, consulte as mudanças no estado da EC2 instância da Amazon no Guia EC2 do usuário da Amazon.

Usando um DLAMI

Tópicos

As seções a seguir descrevem como o Deep Learning AMI with Conda pode ser usado para alternar ambientes, executar códigos de amostra de cada uma das estruturas e executar o Jupyter para que você possa experimentar diferentes tutoriais de notebooks.

Usando o aprendizado profundo AMI com o Conda

Tópicos

Introdução ao aprendizado profundo AMI com Conda

O Conda é um sistema de gerenciamento de pacotes e de ambientes de código aberto que é executado no Windows, macOS e Linux. O Conda instala, executa e atualiza pacotes e suas dependências rapidamente. O Conda cria, salva, carrega e alterna facilmente entre ambientes em seu computador local.

O Deep Learning AMI with Conda foi configurado para que você alterne facilmente entre ambientes de aprendizado profundo. As instruções a seguir orientam você na execução de alguns comandos básicos com `conda`. Elas também ajudam você a verificar se a importação básica da estrutura está funcionando e se você pode executar algumas operações simples com a estrutura. Em seguida, você pode passar para tutoriais mais completos fornecidos com os exemplos DLAMI ou exemplos de estruturas encontrados no site do projeto de cada estrutura.

Faça login no seu DLAMI

Depois de fazer login no seu servidor, você verá uma “mensagem do dia” do servidor (MOTD) descrevendo vários comandos do Conda que você pode usar para alternar entre as diferentes estruturas de aprendizado profundo. Abaixo está um exemploMOTD. Suas especificações MOTD podem variar à medida que novas versões do DLAMI são lançadas.

```
=============================================================================
        AMI Name: Deep Learning OSS Nvidia Driver AMI (Amazon Linux 2) Version 77
        Supported EC2 instances: G4dn, G5, G6, Gr6, P4d, P4de, P5
            * To activate pre-built tensorflow environment, run: 'source activate
 tensorflow2_p310'
            * To activate pre-built pytorch environment, run: 'source activate
 pytorch_p310'
            * To activate pre-built python3 environment, run: 'source activate python3'

        NVIDIA driver version: 535.161.08

    CUDA versions available: cuda-11.7 cuda-11.8 cuda-12.0 cuda-12.1 cuda-12.2

    Default CUDA version is 12.1

    Release notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-
release-notes.html
    AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
    Developer Guide and Release Notes: https://docs.aws.amazon.com/dlami/latest/
devguide/what-is-dlami.html
    Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
    For a fully managed experience, check out Amazon SageMaker at https://
aws.amazon.com/sagemaker
    =============================================================================
```

Inicie o TensorFlow ambiente

> **Note**
>
> Quando iniciar seu primeiro ambiente Conda, aguarde enquanto ele carrega. O Deep Learning AMI with Conda instala automaticamente a versão mais otimizada da estrutura para sua EC2 instância na primeira ativação da estrutura. Não deve haver atrasos subsequentes.

1. Ative o ambiente TensorFlow virtual para Python 3.

```
$ source activate tensorflow2_p310
```

2. Inicie o iPython terminal.

```
(tensorflow2_p310)$ ipython
```

3. Execute um TensorFlow programa rápido.

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Você deve ver "Hello, Tensorflow!"

A seguir

Executar os tutoriais do notebook Jupyter

Mude para o PyTorch ambiente Python 3

Se você ainda estiver no iPython console`quit()`, use e prepare-se para trocar de ambiente.

- Ative o ambiente PyTorch virtual para Python 3.

```
$ source activate pytorch_p310
```

Teste alguns PyTorch códigos

Para testar sua instalação, use o Python para escrever um PyTorch código que cria e imprime uma matriz.

1. Inicie o iPython terminal.

```
(pytorch_p310)$ ipython
```

2. Importar PyTorch.

```
import torch
```

Você pode ver uma mensagem de aviso sobre um pacote de terceiros. Você pode ignorá-lo.

3. Crie uma matriz 5x3 com elementos inicializados de modo aleatório. Imprima a matriz.

```
x = torch.rand(5, 3)
print(x)
```

Verifique o resultado.

```
tensor([[0.3105, 0.5983, 0.5410],
        [0.0234, 0.0934, 0.0371],
        [0.9740, 0.1439, 0.3107],
        [0.6461, 0.9035, 0.5715],
        [0.4401, 0.7990, 0.8913]])
```

Remoção de ambientes

Se você ficar sem espaço noDLAMI, você pode optar por desinstalar os pacotes Conda que você não está usando:

```
conda env list
conda env remove --name <env_name>
```

Usando a Base de Aprendizado Profundo AMI

Usando a Base de Aprendizado Profundo AMI

O Base AMI vem com uma plataforma fundamental de GPU drivers e bibliotecas de aceleração para implantar seu próprio ambiente personalizado de aprendizado profundo. Por padrão, o AMI é configurado com qualquer ambiente de NVIDIA CUDA versão. Você também pode alternar entre diferentes versões doCUDA. Consulte as seguintes instruções para saber como fazer isso.

Configurando versões CUDA

Você pode verificar a CUDA versão executando NVIDIA o `nvcc` programa.

```
nvcc --version
```

Você pode selecionar e verificar uma CUDA versão específica com o seguinte comando bash:

```
sudo rm /usr/local/cuda
sudo ln -s /usr/local/cuda-12.0 /usr/local/cuda
```

Para obter mais informações, consulte as notas de DLAMI versão do Base.

Executar os tutoriais do notebook Jupyter

Tutoriais e exemplos são fornecidos com a fonte de cada projeto de aprendizado profundo e, na maioria dos casos, eles serão executados em qualquer um. DLAMI Se você escolher a Aprendizado profundo AMI com Conda, obterá o benefício adicional de alguns tutoriais escolhidos a dedo já configurados e prontos para serem testados.

> **Important**
>
> Para executar os tutoriais do notebook Jupyter instalados noDLAMI, você precisará. Configurando um servidor Jupyter Notebook em uma instância DLAMI

Assim que o servidor do Jupyter estiver em execução, você poderá executar os tutoriais por meio de seu navegador da web. Se você estiver executando o Deep Learning AMI com o Conda ou se tiver configurado ambientes Python, poderá alternar os kernels do Python a partir da interface do notebook Jupyter. Selecione o kernel apropriado antes de tentar executar um tutorial específico à estrutura. Outros exemplos disso são fornecidos para usuários do Deep Learning AMI with Conda.

> **Note**
>
> Muitos tutoriais exigem módulos Python adicionais que podem não estar configurados no seu. DLAMI Se você receber um erro como`"xyz module not found"`, faça login noDLAMI, ative o ambiente conforme descrito acima e instale os módulos necessários.

> **Tip**
>
> Os tutoriais e exemplos de aprendizado profundo geralmente dependem de um ou mais. GPUs Se o tipo de instância não tiver umGPU, talvez seja necessário alterar parte do código do exemplo para executá-lo.

Navegar pelos tutoriais instalados

Depois de fazer login no servidor Jupyter e ver o diretório de tutoriais (somente no Deep Learning AMI com Conda), você verá pastas de tutoriais com o nome de cada estrutura. Se você não vê uma estrutura listada, então os tutoriais não estão disponíveis para essa estrutura em sua estrutura atual. DLAMI Clique no nome da estrutura para ver os tutoriais listados e, em seguida, clique em um tutorial para iniciá-lo.

Na primeira vez que você executar um notebook sobre o Deep Learning AMI com o Conda, ele desejará saber qual ambiente você gostaria de usar. Você poderá selecionar de uma lista. Cada ambiente é denominado de acordo com este padrão:

`Environment (conda_framework_python-version)`

Por exemplo, você pode ver`Environment (conda_mxnet_p36)`, o que significa que o ambiente tem um MXNet Python 3. A outra variação disso seria`Environment (conda_mxnet_p27)`, o que significa que o ambiente tem um MXNet Python 2.

> **Tip**
>
> Se você está preocupado com a versão do que CUDA está ativa, uma forma de ver isso é MOTD quando você faz login pela primeira vez noDLAMI.

Alternar ambientes com o Jupyter

Se você decidir experimentar um tutorial para uma estrutura diferente, certifique-se de verificar o kernel em execução. Essas informações podem ser vistas no canto superior direito da interface do Jupyter, logo abaixo do botão de logout. Você pode alterar o kernel em qualquer notebook clicando no item de menu Kernel do Jupyter, em seguida, em Change Kernel e, em seguida, clicando no ambiente que atende ao notebook que você está executando.

Neste ponto, você precisará executar novamente todas as células porque uma alteração no kernel apagará o estado de qualquer coisa que tenha executado anteriormente.

> **Tip**
>
> Alternar entre as estruturas pode ser divertido e instrutivo, no entanto, você pode esgotar a memória. Se você começar a receber erros, examine a janela de terminal que tem o servidor do Jupyter em execução. Há mensagens úteis e registros de erros aqui, e você pode ver um out-of-memory erro. Para corrigir isso, você pode ir para a página inicial do servidor do Jupyter, clicar na guia Running e, em seguida, em Shutdown para cada um dos tutoriais que provavelmente ainda estão em execução em segundo plano e consumindo toda a sua memória.

Tutoriais

A seguir estão os tutoriais sobre como usar o Deep Learning AMI com o software da Conda.

Tópicos

Ativar estruturas

A seguir estão as estruturas de aprendizado profundo instaladas no Deep Learning AMI with Conda. Clique em uma estrutura para saber como ativá-la.

Tópicos

PyTorch

Ativando PyTorch

Quando um pacote Conda estável de uma estrutura é lançado, ele é testado e pré-instalado no. DLAMI Se desejar executar as mais recentes compilações noturnas não testadas, você pode PyTorchInstall's Nightly Build (experimental) manualmente.

Para ativar a estrutura atualmente instalada, siga estas instruções em seu Deep Learning AMI com o Conda.

Para PyTorch no Python 3 com CUDA e MKL -DNN, execute este comando:

```
$ source activate pytorch_p310
```

Inicie o iPython terminal.

```
(pytorch_p310)$ ipython
```

Execute um PyTorch programa rápido.

```
import torch
x = torch.rand(5, 3)
print(x)
print(x.size())
y = torch.rand(5, 3)
print(torch.add(x, y))
```

Você deve ver a matriz aleatória inicial impressa, seu tamanho e a adição de outra matriz aleatória.

PyTorchInstall's Nightly Build (experimental)

Como instalar a PyTorch partir de uma compilação noturna

Você pode instalar a PyTorch versão mais recente em um ou em ambos os ambientes PyTorch Conda em seu Deep Learning AMI com o Conda.

1. • (Opção para Python 3) - Ative o ambiente Python 3: PyTorch

   ```
   $ source activate pytorch_p310
   ```

2. As etapas restantes assumem que você está usando o ambiente do `pytorch_p310`. Remova o atualmente instalado PyTorch:

```
(pytorch_p310)$ pip uninstall torch
```

3.
   - (Opção para GPU instâncias) - Instale a versão noturna mais recente do PyTorch with CUDA .0:

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/
nightly/cu100/torch_nightly.html
```

   - (Opção para CPU instâncias) - Instale a versão noturna mais recente de PyTorch for instâncias sem: GPUs

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/
nightly/cpu/torch_nightly.html
```

4. Para verificar se você instalou com sucesso a última compilação noturna, inicie o IPython terminal e verifique a versão do. PyTorch

```
(pytorch_p310)$ ipython
```

```
import torch
print (torch.__version__)
```

   A saída deve imprimir algo semelhante a `1.0.0.dev20180922`

5. Para verificar se a compilação PyTorch noturna funciona bem com o MNIST exemplo, você pode executar um script de teste no repositório de exemplos PyTorch da:

```
(pytorch_p310)$ cd ~
(pytorch_p310)$ git clone https://github.com/pytorch/examples.git pytorch_examples
(pytorch_p310)$ cd pytorch_examples/mnist
(pytorch_p310)$ python main.py || exit 1
```

Mais tutoriais

Para mais tutoriais e exemplos, consulte os documentos oficiais, a PyTorch documentação e o site da estrutura. PyTorch

TensorFlow 2

Este tutorial mostra como ativar TensorFlow 2 em uma instância executando o Deep Learning AMI com o Conda (DLAMIno Conda) e executar um programa TensorFlow 2.

Quando um pacote Conda estável de uma estrutura é lançado, ele é testado e pré-instalado no. DLAMI

Ativando 2 TensorFlow

Para rodar TensorFlow DLAMI com Conda

1. Para ativar TensorFlow 2, abra uma instância do Amazon Elastic Compute Cloud (AmazonEC2) do DLAMI com Conda.
2. Para TensorFlow 2 e Keras 2 no Python 3 CUDA com 10.1 MKL e DNN -, execute este comando:

   ```
   $ source activate tensorflow2_p310
   ```

3. Inicie o iPython terminal:

   ```
   (tensorflow2_p310)$ ipython
   ```

4. Execute um programa TensorFlow 2 para verificar se ele está funcionando corretamente:

   ```
   import tensorflow as tf
   hello = tf.constant('Hello, TensorFlow!')
   tf.print(hello)
   ```

   `Hello, TensorFlow!` deve aparecer na tela.

Mais tutoriais

Para ver mais tutoriais e exemplos, consulte a TensorFlow documentação do TensorFlow Python API ou acesse o site. TensorFlow

Treinamento distribuído usando Elastic Fabric Adapter

Um adaptador Elastic Fabric (EFA) é um dispositivo de rede que você pode conectar à sua DLAMI instância para acelerar aplicativos de computação de alto desempenho (HPC). EFApermite que você alcance o desempenho do aplicativo de um HPC cluster local, com a escalabilidade, a flexibilidade e a elasticidade fornecidas pela nuvem. AWS

Os tópicos a seguir mostram como começar a usar EFA com DLAMI o.

> **Note**
>
> Escolha o DLAMI seu nesta GPUDLAMIlista base

Tópicos

- Executando uma AMIs de deep learning da AWS instância com EFA
- Usando EFA no DLAMI

Executando uma AMIs de deep learning da AWS instância com EFA

O Base mais recente DLAMI está pronto para uso EFA e vem com os drivers necessários, módulos de kernel, libfabric, openmpi e o NCCL OFI plug-in para instâncias. GPU

Você pode encontrar as CUDA versões suportadas de um Base DLAMI nas notas de lançamento.

Nota:

- Ao executar um NCCL aplicativo usando `mpirun` onEFA, você precisará especificar o caminho completo para a instalação EFA suportada como:

  ```
  /opt/amazon/openmpi/bin/mpirun <command>
  ```

- Para permitir que seu aplicativo seja usadoEFA, adicione `FI_PROVIDER="efa"` ao `mpirun` comando conforme mostrado emUsando EFA no DLAMI.

Tópicos

- Preparar um grupo de segurança habilitado para o EFA
- Executar sua instância

- Verificar anexo do EFA

Preparar um grupo de segurança habilitado para o EFA

EFArequer um grupo de segurança que permita todo o tráfego de entrada e saída de e para o próprio grupo de segurança. Para obter mais informações, consulte a EFAdocumentação.

1. Abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. No painel de navegação, escolha Security Groups (Grupos de segurança) e, em seguida, Create Security Group (Criar grupo de segurança).
3. Na janela Security group, faça o seguinte:
   - Em Security group name (Nome do grupo de segurança), insira um nome descritivo para o grupo de segurança, como `EFA-enabled security group`.
   - (Opcional) Em Description (Descrição), insira uma breve descrição do grupo de segurança.
   - Para VPC, selecione aquela VPC na qual você pretende executar suas instâncias EFA habilitadas.
   - Escolha Criar.
4. Selecione o grupo de segurança que você criou e, na guia Description (Descrição), copie o Group ID (ID do grupo).
5. Nas guias Entrada e Saída, faça o seguinte:
   - Selecione Edit.
   - Para Type (Tipo), escolha All traffic (Todo o tráfego).
   - Em Source, escolha Custom.
   - Cole o ID do grupo de segurança que você copiou no campo.
   - Escolha Salvar.
6. Habilite o tráfego de entrada fazendo referência a Autorizar tráfego de entrada para as instâncias do Linux. Se você pular essa etapa, não conseguirá se comunicar com sua DLAMI instância.

Executar sua instância

EFAno momento, o on the AMIs de deep learning da AWS é compatível com os seguintes tipos de instância e sistemas operacionais:

- P3DN.24xlarge: Amazon Linux 2, Ubuntu 20.04
- P4D.24xlarge: Amazon Linux 2, Ubuntu 20.04
- P5.48xlarge: Amazon Linux 2, Ubuntu 20.04

A seção a seguir mostra como iniciar uma DLAMI instância EFA habilitada. Para obter mais informações sobre como iniciar uma instância EFA habilitada, consulte Launch EFA -Enabled Instances into a Cluster Placement Group.

1. Abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. Escolha Executar instância.
3. Na AMI página Escolha um, selecione um suporte DLAMI encontrado na página de notas de DLAMI versão
4. Na página Escolher um tipo de instância, selecione um dos seguintes tipos de instância com suporte e escolha Próximo: Configurar os detalhes da instância. Consulte este link para ver a lista de instâncias compatíveis: Comece com EFA e MPI
5. Na página Configure Instance Details (Configurar detalhes da instância), faça o seguinte:
   - Em Número de instâncias, insira o número de instâncias EFA habilitadas que você deseja iniciar.
   - Em Rede e sub-rede, selecione a sub-rede VPC e na qual executar as instâncias.
   - [Opcional] Em Grupo de posicionamento, selecione Adicionar instância ao grupo de posicionamento. Para obter o melhor desempenho, execute as instâncias dentro de um placement group.
   - [Opcional] Em Nome do grupo de posicionamento, selecione Adicionar a um novo grupo de posicionamento, insira um nome descritivo para o grupo de posicionamento e, em seguida, em Estratégia do grupo de posicionamento, selecione cluster.
   - Habilite o “Elastic Fabric Adapter” nesta página. Se esta opção estiver desabilitada, altere a sub-rede para uma que ofereça suporte ao tipo de instância selecionado.
   - Na seção Interfaces de rede, para dispositivo eth0, escolha Nova interface de rede. Opcionalmente, você pode especificar um IPv4 endereço principal e um ou mais IPv4 endereços secundários. Se você estiver iniciando a instância em uma sub-rede que tenha um IPv6 CIDR bloco associado, você pode especificar opcionalmente um IPv6 endereço primário e um ou mais endereços secundáriosIPv6.
   - Escolha Next: Add Storage (Próximo: adicionar armazenamento).

6. Na página Adicionar armazenamento, especifique os volumes a serem anexados às instâncias, além dos volumes especificados pelo AMI (como o volume do dispositivo raiz) e escolha Avançar: Adicionar tags.
7. Na página Adicionar tags, especifique tags para as instâncias, como nome amigável, e selecione Próximo: Configurar grupo de segurança.
8. Na página Configurar grupo de segurança, em Atribuir um grupo de segurança, selecione Selecionar um grupo de segurança existente e, em seguida, selecione o grupo de segurança que você criou anteriormente.
9. Escolha revisar e iniciar.
10. Na página Revisar execução da instância, reveja as configurações, e escolha Executar para escolher um par de chaves e executar a instâncias.

Verificar anexo do EFA

No console

Depois de iniciar a instância, verifique os detalhes da instância no AWS console. Para fazer isso, selecione a instância no EC2 console e veja a guia Descrição no painel inferior da página. Encontre o parâmetro "Network Interfaces: eth0" e clique em eth0 e um pop-up será aberto. O “Elastic Fabric Adapter” deve estar habilitado.

Se não EFA estiver ativado, você pode corrigir isso da seguinte maneira:

- Encerrar a EC2 instância e iniciar uma nova com as mesmas etapas. Certifique-se de que o EFA esteja anexado.
- Anexe o EFA a uma instância existente.

  1. No EC2 console, vá para Interfaces de rede.
  2. Clique em "Criar uma interface de rede".
  3. Selecione a mesma sub-rede na qual está sua instância.
  4. Habilite o "Elastic Fabric Adapter" e clique em Criar.
  5. Volte para a guia EC2 Instâncias e selecione sua instância.
  6. Vá para Ações: Estado da instância e interrompa a instância antes de anexarEFA.
  7. Em "Ações", selecione "Rede: Anexar Interface de Rede".
  8. Selecione a interface que você acabou de criar e clique em associar.

9. Reinicie sua instância.

Na instância

O script de teste a seguir já está presente noDLAMI. Execute-o para garantir que os módulos do kernel sejam carregados corretamente.

```
$ fi_info -p efa
```

O resultado deve ser semelhante ao seguinte:

```
provider: efa
    fabric: EFA-fe80::e5:56ff:fe34:56a8
    domain: efa_0-rdm
    version: 2.0
    type: FI_EP_RDM
    protocol: FI_PROTO_EFA
provider: efa
    fabric: EFA-fe80::e5:56ff:fe34:56a8
    domain: efa_0-dgrm
    version: 2.0
    type: FI_EP_DGRAM
    protocol: FI_PROTO_EFA
provider: efa;ofi_rxd
    fabric: EFA-fe80::e5:56ff:fe34:56a8
    domain: efa_0-dgrm
    version: 1.0
    type: FI_EP_RDM
    protocol: FI_PROTO_RXD
```

Verificar a configuração do grupo de segurança

O script de teste a seguir já está presente noDLAMI. Execute-o para garantir que o grupo de segurança criado esteja configurado corretamente.

```
$ cd /opt/amazon/efa/test/
$ ./efa_test.sh
```

O resultado deve ser semelhante ao seguinte:

```
Starting server...
```

```
Starting client...
bytes   #sent   #ack     total       time     MB/sec    usec/xfer   Mxfers/sec
64      10      =10      1.2k        0.02s      0.06    1123.55       0.00
256     10      =10      5k          0.00s     17.66      14.50       0.07
1k      10      =10      20k         0.00s     67.81      15.10       0.07
4k      10      =10      80k         0.00s    237.45      17.25       0.06
64k     10      =10      1.2m        0.00s    921.10      71.15       0.01
1m      10      =10      20m         0.01s   2122.41     494.05       0.00
```

Se ele parar de responder ou não for concluído, verifique se o grupo de segurança tem as regras corretas de entrada/saída.

Usando EFA no DLAMI

A seção a seguir descreve como usar EFA para executar aplicativos de vários nós no AMIs de deep learning da AWS.

Executar aplicativos de vários nós com o EFA

Para executar um aplicativo em um cluster de nós, é necessária a seguinte configuração:

Tópicos

Ativar sem senha SSH

Selecione um nó no cluster como o nó líder. Os nós restantes são referidos como nós membros.

1. No nó líder, gere o RSA par de chaves.

   ```
   ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
   ```

2. Altere as permissões da chave privada no nó líder.

   ```
   chmod 600 ~/.ssh/id_rsa
   ```

3. Copie `~/.ssh/id_rsa.pub` a chave pública e anexe-a aos nós membros `~/.ssh/authorized_keys` do cluster.

4. Agora, você poderá fazer login diretamente nos nós membros a partir do nó líder usando o ip privado.

```
ssh <member private ip>
```

5. Desative a strictHostKey verificação e habilite o encaminhamento de agentes no nó principal adicionando o seguinte ao arquivo ~/.ssh/config no nó líder:

```
Host *
    ForwardAgent yes
Host *
    StrictHostKeyChecking no
```

6. Nas instâncias do Amazon Linux 2, execute o seguinte comando no nó líder para fornecer as permissões corretas ao arquivo de configuração:

```
chmod 600 ~/.ssh/config
```

Criar arquivo de hosts

No nó líder, crie um arquivo de hosts para identificar os nós no cluster. O arquivo de hosts deve ter uma entrada para cada nó no cluster. Crie um arquivo ~/hosts e adicione cada nó usando o ip privado da seguinte forma:

```
localhost slots=8
<private ip of node 1> slots=8
<private ip of node 2> slots=8
```

NCCLTestes

> **Note**
>
> Esses testes foram executados usando a EFA versão 1.30.0 e o OFI NCCL Plugin 1.7.4.

Abaixo está listado um subconjunto de NCCL testes fornecidos pela Nvidia para testar a funcionalidade e o desempenho em vários nós de computação

Instâncias suportadas: P3dn, P4, P5

Testes de funcionalidade

NCCLTeste de transferência de mensagens em vários nós

O nccl_message_transfer é um teste simples para garantir que o plug-in esteja funcionando conforme o esperado. NCCL OFI O teste valida a funcionalidade do estabelecimento NCCL da conexão e da transferência APIs de dados. Certifique-se de usar o caminho completo para mpirun, conforme mostrado no exemplo, ao executar NCCL aplicativos com. EFA npAltere os parâmetros `N` com base no número de instâncias e GPUs no seu cluster. Para obter mais informações, consulte a AWS OFINCCLdocumentação.

O teste nccl_message_transfer a seguir é para uma versão xx.x genérica. CUDA Você pode executar os comandos para qualquer CUDA versão disponível na sua EC2 instância da Amazon substituindo a CUDA versão no script.

```
$/opt/amazon/openmpi/bin/mpirun -n 2 -N 1 --hostfile hosts \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:$LD_LIBRARY_PATH \
--mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to none \
opt/aws-ofi-nccl/tests/nccl_message_transfer
```

A saída será semelhante a: Você pode verificar a saída para ver se EFA está sendo usada como OFI provedor.

```
INFO: Function: nccl_net_ofi_init Line: 1069: NET/OFI Selected Provider is efa (found 4
 nics)
INFO: Function: nccl_net_ofi_init Line: 1160: NET/OFI Using transport protocol SENDRECV
INFO: Function: configure_ep_inorder Line: 261: NET/OFI Setting
 FI_OPT_EFA_SENDRECV_IN_ORDER_ALIGNED_128_BYTES not supported.
INFO: Function: configure_nccl_proto Line: 227: NET/OFI Setting NCCL_PROTO to "simple"
INFO: Function: main Line: 86: NET/OFI Process rank 1 started. NCCLNet device used on
 ip-172-31-13-179 is AWS Libfabric.
INFO: Function: main Line: 91: NET/OFI Received 4 network devices
INFO: Function: main Line: 111: NET/OFI Network supports communication using CUDA
 buffers. Dev: 3
INFO: Function: main Line: 118: NET/OFI Server: Listening on dev 3
INFO: Function: main Line: 131: NET/OFI Send connection request to rank 1
INFO: Function: main Line: 173: NET/OFI Send connection request to rank 0
INFO: Function: main Line: 137: NET/OFI Server: Start accepting requests
INFO: Function: main Line: 141: NET/OFI Successfully accepted connection from rank 1
INFO: Function: main Line: 145: NET/OFI Send 8 requests to rank 1
INFO: Function: main Line: 179: NET/OFI Server: Start accepting requests
```

```
INFO: Function: main Line: 183: NET/OFI Successfully accepted connection from rank 0
INFO: Function: main Line: 187: NET/OFI Rank 1 posting 8 receive buffers
INFO: Function: main Line: 161: NET/OFI Successfully sent 8 requests to rank 1
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 0
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 1
```

Testes de desempenho

Teste de NCCL desempenho de vários nós no P4D.24xlarge

Para verificar o NCCL desempenho comEFA, execute o teste de NCCL desempenho padrão que está disponível no repositório oficial NCCL-Tests. DLAMIEle vem com este teste já construído para CUDA XX.X. Você também pode executar seu próprio script com. EFA

Ao criar seu próprio script, consulte as seguintes orientações:

- Use o caminho completo para mpirun, conforme mostrado no exemplo, ao executar NCCL aplicativos com. EFA
- Altere os parâmetros np e N com base no número de instâncias e GPUs no seu cluster.
- Adicione o INFO sinalizador NCCL _ DEBUG = e certifique-se de que os registros indiquem o EFA uso como “O provedor selecionado éEFA”.
- Defina o local do registro de treinamento a ser analisado para validação

```
TRAINING_LOG="testEFA_$(date +"%N").log"
```

Use o comando `watch nvidia-smi` em qualquer um dos nós membros para monitorar o GPU uso. Os `watch nvidia-smi` comandos a seguir são para uma versão CUDA xx.x genérica e dependem do sistema operacional da sua instância. Você pode executar os comandos para qualquer CUDA versão disponível na sua EC2 instância da Amazon substituindo a CUDA versão no script.

- Amazon Linux 2:

```
 $ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib64:/opt/amazon/openmpi/
lib64:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
 none \
```

```
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
 100 | tee ${TRAINING_LOG}
```

- Ubuntu 20.04:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib:/opt/amazon/openmpi/
lib:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
 none \
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
 100 | tee ${TRAINING_LOG}
```

A saída será semelhante a:

```
# nThread 1 nGpus 1 minBytes 8 maxBytes 1073741824 step: 2(factor) warmup iters: 5
 iters: 100 agg iters: 1 validation: 1 graph: 0
#
# Using devices
#  Rank  0 Group  0 Pid   9591 on ip-172-31-4-37 device  0 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  1 Group  0 Pid   9592 on ip-172-31-4-37 device  1 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  2 Group  0 Pid   9593 on ip-172-31-4-37 device  2 [0x20] NVIDIA A100-SXM4-40GB
#  Rank  3 Group  0 Pid   9594 on ip-172-31-4-37 device  3 [0x20] NVIDIA A100-SXM4-40GB
#  Rank  4 Group  0 Pid   9595 on ip-172-31-4-37 device  4 [0x90] NVIDIA A100-SXM4-40GB
#  Rank  5 Group  0 Pid   9596 on ip-172-31-4-37 device  5 [0x90] NVIDIA A100-SXM4-40GB
#  Rank  6 Group  0 Pid   9597 on ip-172-31-4-37 device  6 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank  7 Group  0 Pid   9598 on ip-172-31-4-37 device  7 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank  8 Group  0 Pid  10216 on ip-172-31-13-179 device  0 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank  9 Group  0 Pid  10217 on ip-172-31-13-179 device  1 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank 10 Group  0 Pid  10218 on ip-172-31-13-179 device  2 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank 11 Group  0 Pid  10219 on ip-172-31-13-179 device  3 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank 12 Group  0 Pid  10220 on ip-172-31-13-179 device  4 [0x90] NVIDIA A100-
SXM4-40GB
#  Rank 13 Group  0 Pid  10221 on ip-172-31-13-179 device  5 [0x90] NVIDIA A100-
SXM4-40GB
```

```
#  Rank 14 Group  0 Pid  10222 on ip-172-31-13-179 device  6 [0xa0] NVIDIA A100-
SXM4-40GB
#  Rank 15 Group  0 Pid  10223 on ip-172-31-13-179 device  7 [0xa0] NVIDIA A100-
SXM4-40GB
ip-172-31-4-37:9591:9591 [0] NCCL INFO Bootstrap : Using ens32:172.31.4.37
ip-172-31-4-37:9591:9591 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin_v6
 symbol.
ip-172-31-4-37:9591:9591 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin
 symbol (v4 or v5).
ip-172-31-4-37:9591:9591 [0] NCCL INFO cudaDriverVersion 12020
NCCL version 2.18.5+cuda12.2
...
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.7.4-aws
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Using CUDA runtime version 11070
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Using CUDA runtime version 11070
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
 variable to 1
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Disabling NVLS support due to NCCL
 version 21602
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
 variable to 1
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Disabling NVLS support due to NCCL
 version 21602
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Running on p4d.24xlarge platform,
 Setting NCCL_TOPO_FILE environment variable to /opt/aws-ofi-nccl/share/aws-ofi-nccl/
xml/p4d-24xl-topo.xml
...
-----------------------------some output truncated-----------------------------------
#                                                              out-of-place
          in-place
#       size         count      type   redop    root     time   algbw   busbw #wrong
  time   algbw   busbw #wrong
#        (B)    (elements)                               (us)  (GB/s)  (GB/s)
  (us)  (GB/s)  (GB/s)
           0             0     float     sum      -1    11.02    0.00    0.00      0
 11.04    0.00    0.00      0
           0             0     float     sum      -1    11.01    0.00    0.00      0
 11.00    0.00    0.00      0
           0             0     float     sum      -1    11.02    0.00    0.00      0
 11.02    0.00    0.00      0
```

```
           0             0     float     sum      -1    11.01    0.00    0.00       0
11.00    0.00    0.00       0
           0             0     float     sum      -1    11.02    0.00    0.00       0
11.02    0.00    0.00       0
         256             4     float     sum      -1    632.7    0.00    0.00       0
628.2    0.00    0.00       0
         512             8     float     sum      -1    627.4    0.00    0.00       0
629.6    0.00    0.00       0
        1024            16     float     sum      -1    632.2    0.00    0.00       0
631.7    0.00    0.00       0
        2048            32     float     sum      -1    631.0    0.00    0.00       0
634.2    0.00    0.00       0
        4096            64     float     sum      -1    623.3    0.01    0.01       0
633.6    0.01    0.01       0
        8192           128     float     sum      -1    635.1    0.01    0.01       0
633.5    0.01    0.01       0
       16384           256     float     sum      -1    634.8    0.03    0.02       0
637.0    0.03    0.02       0
       32768           512     float     sum      -1    647.9    0.05    0.05       0
636.8    0.05    0.05       0
       65536          1024     float     sum      -1    658.9    0.10    0.09       0
667.0    0.10    0.09       0
      131072          2048     float     sum      -1    671.9    0.20    0.18       0
662.9    0.20    0.19       0
      262144          4096     float     sum      -1    692.1    0.38    0.36       0
685.1    0.38    0.36       0
      524288          8192     float     sum      -1    715.3    0.73    0.69       0
696.6    0.75    0.71       0
     1048576         16384     float     sum      -1    734.6    1.43    1.34       0
729.2    1.44    1.35       0
     2097152         32768     float     sum      -1    785.9    2.67    2.50       0
794.5    2.64    2.47       0
     4194304         65536     float     sum      -1    837.2    5.01    4.70       0
837.6    5.01    4.69       0
     8388608        131072     float     sum      -1    929.2    9.03    8.46       0
931.4    9.01    8.44       0
    16777216        262144     float     sum      -1   1773.6    9.46    8.87       0
1772.8    9.46    8.87       0
    33554432        524288     float     sum      -1   2110.2   15.90   14.91       0
2116.1   15.86   14.87       0
    67108864       1048576     float     sum      -1   2650.9   25.32   23.73       0
2658.1   25.25   23.67       0
   134217728       2097152     float     sum      -1   3943.1   34.04   31.91       0
3945.9   34.01   31.89       0
```

```
   268435456       4194304     float     sum      -1   7216.5   37.20   34.87      0
 7178.6   37.39   35.06      0
   536870912       8388608     float     sum      -1    13680   39.24   36.79      0
 13676   39.26   36.80      0
[  1073741824      16777216     float     sum      -1    25645   41.87   39.25      0
 25497   42.11   39.48      0 ] <- Used For Benchmark
...
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 7.46044
```

Testes de validação

Para validar se os EFA testes retornaram um resultado válido, use os seguintes testes para confirmar:

- Obtenha o tipo de instância usando os metadados da EC2 instância:

```
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-
token-ttl-seconds: 21600")
INSTANCE_TYPE=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/
latest/meta-data/instance-type)
```

- Execute a Testes de desempenho
- Defina os seguintes parâmetros

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Valide os resultados conforme mostrado:

```
RETURN_VAL=`echo $?`
if [ ${RETURN_VAL} -eq 0 ]; then

    # Information on how the version come from logs
    #
    # ip-172-31-27-205:6427:6427 [0] NCCL INFO cudaDriverVersion 12020
    # NCCL version 2.16.2+cuda11.8
    # ip-172-31-27-205:6427:6820 [0] NCCL INFO NET/OFI Initializing aws-ofi-nccl
 1.7.1-aws
    # ip-172-31-27-205:6427:6820 [0] NCCL INFO NET/OFI Using CUDA runtime version
 11060
```

```
    # cudaDriverVersion 12020  --> This is max supported cuda version by nvidia
 driver
    # NCCL version 2.16.2+cuda11.8 --> This is NCCL version compiled with cuda
 version
    # Using CUDA runtime version 11060 --> This is selected cuda version

    # Validation of logs
    grep "NET/OFI Using CUDA runtime version ${CUDA_RUNTIME_VERSION}" ${TRAINING_LOG}
 || { echo "Runtime cuda text not found"; exit 1; }
    grep "NET/OFI Initializing aws-ofi-nccl" ${TRAINING_LOG} || { echo "aws-ofi-nccl
 is not working, please check if it is installed correctly"; exit 1; }
    grep "NET/OFI Configuring AWS-specific options" ${TRAINING_LOG} || { echo "AWS-
specific options text not found"; exit 1; }
    grep "Using network AWS Libfabric" ${TRAINING_LOG} || { echo "AWS Libfabric text
 not found"; exit 1; }
    grep "busbw" ${TRAINING_LOG} || { echo "busbw text not found"; exit 1; }
    grep "Avg bus bandwidth " ${TRAINING_LOG} || { echo "Avg bus bandwidth text not
 found"; exit 1; }
    grep "NCCL version $NCCL_VERSION" ${TRAINING_LOG} || { echo "Text not found: NCCL
 version $NCCL_VERSION"; exit 1; }

    if [[ ${INSTANCE_TYPE} == "p4d.24xlarge" ]]; then
        grep "NET/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Text not found:
 NET/AWS Libfabric/0/GDRDMA"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
 { echo "Selected Provider is efa text not found"; exit 1; }
        grep "aws-ofi-nccl/xml/p4d-24xl-topo.xml" ${TRAINING_LOG} || { echo "Topology
 file not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p4de.24xlarge" ]]; then
        grep "NET/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus
 bandwidth text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
 { echo "Avg bus bandwidth text not found"; exit 1; }
        grep "aws-ofi-nccl/xml/p4de-24xl-topo.xml" ${TRAINING_LOG} || { echo
 "Topology file not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p5.48xlarge" ]]; then
        grep "NET/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus
 bandwidth text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
 { echo "Avg bus bandwidth text not found"; exit 1; }
        grep "aws-ofi-nccl/xml/p5.48xl-topo.xml" ${TRAINING_LOG} || { echo "Topology
 file not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p3dn.24xlarge" ]]; then
```

```
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
 { echo "Selected Provider is efa text not found"; exit 1; }
    fi
    echo "*************************** check_efa_nccl_all_reduce passed for cuda
 version ${CUDA_VERSION} ***************************"
else
    echo "*************************** check_efa_nccl_all_reduce failed for cuda
 version ${CUDA_VERSION} ***************************"
fi
```

- Para acessar os dados do benchmark, podemos analisar a linha final da saída da tabela do teste Multi Node all_reduce:

```
benchmark=$(sudo cat ${TRAINING_LOG} | grep '1073741824' | tail -n1 | awk -F " "
 '{{print $12}}' | sed 's/ //' | sed  's/  5e-07//')
if [[ -z "${benchmark}" ]]; then
  echo "benchmark variable is empty"
  exit 1
fi

echo "Benchmark throughput: ${benchmark}"
```

GPUMonitoramento e otimização

A seção a seguir o guiará pelas opções GPU de otimização e monitoramento. Esta seção é organizada como um fluxo de trabalho típico com monitoramento, supervisão, pré-processamento e treinamento.

- Monitoramento
  - Monitor GPUs com CloudWatch
- Otimização
  - Pré-processamento
  - Treinamento

Monitoramento

O seu DLAMI vem pré-instalado com várias ferramentas GPU de monitoramento. Este guia menciona também ferramentas que estão disponíveis para download e instalação.

- Monitor GPUs com CloudWatch- um utilitário pré-instalado que reporta estatísticas GPU de uso para a Amazon CloudWatch.
- nvidia-smi CLI - um utilitário para monitorar a utilização geral GPU da computação e da memória. Isso está pré-instalado no seu AMIs de deep learning da AWS (DLAMI).
- NVMLBiblioteca C - baseada em C API para acessar diretamente as funções de GPU monitoramento e gerenciamento. É usado pelo nvidia-smi CLI sob o capô e está pré-instalado no seu. DLAMI Também tem associações Python e Perl para facilitar o desenvolvimento nessas linguagens. O utilitário gpumon.py pré-instalado no seu DLAMI usa o pacote pynvml do. nvidia-ml-py
- NVIDIADCGM- Uma ferramenta de gerenciamento de clusters. Visite a página do desenvolvedor para saber como instalar e configurar essa ferramenta.

> **Tip**
>
> Confira o blog NVIDIA do desenvolvedor para obter as informações mais recentes sobre como usar as CUDA ferramentas instaladas emDLAMI:
>
> - Monitorando TensorCore a utilização usando Nsight IDE e nvprof.

Monitor GPUs com CloudWatch

Ao usar o seu DLAMI com um, GPU você pode descobrir que está procurando maneiras de monitorar seu uso durante o treinamento ou a inferência. Isso pode ser útil para otimizar o pipeline de dados e ajustar sua rede de aprendizado profundo.

Há duas maneiras de configurar GPU métricas com CloudWatch:

- Configurar métricas com o AWS CloudWatch agente (recomendado)
- Configurar métricas com o script `gpumon.py` pré-instalado

Configurar métricas com o AWS CloudWatch agente (recomendado)

Integre você DLAMI com o CloudWatch agente unificado para configurar GPU métricas e monitorar a utilização de GPU coprocessos nas instâncias EC2 aceleradas da Amazon.

Há quatro maneiras de configurar GPUmétricas com o seuDLAMI:

- Configurar GPU métricas mínimas
- Configurar GPU métricas parciais
- Configure todas as GPU métricas disponíveis
- Configurar GPU métricas personalizadas

Para obter mais informações sobre atualizações e patches de segurança, consulte Patches de segurança para o agente AWS CloudWatch

Pré-requisitos

Para começar, você deve configurar IAM as permissões de EC2 instância da Amazon que permitam que sua instância envie métricas para CloudWatch. Para obter etapas detalhadas, consulte Criar IAM funções e usuários para uso com o CloudWatch agente.

Configurar GPU métricas mínimas

Configure GPU métricas mínimas usando o `dlami-cloudwatch-agent@minimal` `systemd` serviço. Esse serviço configura as seguintes métricas:

- `utilization_gpu`
- `utilization_memory`

Você pode encontrar o `systemd` serviço para GPU métricas mínimas pré-configuradas no seguinte local:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-minimal.json
```

Habilite e inicie o serviço `systemd` com os seguintes comandos:

```
sudo systemctl enable dlami-cloudwatch-agent@minimal
sudo systemctl start dlami-cloudwatch-agent@minimal
```

Configurar GPU métricas parciais

Configure GPU métricas parciais usando o `dlami-cloudwatch-agent@partial` `systemd` serviço. Esse serviço configura as seguintes métricas:

- utilization_gpu
- utilization_memory
- memory_total
- memory_used
- memory_free

Você pode encontrar o systemd serviço para GPU métricas parciais pré-configuradas no seguinte local:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-partial.json
```

Habilite e inicie o serviço systemd com os seguintes comandos:

```
sudo systemctl enable dlami-cloudwatch-agent@partial
sudo systemctl start dlami-cloudwatch-agent@partial
```

Configure todas as GPU métricas disponíveis

Configure todas as GPU métricas disponíveis usando o dlami-cloudwatch-agent@all systemd serviço. Esse serviço configura as seguintes métricas:

- utilization_gpu
- utilization_memory
- memory_total
- memory_used
- memory_free
- temperature_gpu
- power_draw
- fan_speed
- pcie_link_gen_current
- pcie_link_width_current
- encoder_stats_session_count
- encoder_stats_average_fps

- encoder_stats_average_latency
- clocks_current_graphics
- clocks_current_sm
- clocks_current_memory
- clocks_current_video

Você pode encontrar o systemd serviço para todas as GPU métricas pré-configuradas disponíveis no seguinte local:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-all.json
```

Habilite e inicie o serviço systemd com os seguintes comandos:

```
sudo systemctl enable dlami-cloudwatch-agent@all
sudo systemctl start dlami-cloudwatch-agent@all
```

Configurar GPU métricas personalizadas

Se as métricas pré-configuradas não atenderem aos seus requisitos, você poderá criar um arquivo personalizado de configuração do CloudWatch agente.

Criar um arquivo de configuração personalizada

Para criar um arquivo de configuração personalizado, consulte as etapas detalhadas em Criar ou editar manualmente o arquivo de configuração do CloudWatch agente.

Neste exemplo, suponha que a definição do esquema esteja localizada em /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json.

Configurar métricas com seu arquivo personalizado

Execute o comando a seguir para configurar o CloudWatch agente de acordo com seu arquivo personalizado:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config -m ec2 -s -c \
file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
```

Patches de segurança para o agente AWS CloudWatch

Os recém-lançados DLAMIs são configurados com os patches de segurança de AWS CloudWatch agentes mais recentes disponíveis. Consulte as seções a seguir para atualizar seu atual DLAMI com os patches de segurança mais recentes, dependendo do sistema operacional de sua escolha.

Amazon Linux 2

Use `yum` para obter os patches de segurança de AWS CloudWatch agentes mais recentes para um Amazon Linux 2DLAMI.

```
sudo yum update
```

Ubuntu

Para obter os patches AWS CloudWatch de segurança mais recentes para um DLAMI com o Ubuntu, é necessário reinstalar o AWS CloudWatch agente usando um link de download do Amazon S3.

```
wget https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/
amazon-cloudwatch-agent.deb
```

Para obter mais informações sobre como instalar o AWS CloudWatch agente usando os links de download do Amazon S3, consulte Instalando e executando o CloudWatch agente em seus servidores.

Configurar métricas com o script **`gpumon.py`** pré-instalado

Um utilitário chamado gpumon.py está pré-instalado no seuDLAMI. Ele se integra CloudWatch e suporta o monitoramento de GPU pré-uso: GPU memória, GPU temperatura e GPU energia. O script envia periodicamente os dados monitorados para CloudWatch o. Você pode configurar o nível de granularidade dos dados enviados CloudWatch alterando algumas configurações no script. Antes de iniciar o script, no entanto, você precisará configurar CloudWatch para receber as métricas.

Como configurar e executar o GPU monitoramento com CloudWatch

1. Crie um IAM usuário ou modifique um existente para ter uma política para publicar a métrica CloudWatch. Se você criar um novo usuário, anote as credenciais, pois elas serão necessárias na próxima etapa.

   A IAM política a ser pesquisada é “cloudwatch:PutMetricData”. A política que é adicionada é a seguinte:

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData"
             ],
             "Effect": "Allow",
             "Resource": "*"
        }
   ]
}
```

> **Tip**
>
> Para obter mais informações sobre como criar um IAM usuário e adicionar políticas para CloudWatch, consulte a CloudWatch documentação.

2. No seuDLAMI, execute AWS configure e especifique as credenciais IAM do usuário.

```
$ aws configure
```

3. Talvez você precise fazer algumas modificações no utilitário gpumon antes de executá-lo. Você pode encontrar o utilitário gpumon e README no local definido no bloco de código a seguir. Para obter mais informações sobre o script `gpumon.py`, consulte a localização do script no Amazon S3.

```
Folder: ~/tools/GPUCloudWatchMonitor
Files:  ~/tools/GPUCloudWatchMonitor/gpumon.py
       ~/tools/GPUCloudWatchMonitor/README
```

   Opções:

   - Altere a região em gpumon.py se sua instância estiver NOT em us-east-1.
   - Altere outros parâmetros, como o CloudWatch `namespace` ou o período do relatório, com`store_reso`.

4. No momento, o script oferece suporte apenas ao Python 3. Ative o ambiente Python 3 da sua estrutura preferida ou ative o ambiente geral do DLAMI Python 3.

```
$ source activate python3
```

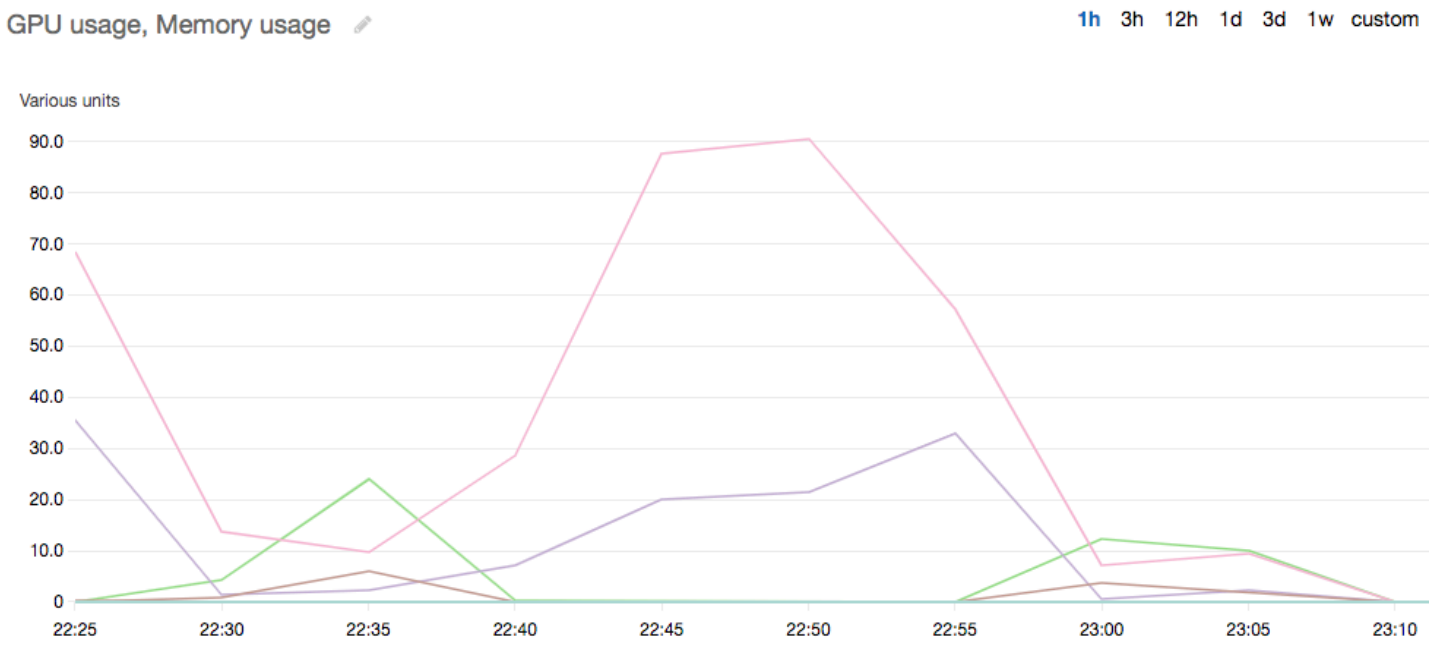
5. Execute o utilitário gpumon em segundo plano.

```
(python3)$ python gpumon.py &
```

6. Abra seu navegador na métrica https://console.aws.amazon.com/cloudwatch/e selecione. Ele terá um namespace ". DeepLearningTrain

> **Tip**
>
> Você pode alterar o namespace modificando o gpumon.py. Você também pode modificar o intervalo de relatório ajustando `store_reso`.

Veja a seguir um exemplo de CloudWatch gráfico relatando uma execução do gpumon.py monitorando um trabalho de treinamento na instância p2.8xlarge.



Talvez você se interesse por esses outros tópicos sobre GPU monitoramento e otimização:

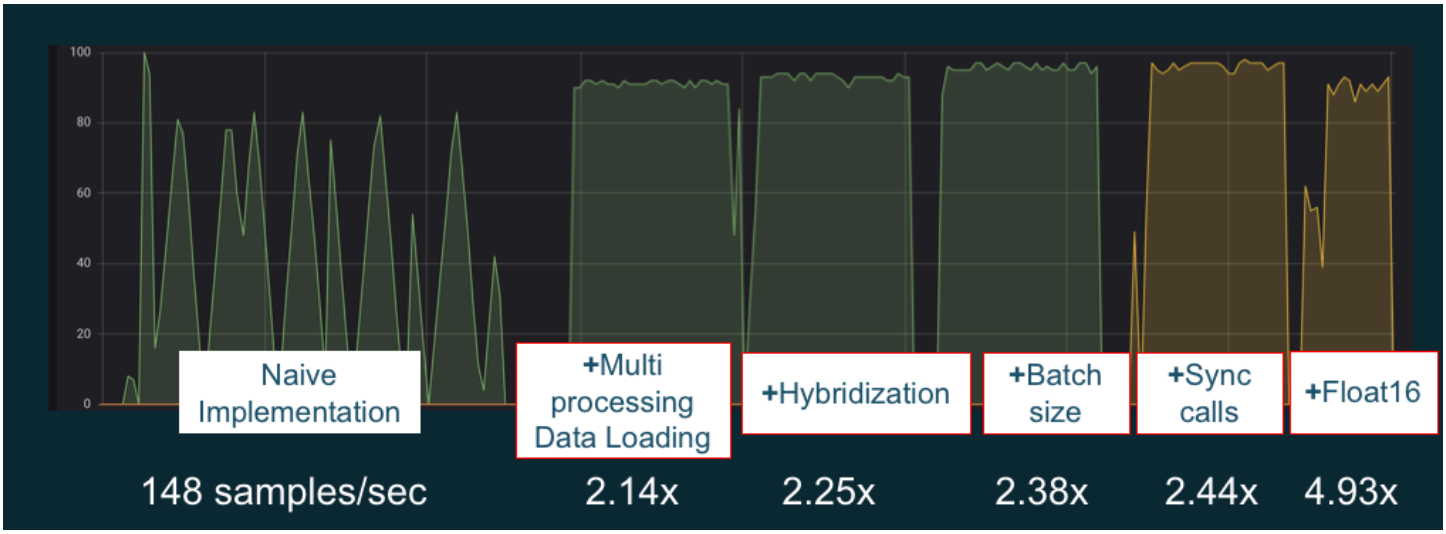
- Monitoramento
  - Monitor GPUs com CloudWatch
- Otimização

- Pré-processamento
- Treinamento

Otimização

Para aproveitar ao máximoGPUs, você pode otimizar seu pipeline de dados e ajustar sua rede de aprendizado profundo. Conforme descrito no gráfico a seguir, uma implementação ingênua ou básica de uma rede neural pode usá-la de GPU forma inconsistente e não atingir todo o seu potencial. Ao otimizar o pré-processamento e o carregamento de dados, você pode reduzir o gargalo do seu para o seuCPU. GPU Você pode ajustar a rede neural em si usando hibridização (quando compatível com a estrutura), ajustando o tamanho do lote e sincronizando as chamadas. Você também pode usar treinamento de precisão múltipla (float16 ou int8) na maioria das estruturas, o que pode causar um efeito enorme na melhoria da taxa de transferência.

A tabela a seguir mostra os ganhos de desempenho cumulativos ao aplicar otimizações diferentes. Seus resultados dependerão dos dados que você está processando e da rede que está otimizando.



Exemplo de otimizações de GPU desempenho. Fonte do gráfico: Truques de desempenho com MXNet Gluon

Os guias a seguir apresentam opções que funcionarão com você DLAMI e ajudarão você a melhorar o GPU desempenho.

Tópicos

- Pré-processamento
- Treinamento

Pré-processamento

O pré-processamento de dados por meio de transformações ou aumentos geralmente pode ser um processo CPU limitado, e isso pode ser o gargalo em seu pipeline geral. As estruturas têm operadores integrados para processamento de imagens, mas a DALI (Biblioteca de Aumento de Dados) demonstra um desempenho aprimorado em relação às opções integradas das estruturas.

- NVIDIABiblioteca de aumento de dados (DALI): DALI descarrega o aumento de dados para o. GPU Ele não está pré-instalado noDLAMI, mas você pode acessá-lo instalando-o ou carregando um contêiner de estrutura compatível na sua DLAMI ou em outra instância do Amazon Elastic Compute Cloud. Consulte a página DALI do projeto no NVIDIA site para obter detalhes. Para ver um exemplo de caso de uso e para baixar amostras de código, consulte a amostra de desempenho do treinamento SageMaker de pré-processamento.
- nvJPEG: uma biblioteca de GPU JPEG decodificadores acelerados para programadores C. Ele suporta a decodificação de imagens únicas ou lotes, bem como operações de transformação subsequentes que são comuns no aprendizado profundo. O nv JPEG vem embutido ou você pode fazer o download da página nvjpeg do NVIDIA site e usá-lo separadamente. DALI

Talvez você se interesse por esses outros tópicos sobre GPU monitoramento e otimização:

- Monitoramento
  - Monitor GPUs com CloudWatch
- Otimização
  - Pré-processamento
  - Treinamento

Treinamento

Com treinamento de precisão mista, você pode implantar redes maiores com a mesma quantidade de memória ou reduzir o uso de memória em comparação com sua rede de precisão única ou dupla, e você verá aumentos de desempenho de computação. Você também pode obter o benefício de transferências de dados menores e mais rápidas, um fator importante em um treinamento distribuído em vários nós. Para aproveitar o treinamento de precisão mista, você precisa ajustar a conversão de dados e a escalabilidade de perdas. Veja a seguir os guias que descrevem como fazer isso para as estruturas que oferecem suporte à precisão mista.

- NVIDIAAprendizado profundo SDK - documentos no NVIDIA site que descrevem a implementação de precisão mista paraMXNet, e. PyTorch TensorFlow

> **Tip**
>
> Verifique se o site oferece a estrutura de sua preferência e procure por "precisão mista" ou "fp16" para encontrar as técnicas de otimização mais recentes. Veja a seguir alguns guias sobre precisão mista que podem ser úteis:
>
> - Treinamento de precisão mista com TensorFlow (vídeo) - no NVIDIA site do blog.
> - Treinamento de precisão mista usando float16 com MXNet - um FAQ artigo no site. MXNet
> - NVIDIAApex: uma ferramenta para treinamento fácil de precisão mista com PyTorch - um artigo de blog no site. NVIDIA

Talvez você se interesse por esses outros tópicos sobre GPU monitoramento e otimização:

- Monitoramento
  - Monitor GPUs com CloudWatch
- Otimização
  - Pré-processamento
  - Treinamento

O chip de AWS inferência com DLAMI

AWS O Inferentia é um chip de aprendizado de máquina personalizado projetado por AWS ele que você pode usar para previsões de inferência de alto desempenho. Para usar o chip, configure uma instância do Amazon Elastic Compute Cloud e use o kit de desenvolvimento de software AWS Neuron (SDK) para invocar o chip Inferentia. Para oferecer aos clientes a melhor experiência de inferência, o Neuron foi incorporado ao AMIs de deep learning da AWS ()DLAMI.

Os tópicos a seguir mostram como começar a usar Inferentia com o. DLAMI

Conteúdo

- Iniciando uma DLAMI instância com o AWS Neuron
- Usando o DLAMI com AWS Neuron

Iniciando uma DLAMI instância com o AWS Neuron

O mais recente DLAMI está pronto para uso com AWS Inferentia e vem com o pacote AWS Neuron. API Para iniciar uma DLAMI instância, consulte Como iniciar e configurar uma DLAMI. Depois de ter umDLAMI, use as etapas aqui para garantir que seu chip de AWS inferência e os recursos do AWS Neuron estejam ativos.

Conteúdo

Verifique a instância

Antes de usar a instância, verifique se ela está corretamente definida e configurada com o Neuron.

Identificação de AWS dispositivos de inferência

Para identificar o número de dispositivos do Inferentia na sua instância, use o seguinte comando:

```
neuron-ls
```

Se a instância tiver dispositivos do Inferentia conectados a ela, a saída será semelhante à seguinte:

```
+--------+--------+--------+-----------+--------------+
| NEURON | NEURON | NEURON | CONNECTED |     PCI      |
| DEVICE | CORES  | MEMORY |  DEVICES  |     BDF      |
+--------+--------+--------+-----------+--------------+
| 0      | 4      | 8 GB   | 1         | 0000:00:1c.0 |
| 1      | 4      | 8 GB   | 2, 0      | 0000:00:1d.0 |
| 2      | 4      | 8 GB   | 3, 1      | 0000:00:1e.0 |
| 3      | 4      | 8 GB   | 2         | 0000:00:1f.0 |
+--------+--------+--------+-----------+--------------+
```

O resultado fornecido é obtido de uma instância Inf1.6xlarge e inclui as seguintes colunas:

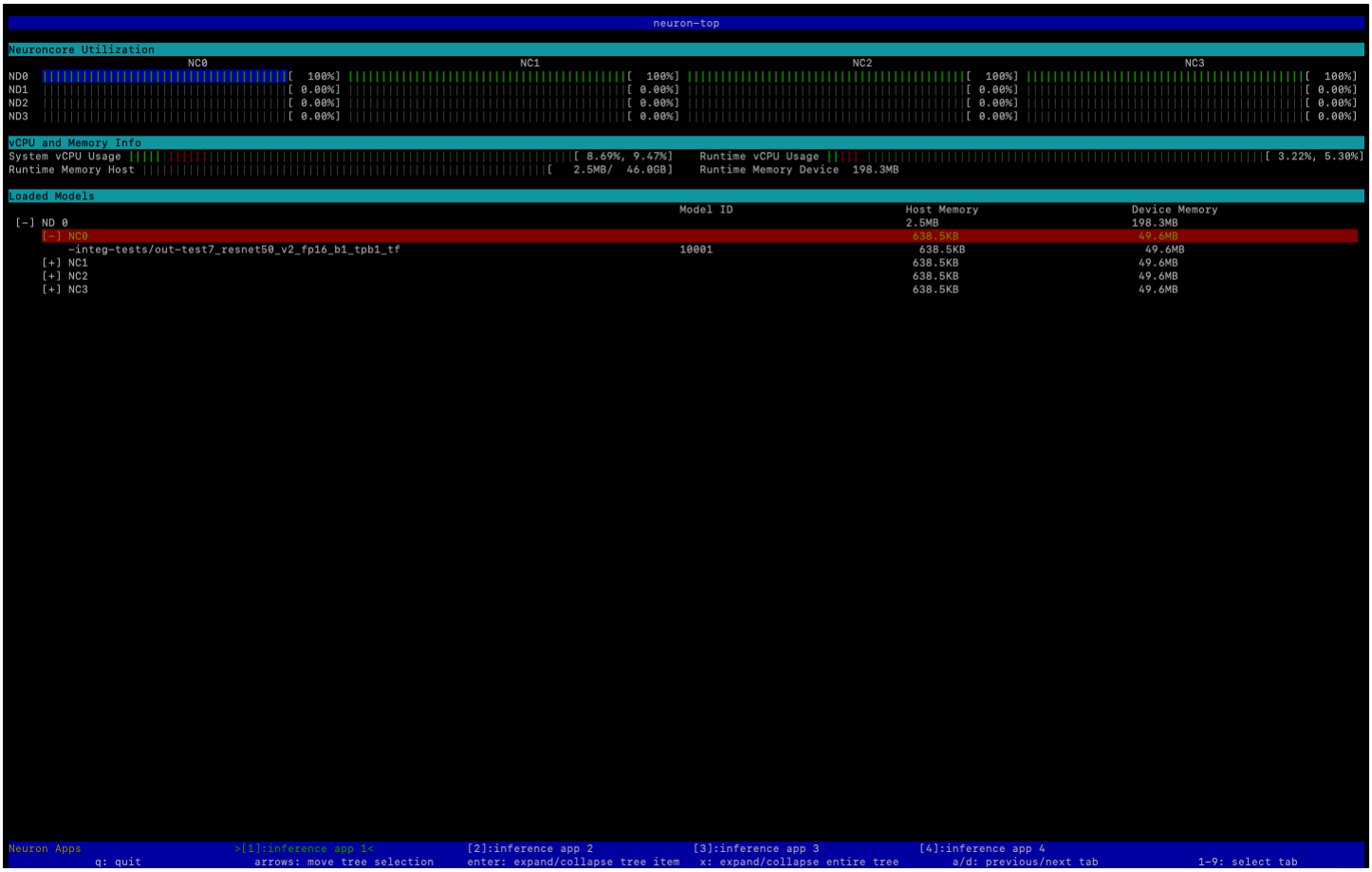
- NEURONDEVICE: a ID lógica atribuída ao NeuronDevice. Esse ID é usado ao configurar vários tempos de execução para usar diferentes. NeuronDevices
- NEURONCORES: O número de NeuronCores presentes no NeuronDevice.
- NEURONMEMORY: A quantidade de DRAM memória no NeuronDevice.
- CONNECTEDDEVICES: Outro NeuronDevices conectado ao NeuronDevice.
- PCIBDF: A função do dispositivo de PCI ônibus (BDF) ID do NeuronDevice.

Exibir o uso de recursos

Visualize informações úteis sobre a CPU utilização de NeuronCore and v, uso de memória, modelos carregados e aplicativos Neuron com o `neuron-top` comando. O lançamento `neuron-top` sem argumentos mostrará os dados de todos os aplicativos de aprendizado de máquina que utilizam NeuronCores.

```
neuron-top
```

Quando um aplicativo está usando quatro NeuronCores, a saída deve ser semelhante à imagem a seguir:



Para obter mais informações sobre recursos para monitorar e otimizar aplicações de inferência que usam como base o Neuron, consulte Ferramentas do Neuron.

Como usar o Monitor do Neuron

O Neuron Monitor coleta métricas dos tempos de execução do Neuron em execução no sistema e transmite os dados coletados para o formato stdout. JSON Elas são organizadas em grupos de métricas que você configura fornecendo um arquivo de configuração. Para obter mais informações sobre o Monitor do Neuron, consulte o Guia do usuário do monitor do Neuron.

Atualização do software Neuron

Para obter informações sobre como atualizar o SDK software NeuronDLAMI, consulte o Guia de configuração do AWS Neuron.

Próxima etapa

Usando o DLAMI com AWS Neuron

Usando o DLAMI com AWS Neuron

Um fluxo de trabalho típico com o AWS Neuron SDK é compilar um modelo de aprendizado de máquina previamente treinado em um servidor de compilação. Depois disso, distribua os artefatos para as instâncias Inf1 para execução. AMIs de deep learning da AWS (DLAMI) vem pré-instalado com tudo o que você precisa para compilar e executar inferência em uma instância Inf1 que usa Inferentia.

As seções a seguir descrevem como usar o DLAMI com inferência.

Conteúdo

Usando TensorFlow -Neuron e o compilador Neuron AWS

Este tutorial mostra como usar o compilador AWS Neuron para compilar o modelo Keras ResNet -50 e exportá-lo como um modelo salvo em formato. SavedModel Esse formato é um formato típico de TensorFlow modelo intercambiável. Você também aprenderá a executar a inferência em uma instância do Inf1 com exemplo de entrada.

Para obter mais informações sobre o NeuronSDK, consulte a documentação do AWS Neuron. SDK

Conteúdos

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em Iniciando uma DLAMI instância com o AWS Neuron. Você também deve estar familiarizado com o aprendizado profundo e com o uso doDLAMI.

Ative o ambiente Conda

Ative o ambiente TensorFlow -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_tensorflow_p36
```

Para sair do ambiente Conda atual, execute o seguinte comando:

```
source deactivate
```

Compilação ResNet50

Crie um script Python chamado **`tensorflow_compile_resnet50.py`** com o seguinte conteúdo. Esse script Python compila o modelo Keras ResNet 50 e o exporta como um modelo salvo.

```
import os
import time
import shutil
import tensorflow as tf
import tensorflow.neuron as tfn
import tensorflow.compat.v1.keras as keras
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input

# Create a workspace
WORKSPACE = './ws_resnet50'
os.makedirs(WORKSPACE, exist_ok=True)

# Prepare export directory (old one removed)
model_dir = os.path.join(WORKSPACE, 'resnet50')
compiled_model_dir = os.path.join(WORKSPACE, 'resnet50_neuron')
shutil.rmtree(model_dir, ignore_errors=True)
shutil.rmtree(compiled_model_dir, ignore_errors=True)

# Instantiate Keras ResNet50 model
keras.backend.set_learning_phase(0)
model = ResNet50(weights='imagenet')

# Export SavedModel
```

```
tf.saved_model.simple_save(
 session            = keras.backend.get_session(),
 export_dir         = model_dir,
 inputs             = {'input': model.inputs[0]},
 outputs            = {'output': model.outputs[0]})

# Compile using Neuron
tfn.saved_model.compile(model_dir, compiled_model_dir)

# Prepare SavedModel for uploading to Inf1 instance
shutil.make_archive(compiled_model_dir, 'zip', WORKSPACE, 'resnet50_neuron')
```

Compile o modelo usando o seguinte comando:

```
python tensorflow_compile_resnet50.py
```

O processo de compilação leva alguns minutos. Quando concluído, sua saída será semelhante a:

```
...
INFO:tensorflow:fusing subgraph neuron_op_d6f098c01c780733 with neuron-cc
INFO:tensorflow:Number of operations in TensorFlow session: 4638
INFO:tensorflow:Number of operations after tf.neuron optimizations: 556
INFO:tensorflow:Number of operations placed on Neuron runtime: 554
INFO:tensorflow:Successfully converted ./ws_resnet50/resnet50 to ./ws_resnet50/
resnet50_neuron
...
```

Após a compilação, o modelo salvo será compactado em **`ws_resnet50/resnet50_neuron.zip`**. Descompacte o modelo e faça download da imagem de exemplo para a inferência, usando os seguintes comandos:

```
unzip ws_resnet50/resnet50_neuron.zip -d .
curl -O https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg
```

ResNet50 Inferência

Crie um script Python chamado **`tensorflow_infer_resnet50.py`** com o seguinte conteúdo. Esse script executa a inferência no modelo obtido por download usando um modelo de inferência previamente compilado.

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import resnet50

# Create input from image
img_sgl = image.load_img('kitten_small.jpg', target_size=(224, 224))
img_arr = image.img_to_array(img_sgl)
img_arr2 = np.expand_dims(img_arr, axis=0)
img_arr3 = resnet50.preprocess_input(img_arr2)
# Load model
COMPILED_MODEL_DIR = './ws_resnet50/resnet50_neuron/'
predictor_inferentia = tf.contrib.predictor.from_saved_model(COMPILED_MODEL_DIR)
# Run inference
model_feed_dict={'input': img_arr3}
infa_rslts = predictor_inferentia(model_feed_dict);
# Display results
print(resnet50.decode_predictions(infa_rslts["output"], top=5)[0])
```

Execute a inferência no modelo usando o seguinte comando:

```
python tensorflow_infer_resnet50.py
```

A saída será semelhante a:

```
...
[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
 'snow_leopard', 0.009290541)]
```

Próxima etapa

Usando AWS Neuron Serving TensorFlow

Usando AWS Neuron Serving TensorFlow

Este tutorial mostra como construir um gráfico e adicionar uma etapa de compilação do AWS Neuron antes de exportar o modelo salvo para uso com o Serving. TensorFlow TensorFlow Serving é um sistema de atendimento que permite ampliar a inferência em uma rede. O Neuron TensorFlow Serving usa o mesmo que API o TensorFlow Serving normal. A única diferença é que um modelo salvo deve ser compilado para AWS Inferentia e o ponto de entrada é um binário diferente chamado`tensorflow_model_server_neuron`. O binário é encontrado em `/usr/local/bin/tensorflow_model_server_neuron` e está pré-instalado noDLAMI.

Para obter mais informações sobre o NeuronSDK, consulte a documentação do AWS Neuron. SDK

Conteúdos

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em Iniciando uma DLAMI instância com o AWS Neuron. Você também deve estar familiarizado com o aprendizado profundo e com o uso doDLAMI.

Ative o ambiente Conda

Ative o ambiente TensorFlow -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_tensorflow_p36
```

Se você precisar sair do ambiente Conda atual, execute:

```
source deactivate
```

Compile e exporte o modelo salvo

Crie um script Python chamado `tensorflow-model-server-compile.py` com o conteúdo a seguir. Ele constrói um gráfico e o compila usando o Neuron. Depois, exporta o gráfico compilado como modelo salvo.

```
import tensorflow as tf
import tensorflow.neuron
import os

tf.keras.backend.set_learning_phase(0)
model = tf.keras.applications.ResNet50(weights='imagenet')
sess = tf.keras.backend.get_session()
inputs = {'input': model.inputs[0]}
outputs = {'output': model.outputs[0]}

# save the model using tf.saved_model.simple_save
modeldir = "./resnet50/1"
tf.saved_model.simple_save(sess, modeldir, inputs, outputs)

# compile the model for Inferentia
neuron_modeldir = os.path.join(os.path.expanduser('~'), 'resnet50_inf1', '1')
tf.neuron.saved_model.compile(modeldir, neuron_modeldir, batch_size=1)
```

Compile o modelo usando o seguinte comando:

```
python tensorflow-model-server-compile.py
```

A saída será semelhante a:

```
...
INFO:tensorflow:fusing subgraph neuron_op_d6f098c01c780733 with neuron-cc
INFO:tensorflow:Number of operations in TensorFlow session: 4638
INFO:tensorflow:Number of operations after tf.neuron optimizations: 556
INFO:tensorflow:Number of operations placed on Neuron runtime: 554
INFO:tensorflow:Successfully converted ./resnet50/1 to /home/ubuntu/resnet50_inf1/1
```

Fornecer o modelo salvo

Depois que o modelo foi compilado, você pode usar o seguinte comando para fornecer o modelo salvo com o binário tensorflow_model_server_neuron:

```
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
    --model_base_path=$HOME/resnet50_inf1/ --port=8500 &
```

A saída será semelhante a: O modelo compilado é preparado no dispositivo Inferentia DRAM pelo servidor para se preparar para a inferência.

```
...
2019-11-22 01:20:32.075856: I external/org_tensorflow/tensorflow/cc/saved_model/
loader.cc:311] SavedModel load for tags { serve }; Status: success. Took 40764
 microseconds.
2019-11-22 01:20:32.075888: I tensorflow_serving/servables/tensorflow/
saved_model_warmup.cc:105] No warmup data file found at /home/ubuntu/resnet50_inf1/1/
assets.extra/tf_serving_warmup_requests
2019-11-22 01:20:32.075950: I tensorflow_serving/core/loader_harness.cc:87]
 Successfully loaded servable version {name: resnet50_inf1 version: 1}
2019-11-22 01:20:32.077859: I tensorflow_serving/model_servers/
server.cc:353] Running gRPC ModelServer at 0.0.0.0:8500 ...
```

Gerar solicitações de inferência para o modelo de servidor

Crie um script Python chamado `tensorflow-model-server-infer.py` com o conteúdo a seguir. Esse script executa inferência via gRPC, que é uma estrutura de serviço.

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
```

```
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        "./kitten_small.jpg",
        "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input'].CopyFrom(
        tf.contrib.util.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output'])
    print(decode_predictions(prediction))
```

Execute a inferência no modelo usando g RPC com o seguinte comando:

```
python tensorflow-model-server-infer.py
```

A saída será semelhante a:

```
[[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
 'snow_leopard', 0.009290541)]]
```

Usando MXNet -Neuron e o compilador Neuron AWS

A compilação MXNet -Neuron API fornece um método para compilar um gráfico de modelo que você pode executar em um dispositivo Inferentia. AWS

Neste exemplo, você usa o API para compilar um modelo ResNet -50 e usá-lo para executar inferência.

Para obter mais informações sobre o NeuronSDK, consulte a documentação do AWS Neuron. SDK

Conteúdos

- Pré-requisitos
- Ative o ambiente Conda
- Compilação ResNet50

- ResNet50 Inferência

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em Iniciando uma DLAMI instância com o AWS Neuron. Você também deve estar familiarizado com o aprendizado profundo e com o uso doDLAMI.

Ative o ambiente Conda

Ative o ambiente MXNet -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_mxnet_p36
```

Para sair do ambiente Conda atual, execute:

```
source deactivate
```

Compilação ResNet50

Crie um script Python chamado **`mxnet_compile_resnet50.py`** com o conteúdo a seguir. Este script usa a compilação MXNet -Neuron API Python para compilar um modelo -50. ResNet

```
import mxnet as mx
import numpy as np

print("downloading...")
path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
print("download finished.")

sym, args, aux = mx.model.load_checkpoint('resnet-50', 0)

print("compile for inferentia using neuron... this will take a few minutes...")
inputs = { "data" : mx.nd.ones([1,3,224,224], name='data', dtype='float32') }

sym, args, aux = mx.contrib.neuron.compile(sym, args, aux, inputs)
```

```
print("save compiled model...")
mx.model.save_checkpoint("compiled_resnet50", 0, sym, args, aux)
```

Compile o modelo usando o seguinte comando:

```
python mxnet_compile_resnet50.py
```

A compilação demora alguns minutos. Quando ela terminar, os seguintes arquivos estarão no diretório atual:

```
resnet-50-0000.params
resnet-50-symbol.json
compiled_resnet50-0000.params
compiled_resnet50-symbol.json
```

ResNet50 Inferência

Crie um script Python chamado **`mxnet_infer_resnet50.py`** com o conteúdo a seguir. Esse script faz download de uma imagem de amostra e a usa para executar a inferência com o modelo compilado.

```
import mxnet as mx
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'synset.txt')

fname = mx.test_utils.download('https://raw.githubusercontent.com/awslabs/mxnet-model-
server/master/docs/images/kitten_small.jpg')
img = mx.image.imread(fname)

# convert into format (batch, RGB, width, height)
img = mx.image.imresize(img, 224, 224)
# resize
img = img.transpose((2, 0, 1))
# Channel first
img = img.expand_dims(axis=0)
# batchify
img = img.astype(dtype='float32')
```

```
sym, args, aux = mx.model.load_checkpoint('compiled_resnet50', 0)
softmax = mx.nd.random_normal(shape=(1,))
args['softmax_label'] = softmax
args['data'] = img
# Inferentia context
ctx = mx.neuron()

exe = sym.bind(ctx=ctx, args=args, aux_states=aux, grad_req='null')
with open('synset.txt', 'r') as f:
    labels = [l.rstrip() for l in f]

exe.forward(data=img)
prob = exe.outputs[0].asnumpy()
# print the top-5
prob = np.squeeze(prob)
a = np.argsort(prob)[::-1]
for i in a[0:5]:
    print('probability=%f, class=%s' %(prob[i], labels[i]))
```

Execute a inferência com o modelo compilado usando o seguinte comando:

```
python mxnet_infer_resnet50.py
```

A saída será semelhante a:

```
probability=0.642454, class=n02123045 tabby, tabby cat
probability=0.189407, class=n02123159 tiger cat
probability=0.100798, class=n02124075 Egyptian cat
probability=0.030649, class=n02127052 lynx, catamount
probability=0.016278, class=n02129604 tiger, Panthera tigris
```

Próxima etapa

Usando o MXNet -Neuron Model Serving

Usando o MXNet -Neuron Model Serving

Neste tutorial, você aprende a usar um MXNet modelo pré-treinado para realizar a classificação de imagens em tempo real com o Multi Model Server ()MMS. MMSé uma easy-to-use ferramenta flexível para servir modelos de aprendizado profundo que são treinados usando qualquer estrutura

de aprendizado de máquina ou aprendizado profundo. Este tutorial inclui uma etapa de compilação usando o AWS Neuron e uma implementação do MMS uso. MXNet

Para obter mais informações sobre o NeuronSDK, consulte a documentação do AWS Neuron. SDK

Conteúdos

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em Iniciando uma DLAMI instância com o AWS Neuron. Você também deve estar familiarizado com o aprendizado profundo e com o uso doDLAMI.

Ative o ambiente Conda

Ative o ambiente MXNet -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_mxnet_p36
```

Para sair do ambiente Conda atual, execute:

```
source deactivate
```

Faça download do código de exemplo

Para executar o exemplo, faça download do código de exemplo usando os seguintes comandos:

```
git clone https://github.com/awslabs/multi-model-server
cd multi-model-server/examples/mxnet_vision
```

Compile o modelo.

Crie um script Python chamado `multi-model-server-compile.py` com o conteúdo a seguir. Esse script compila o modelo ResNet 50 para o alvo do dispositivo Inferentia.

```
import mxnet as mx
from mxnet.contrib import neuron
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
mx.test_utils.download(path+'synset.txt')

nn_name = "resnet-50"

#Load a model
sym, args, auxs = mx.model.load_checkpoint(nn_name, 0)

#Define compilation parameters#  - input shape and dtype
inputs = {'data' : mx.nd.zeros([1,3,224,224], dtype='float32') }

# compile graph to inferentia target
csym, cargs, cauxs = neuron.compile(sym, args, auxs, inputs)

# save compiled model
mx.model.save_checkpoint(nn_name + "_compiled", 0, csym, cargs, cauxs)
```

Para compilar o modelo, use o seguinte comando:

```
python multi-model-server-compile.py
```

A saída será semelhante a:

```
...
[21:18:40] src/nnvm/legacy_json_util.cc:209: Loading symbol saved by previous version
 v0.8.0. Attempting to upgrade...
[21:18:40] src/nnvm/legacy_json_util.cc:217: Symbol successfully upgraded!
[21:19:00] src/operator/subgraph/build_subgraph.cc:698: start to execute partition
 graph.
[21:19:00] src/nnvm/legacy_json_util.cc:209: Loading symbol saved by previous version
 v0.8.0. Attempting to upgrade...
[21:19:00] src/nnvm/legacy_json_util.cc:217: Symbol successfully upgraded!
```

Crie um arquivo chamado `signature.json` com o seguinte conteúdo para configurar o nome e a forma de entrada:

```
{
  "inputs": [
    {
      "data_name": "data",
      "data_shape": [
        1,
        3,
        224,
        224
      ]
    }
  ]
}
```

Faça download do arquivo `synset.txt` usando o seguinte comando: Esse arquivo é uma lista de nomes para classes de ImageNet predição.

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/
squeezenet_v1.1/synset.txt
```

Crie uma classe de serviço personalizada seguindo o modelo na pasta `model_server_template`. Copie o modelo para o diretório de trabalho atual usando o seguinte comando:

```
cp -r ../model_service_template/* .
```

Edite o módulo `mxnet_model_service.py` para substituir o contexto `mx.cpu()` pelo contexto `mx.neuron()`, da seguinte forma. Você também precisa comentar a cópia de dados desnecessária `model_input` porque MXNet -Neuron não suporta o NDArray e o Gluon. APIs

```
...
self.mxnet_ctx = mx.neuron() if gpu_id is None else mx.gpu(gpu_id)
...
#model_input = [item.as_in_context(self.mxnet_ctx) for item in model_input]
```

Empacote o modelo com arquivador de modelos, usando os seguintes comandos:

```
cd ~/multi-model-server/examples
model-archiver --force --model-name resnet-50_compiled --model-path mxnet_vision --
handler mxnet_vision_service:handle
```

Execute a inferência

Inicie o Multi Model Server e carregue o modelo que usa o RESTful API usando os seguintes comandos. Certifique-se de que o neuron-rtd está sendo executado com as configurações padrão.

```
cd ~/multi-model-server/
multi-model-server --start --model-store examples > /dev/null # Pipe to log file if you
 want to keep a log of MMS
curl -v -X POST "http://localhost:8081/models?
initial_workers=1&max_workers=4&synchronous=true&url=resnet-50_compiled.mar"
sleep 10 # allow sufficient time to load model
```

Execute a inferência usando uma imagem de exemplo com os seguintes comandos:

```
curl -O https://raw.githubusercontent.com/awslabs/multi-model-server/master/docs/
images/kitten_small.jpg
curl -X POST http://127.0.0.1:8080/predictions/resnet-50_compiled -T kitten_small.jpg
```

A saída será semelhante a:

```
[
  {
    "probability": 0.6388034820556641,
    "class": "n02123045 tabby, tabby cat"
  },
  {
    "probability": 0.16900072991847992,
    "class": "n02123159 tiger cat"
  },
  {
    "probability": 0.12221276015043259,
    "class": "n02124075 Egyptian cat"
  },
  {
    "probability": 0.028706775978207588,
    "class": "n02127052 lynx, catamount"
  },
  {
    "probability": 0.01915954425930977,
    "class": "n02129604 tiger, Panthera tigris"
  }
]
```

Para limpar após o teste, emita um comando delete por meio do RESTful API e interrompa o servidor do modelo usando os seguintes comandos:

```
curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled

multi-model-server --stop
```

A seguinte saída deverá ser mostrada:

```
{
  "status": "Model \"resnet-50_compiled\" unregistered"
}
Model server stopped.
Found 1 models and 1 NCGs.
Unloading 10001 (MODEL_STATUS_STARTED) :: success
Destroying NCG 1 :: success
```

Usando PyTorch -Neuron e o compilador Neuron AWS

A compilação PyTorch -Neuron API fornece um método para compilar um gráfico de modelo que você pode executar em um dispositivo Inferentia. AWS

Um modelo treinado deve ser compilado para um destino Inferentia antes que ele possa ser implantado em instâncias Inf1. O tutorial a seguir compila o modelo torchvision ResNet 50 e o exporta como um módulo salvo. TorchScript Esse modelo é, assim sendo, usado para executar inferência.

Por conveniência, este tutorial usa uma instância Inf1 para compilação e inferência. Na prática, você pode compilar o modelo usando outro tipo de instância, como a família de instâncias c5. Depois, você deve implantar o modelo compilado no servidor de inferência Inf1. Para obter mais informações, consulte a PyTorch SDKdocumentação do AWS neurônio.

Conteúdos

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em Iniciando uma DLAMI instância com o AWS Neuron. Você também deve estar familiarizado com o aprendizado profundo e com o uso doDLAMI.

Ative o ambiente Conda

Ative o ambiente PyTorch -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_pytorch_p36
```

Para sair do ambiente Conda atual, execute:

```
source deactivate
```

Compilação ResNet50

Crie um script Python chamado **`pytorch_trace_resnet50.py`** com o conteúdo a seguir. Este script usa a compilação PyTorch -Neuron API Python para compilar um modelo -50. ResNet

> **Note**
>
> Há uma dependência entre as versões do torchvision e do pacote torch que você deve conhecer ao compilar os modelos do torchvision. Essas regras de dependência podem ser gerenciadas por meio do pip. Torchvision==0.6.1 corresponde à versão torch==1.5.1, enquanto torchvision==0.8.2 corresponde à versão torch==1.7.1.

```
import torch
import numpy as np
import os
import torch_neuron
from torchvision import models

image = torch.zeros([1, 3, 224, 224], dtype=torch.float32)

## Load a pretrained ResNet50 model
```

```
model = models.resnet50(pretrained=True)

## Tell the model we are using it for evaluation (not training)
model.eval()
model_neuron = torch.neuron.trace(model, example_inputs=[image])

## Export to saved model
model_neuron.save("resnet50_neuron.pt")
```

Execute o script de compilação.

```
python pytorch_trace_resnet50.py
```

A compilação demora alguns minutos. Quando terminar, o modelo compilado será salvo como `resnet50_neuron.pt` no diretório local.

ResNet50 Inferência

Crie um script Python chamado **`pytorch_infer_resnet50.py`** com o conteúdo a seguir. Esse script faz download de uma imagem de amostra e a usa para executar a inferência com o modelo compilado.

```
import os
import time
import torch
import torch_neuron
import json
import numpy as np

from urllib import request

from torchvision import models, transforms, datasets

## Create an image directory containing a small kitten
os.makedirs("./torch_neuron_test/images", exist_ok=True)
request.urlretrieve("https://raw.githubusercontent.com/awslabs/mxnet-model-server/
master/docs/images/kitten_small.jpg",
                    "./torch_neuron_test/images/kitten_small.jpg")


## Fetch labels to output the top classifications
```

```
request.urlretrieve("https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json","imagenet_class_index.json")
idx2label = []

with open("imagenet_class_index.json", "r") as read_file:
    class_idx = json.load(read_file)
    idx2label = [class_idx[str(k)][1] for k in range(len(class_idx))]

## Import a sample image and normalize it into a tensor
normalize = transforms.Normalize(
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])

eval_dataset = datasets.ImageFolder(
    os.path.dirname("./torch_neuron_test/"),
    transforms.Compose([
    transforms.Resize([224, 224]),
    transforms.ToTensor(),
    normalize,
    ])
)

image, _ = eval_dataset[0]
image = torch.tensor(image.numpy()[np.newaxis, ...])

## Load model
model_neuron = torch.jit.load( 'resnet50_neuron.pt' )

## Predict
results = model_neuron( image )

# Get the top 5 results
top5_idx = results[0].sort()[1][-5:]

# Lookup and print the top 5 labels
top5_labels = [idx2label[idx] for idx in top5_idx]

print("Top 5 labels:\n {}".format(top5_labels) )
```

Execute a inferência com o modelo compilado usando o seguinte comando:

```
python pytorch_infer_resnet50.py
```

A saída será semelhante a:

```
Top 5 labels:
 ['tiger', 'lynx', 'tiger_cat', 'Egyptian_cat', 'tabby']
```

O ARM64 DLAMI

AWS ARM64GPUDLAMIssão projetados para fornecer alto desempenho e economia para cargas de trabalho de aprendizado profundo. Especificamente, o tipo de instância G5g apresenta o processador AWS Graviton2 baseado em ARM64, que foi desenvolvido do zero AWS e otimizado para a forma como os clientes executam suas cargas de trabalho na nuvem. AWS ARM64GPUDLAMIssão pré-configurados com Docker, NVIDIA Docker, NVIDIA Driver, CuCUDA,, DNNNCCL, bem como estruturas populares de aprendizado de máquina, como e. TensorFlow PyTorch

Com o tipo de instância G5g, você pode aproveitar os benefícios de preço e desempenho do Graviton2 para implantar modelos de aprendizado profundo GPU acelerados a um custo significativamente menor em comparação com instâncias baseadas em x86 com aceleração. GPU

Selecione um ARM64 DLAMI

Execute uma instância G5g com a ARM64 DLAMI de sua escolha.

Para step-by-step obter instruções sobre como iniciar umDLAMI, consulte Iniciando e configurando um DLAMI.

Para obter uma lista dos mais recentes ARM64DLAMIs, consulte as notas de lançamento doDLAMI.

Conceitos básicos

Os tópicos a seguir mostram como começar a usar ARM64 DLAMI o.

Conteúdo

- Usando o ARM64 GPU PyTorch DLAMI

Usando o ARM64 GPU PyTorch DLAMI

O AMIs de deep learning da AWS está pronto para uso com processador Arm64 baseado em processador e vem GPUs otimizado para. PyTorch ARM64GPU PyTorch DLAMIIsso inclui um

ambiente Python pré-configurado com PyTorch, TorchVision, e TorchServepara casos de uso de treinamento e inferência de aprendizado profundo.

Conteúdo

Verifique o PyTorch ambiente Python

Conecte-se à sua instância G5g e ative o ambiente básico do Conda com o seguinte comando:

```
source activate base
```

Seu prompt de comando deve indicar que você está trabalhando no ambiente básico do Conda, que contém PyTorch TorchVision, e outras bibliotecas.

```
(base) $
```

Verifique os caminhos de ferramentas padrão do PyTorch ambiente:

```
(base) $ which python
(base) $ which pip
(base) $ which conda
(base) $ which mamba
>>> import torch, torchvision
>>> torch.__version__
>>> torchvision.__version__
>>> v = torch.autograd.Variable(torch.randn(10, 3, 224, 224))
>>> v = torch.autograd.Variable(torch.randn(10, 3, 224, 224)).cuda()
>>> assert isinstance(v, torch.Tensor)
```

Execute uma amostra de treinamento com PyTorch

Execute um exemplo de trabalho MNIST de treinamento:

```
git clone https://github.com/pytorch/examples.git
cd examples/mnist
```

```
python main.py
```

Sua saída deve ser semelhante à seguinte:

```
...
Train Epoch: 14 [56320/60000 (94%)]    Loss: 0.021424
Train Epoch: 14 [56960/60000 (95%)]    Loss: 0.023695
Train Epoch: 14 [57600/60000 (96%)]    Loss: 0.001973
Train Epoch: 14 [58240/60000 (97%)]    Loss: 0.007121
Train Epoch: 14 [58880/60000 (98%)]    Loss: 0.003717
Train Epoch: 14 [59520/60000 (99%)]    Loss: 0.001729
Test set: Average loss: 0.0275, Accuracy: 9916/10000 (99%)
```

Execute uma amostra de inferência com PyTorch

Use os comandos a seguir para baixar um modelo densenet161 pré-treinado e executar a inferência usando: TorchServe

```
# Set up TorchServe
cd $HOME
git clone https://github.com/pytorch/serve.git
mkdir -p serve/model_store
cd serve

# Download a pre-trained densenet161 model
wget https://download.pytorch.org/models/densenet161-8d451a50.pth >/dev/null

# Save the model using torch-model-archiver
torch-model-archiver --model-name densenet161 \
    --version 1.0 \
    --model-file examples/image_classifier/densenet_161/model.py \
    --serialized-file densenet161-8d451a50.pth \
    --handler image_classifier \
    --extra-files examples/image_classifier/index_to_name.json  \
    --export-path model_store

# Start the model server
torchserve --start --no-config-snapshots \
    --model-store model_store \
    --models densenet161=densenet161.mar &> torchserve.log

# Wait for the model server to start
sleep 30
```

```
# Run a prediction request
curl http://127.0.0.1:8080/predictions/densenet161 -T examples/image_classifier/
kitten.jpg
```

Sua saída deve ser semelhante à seguinte:

```
{
  "tiger_cat": 0.4693363308906555,
  "tabby": 0.4633873701095581,
  "Egyptian_cat": 0.06456123292446136,
  "lynx": 0.0012828150065615773,
  "plastic_bag": 0.00023322898778133094
}
```

Use os comandos a seguir para cancelar o registro do modelo densenet161 e parar o servidor:

```
curl -X DELETE http://localhost:8081/models/densenet161/1.0
torchserve --stop
```

Sua saída deve ser semelhante à seguinte:

```
{
  "status": "Model \"densenet161\" unregistered"
}
TorchServe has stopped.
```

Inferência

Esta seção fornece tutoriais sobre como executar inferência usando as estruturas e ferramentas DLAMI do.

Ferramentas de inferência

- TensorFlow Servindo

Fornecimento de modelos

A seguir estão as opções de fornecimento de modelos instaladas no Deep Learning AMI with Conda. Clique em uma das opções para saber como usá-la.

Tópicos

TensorFlow Servindo

TensorFlow O Serving é um sistema de atendimento flexível e de alto desempenho para modelos de aprendizado de máquina.

O `tensorflow-serving-api` vem pré-instalado com Deep Learning AMI with Conda! Você encontrará exemplos de scripts para treinar, exportar e servir um MNIST modelo`~/examples/tensorflow-serving/`.

Para executar qualquer um desses exemplos, primeiro conecte-se ao seu Deep Learning AMI com o Conda e ative o TensorFlow ambiente.

```
$ source activate tensorflow2_p310
```

Agora, altere os diretórios para a pasta de scripts de exemplo de fornecimento.

```
$ cd ~/examples/tensorflow-serving/
```

Fornecer um modelo de origem pré-treinado

Veja a seguir um exemplo que você pode seguir para fornecer diferentes modelos como origem. Como regra geral, você precisa que um modelo útil e scripts de cliente já estejam baixados para o seuDLAMI.

Forneça e teste a inferência com um modelo de início

1. Faça download do modelo.

   ```
   $ curl -O https://s3-us-west-2.amazonaws.com/tf-test-models/INCEPTION.zip
   ```

2. Descompacte o modelo.

   ```
   $ unzip INCEPTION.zip
   ```

3. Faça download de uma imagem de um husky.

```
$ curl -O https://upload.wikimedia.org/wikipedia/commons/b/b5/Siberian_Husky_bi-eyed_Flickr.jpg
```

4. Inicie o servidor. Observe que, para o Amazon Linux, você deve alterar o diretório usado para `model_base_path`, de `/home/ubuntu` para `/home/ec2-user`.

```
$ tensorflow_model_server --model_name=INCEPTION --model_base_path=/home/ubuntu/examples/tensorflow-serving/INCEPTION/INCEPTION --port=9000
```

5. Com o servidor em execução em primeiro plano, é necessário iniciar outra sessão de terminal para continuar. Abra um novo terminal e ative TensorFlow com`source activate tensorflow2_p310`. Em seguida, use um editor de texto de sua preferência para criar um script com o conteúdo a seguir. Chame-o de `inception_client.py`. Esse script usará um nome de arquivo de imagem como um parâmetro e obterá o resultado da previsão do modelo pré-treinado.

```
from __future__ import print_function

import grpc
import tensorflow as tf
import argparse

from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc

parser = argparse.ArgumentParser(
    description='TF Serving Test',
    formatter_class=argparse.ArgumentDefaultsHelpFormatter
)
parser.add_argument('--server_address', default='localhost:9000',
                    help='Tenforflow Model Server Address')
parser.add_argument('--image', default='Siberian_Husky_bi-eyed_Flickr.jpg',
                    help='Path to the image')
args = parser.parse_args()


def main():
  channel = grpc.insecure_channel(args.server_address)
  stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
  # Send request
```

```
  with open(args.image, 'rb') as f:
    # See prediction_service.proto for gRPC request/response details.
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'INCEPTION'
    request.model_spec.signature_name = 'predict_images'

    input_name = 'images'
    input_shape = [1]
    input_data = f.read()
    request.inputs[input_name].CopyFrom(
      tf.make_tensor_proto(input_data, shape=input_shape))

    result = stub.Predict(request, 10.0)  # 10 secs timeout
    print(result)

  print("Inception Client Passed")


if __name__ == '__main__':
  main()
```

6. Agora execute o script passando o local do servidor, a porta e o nome do arquivo da fotografia do husky como parâmetros.

```
$ python3 inception_client.py --server=localhost:9000 --image Siberian_Husky_bi-
eyed_Flickr.jpg
```

Treine e sirva um MNIST modelo

Neste tutorial, vamos exportar um modelo e, em seguida, fornecê-lo com o aplicativo `tensorflow_model_server`. Finalmente, você pode testar o servidor modelo com um exemplo de script do cliente.

Execute o script que treinará e exportará um MNIST modelo. Como único argumento do script, você precisa fornecer uma pasta local para salvar o modelo. Por enquanto, podemos colocá-lo em `mnist_model`. O script criará a pasta para você.

```
$ python mnist_saved_model.py /tmp/mnist_model
```

Aguarde. Este script pode demorar um pouco antes de fornecer resultados. Ao concluir o treinamento e, por fim, exportar o modelo, você verá o seguinte:

```
Done training!
Exporting trained model to mnist_model/1
Done exporting!
```

A próxima etapa é executar o `tensorflow_model_server` para fornecer o modelo exportado.

```
$ tensorflow_model_server --port=9000 --model_name=mnist --model_base_path=/tmp/
mnist_model
```

Um script de cliente é disponibilizado para testar o servidor.

Para testá-lo, você precisa abrir uma nova janela no terminal.

```
$ python mnist_client.py --num_tests=1000 --server=localhost:9000
```

Outros recursos e exemplos

Se você estiver interessado em saber mais sobre o TensorFlow Serving, confira o TensorFlow site.

Você também pode usar o TensorFlow Serving with Amazon Elastic Inference. Confira o guia sobre como usar o Elastic Inference with TensorFlow Serving para obter mais informações.

TorchServe

TorchServe é uma ferramenta flexível para servir modelos de aprendizado profundo que foram exportados do PyTorch. TorchServe vem pré-instalado com o Deep Learning AMI with Conda.

Para obter mais informações sobre o uso TorchServe, consulte Model Server para PyTorch documentação.

Tópicos

Ofereça um modelo de classificação de imagens em TorchServe

Este tutorial mostra como servir um modelo de classificação de imagens com TorchServe o. Ele usa um modelo DenseNet -161 fornecido pela. PyTorch Quando o servidor está em execução, ele escuta as solicitações de previsão. Quando você carrega uma imagem, neste caso, uma imagem de um gatinho, o servidor retorna uma estimativa das cinco principais classes correspondentes das classes em que o modelo foi treinado.

Para fornecer um exemplo de modelo de classificação de imagens em TorchServe

1. Conecte-se a uma instância do Amazon Elastic Compute Cloud (AmazonEC2) com o Deep Learning AMI com o Conda v34 ou posterior.
2. Ative o ambiente `pytorch_p310`.

```
source activate pytorch_p310
```

3. Clone o TorchServe repositório e crie um diretório para armazenar seus modelos.

```
git clone https://github.com/pytorch/serve.git
mkdir model_store
```

4. Arquive o modelo usando o arquivador de modelos. O `extra-files` parâmetro usa um arquivo do `TorchServe` repositório, portanto, atualize o caminho, se necessário. Para obter mais informações sobre o arquivador de modelos, consulte Torch Model archiver for. TorchServe

```
wget https://download.pytorch.org/models/densenet161-8d451a50.pth
torch-model-archiver --model-name densenet161 --version 1.0 --model-file ./
serve/examples/image_classifier/densenet_161/model.py --serialized-file
 densenet161-8d451a50.pth --export-path model_store --extra-files ./serve/examples/
image_classifier/index_to_name.json --handler image_classifier
```

5. Execute TorchServe para iniciar um endpoint. A adição de `> /dev/null` silencia a saída do log.

```
torchserve --start --ncs --model-store model_store --models densenet161.mar > /dev/
null
```

6. Baixe uma imagem de um gatinho e envie-a para o endpoint de TorchServe previsão:

```
curl -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
curl http://127.0.0.1:8080/predictions/densenet161 -T kitten.jpg
```

O ponto final da previsão retorna uma previsão JSON semelhante às cinco principais previsões a seguir, em que a imagem tem 47% de probabilidade de conter um gato egípcio, seguida por 46% de chance de ter um gato malhado.

```
{
 "tiger_cat": 0.46933576464653015,
```

```
 "tabby": 0.463387668132782,
 "Egyptian_cat": 0.0645613968372345,
 "lynx": 0.0012828196631744504,
 "plastic_bag": 0.00023323058849200606
}
```

7. Ao terminar o teste, interrompa o servidor.

```
torchserve --stop
```

Outros exemplos

TorchServe tem vários exemplos que você pode executar na sua DLAMI instância. Você pode visualizá-los na página de exemplos TorchServe do repositório do projeto.

Mais informações

Para obter mais TorchServe documentação, incluindo como configurar TorchServe o Docker e os TorchServe recursos mais recentes, consulte a página do TorchServe projeto em GitHub.

Atualizando seu DLAMI

Aqui você encontrará informações sobre como atualizar seu DLAMI e dicas sobre como atualizar o software no seuDLAMI.

Sempre mantenha o sistema operacional e outros softwares instalados atualizados executando patches e atualizações assim que forem disponibilizados.

Se você estiver usando Amazon Linux ou Ubuntu, ao fazer login no seuDLAMI, você será notificado se houver atualizações disponíveis e verá as instruções de atualização. Para obter mais informações sobre a manutenção do Amazon Linux, consulte Atualizar o software de instância. Para instâncias do Ubuntu, consulte a Documentação oficial do Ubuntu.

No Windows, verifique o Windows Update regularmente para instalar atualizações de software e de segurança. Se você preferir, as atualizações podem ser instaladas automaticamente.

> **Important**
>
> Para obter informações sobre as vulnerabilidades Meltdown e Spectre e como corrigir seu sistema operacional para resolvê-las, consulte o Boletim de Segurança -2018-013. AWS

Tópicos

Atualizando para a nova versão da DLAMI

DLAMIAs imagens do sistema da são atualizadas regularmente para aproveitar os novos lançamentos da estrutura de aprendizado profundo, CUDA outras atualizações de software e ajustes de desempenho. Se você estiver usando um DLAMI há algum tempo e quiser aproveitar uma atualização, precisará iniciar uma nova instância. Também seria necessário transferir manualmente todos os conjuntos de dados, pontos de verificação ou outros dados importantes. Em vez disso, você pode usar EBS a Amazon para reter seus dados e anexá-los a um novoDLAMI. Dessa forma, você pode fazer atualizações com frequência e, ao mesmo tempo, minimizar o tempo necessário para transferir os dados.

> **Note**
>
> Ao conectar e mover EBS volumes da Amazon entre DLAMIs eles, você deve ter o volume DLAMIs e o novo na mesma zona de disponibilidade.

1. Use a Amazon EC2console para criar um novo EBS volume da Amazon. Para obter instruções detalhadas, consulte Criação de um EBS volume na Amazon.
2. Anexe seu volume recém-criado da Amazon ao seu EBS volume existenteDLAMI. Para obter instruções detalhadas, consulte Anexar um EBS volume da Amazon.
3. Transfira seus dados, como conjuntos de dados, pontos de verificação e arquivos de configuração.
4. Lance umDLAMI. Para obter instruções detalhadas, consulte Configurando uma DLAMI instância.
5. Separe o EBS volume da Amazon do seu antigoDLAMI. Para obter instruções detalhadas, consulte Separar um EBS volume da Amazon.
6. Anexe o EBS volume da Amazon ao seu novoDLAMI. Siga as instruções da etapa 2 para associar o volume.
7. Depois de verificar se os dados estão disponíveis no novoDLAMI, interrompa e encerre o antigoDLAMI. Para obter instruções detalhadas para limpeza, consulte Limpando uma DLAMI instância.

Dicas para as atualizações de software

De tempos em tempos, talvez você queira atualizar manualmente o software no seuDLAMI. É recomendável usar `pip` para atualizar pacotes Python. Você também deve usar `pip` para atualizar pacotes em um ambiente Conda no Deep Learning AMI with Conda. Consulte o site do software ou do framework específico para obter instruções de atualização e instalação.

> **Note**
>
> Não podemos garantir que uma atualização de pacote será bem-sucedida. A tentativa de atualizar um pacote em um ambiente com dependências incompatíveis pode resultar em uma falha. Nesse caso, você deve entrar em contato com o responsável pela biblioteca para conferir se é possível atualizar as dependências do pacote. Como alternativa, é possível

tentar modificar o ambiente para permitir a atualização. No entanto, essa modificação provavelmente significará remover ou atualizar os pacotes existentes, ou seja, não poderemos mais garantir a estabilidade desse ambiente.

AMIs de deep learning da AWS Ele vem com muitos ambientes Conda e muitos pacotes pré-instalados. Devido ao número de pacotes pré-instalados, é difícil encontrar um conjunto de pacotes com garantia de compatibilidade. Você pode ver um aviso “O ambiente é inconsistente, verifique o plano do pacote com cuidado”. DLAMIgarante que todos os ambientes DLAMI fornecidos estejam corretos, mas não pode garantir que nenhum pacote instalado pelo usuário funcione corretamente.

Receba notificações sobre novas atualizações

Note

AWS O Deep Learning AMIs tem uma cadência semanal de lançamentos de patches de segurança. Notificações de lançamento serão enviadas para esses patches de segurança incrementais, embora eles possam não estar incluídos nas notas oficiais de lançamento.

Você pode receber notificações sempre que uma nova DLAMI for lançada. As notificações são publicadas com a Amazon SNS usando o tópico a seguir.

```
arn:aws:sns:us-west-2:767397762724:dlami-updates
```

As mensagens são postadas aqui quando uma nova DLAMI é publicada. A versão, os metadados e as AMI IDs regionais do AMI serão incluídos na mensagem.

Essas mensagens podem ser recebidas usando vários métodos diferentes. Recomendamos que você use o método a seguir.

1. Abra o SNSconsole da Amazon.
2. Na barra de navegação, altere a AWS Região para Oeste dos EUA (Oregon), se necessário. Você deve selecionar a região em que a SNS notificação na qual você está se inscrevendo foi criada.
3. No painel de navegação, escolha Assinaturas, Criar assinatura.
4. Na caixa de diálogo Create subscription, faça o seguinte:

a. Para Tópico ARN, copie e cole o seguinte nome de recurso da Amazon (ARN): **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**

b. Para Protocolo, escolha um entre [AmazonSQS, AWS Lamda, Email, Email- JSON]

c. Para Endpoint, insira o endereço de e-mail ou o Amazon Resource Name (ARN) do recurso que você usará para receber as notificações.

d. Selecione Create subscription.

5. Você recebe um e-mail de confirmação com o assunto AWS Notificação - Confirmação de assinatura. Abra o e-mail e escolha Confirm subscription para concluir a assinatura.

Segurança em AMIs de deep learning da AWS

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que é executada Serviços da AWS no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade que se aplicam AMIs de deep learning da AWS, consulte AWS Serviços no escopo do programa de conformidade AWS .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS service (Serviço da AWS) que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usarDLAMI. Os tópicos a seguir mostram como configurar para atender DLAMI aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros Serviços da AWS que o ajudem a monitorar e proteger seus DLAMI recursos.

Para obter mais informações, consulte Segurança na Amazon EC2 no Guia do EC2 usuário da Amazon.

Tópicos

Proteção de dados em AMIs de deep learning da AWS

O modelo de responsabilidade AWS compartilhada modelo se aplica à proteção de dados em AMIs de deep learning da AWS. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Privacidade de dados FAQ. Para obter informações sobre proteção de dados na Europa, consulte o Modelo de Responsabilidade AWS Compartilhada e GDPR a postagem no blog AWS de segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use a autenticação multifator (MFA) com cada conta.
- UseSSL/TLSpara se comunicar com AWS os recursos. Exigimos TLS 1,2 e recomendamos TLS 1,3.
- Configure API e registre as atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de FIPS 140-3 módulos criptográficos validados ao acessar AWS por meio de uma interface de linha de comando ou umaAPI, use um endpoint. FIPS Para obter mais informações sobre os FIPS endpoints disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com DLAMI ou Serviços da AWS usa o console,API, AWS CLI, ou AWS SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre

usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é altamente recomendável que você não inclua informações de credenciais no URL para validar sua solicitação para esse servidor.

Gerenciamento de identidade e acesso para AMIs de deep learning da AWS

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. IAMos administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar DLAMI os recursos. IAMé um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Para obter mais informações sobre gerenciamento de identidade e acesso, consulte Gerenciamento de identidade e acesso para a Amazon EC2.

Tópicos

- Autenticando com identidades
- Gerenciando acesso usando políticas
- IAMcom a Amazon EMR

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como IAM usuário ou assumindo uma IAM função. Usuário raiz da conta da AWS

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Os usuários (do IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você entra como uma identidade federada, seu administrador configurou previamente a federação de identidades usando IAM funções. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para você mesmo assinar solicitações, consulte Assinar AWS API solicitações no Guia IAM do usuário.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia AWS IAM Identity Center do usuário e Uso da autenticação multifator (MFA) AWS no Guia do IAMusuário.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para ver a lista completa de tarefas que exigem que você faça login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do IAM usuário.

Grupos e usuários do IAM

Um IAMusuário é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos confiar em credenciais temporárias em vez de criar IAM usuários que tenham credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com IAM os usuários, recomendamos que você alterne as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exigem credenciais de longo prazo no Guia do IAMusuário.

Um IAMgrupo é uma identidade que especifica uma coleção de IAM usuários. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar IAM recursos.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um IAM usuário (em vez de uma função) no Guia do IAM usuário.

IAMfunções

Uma IAMfunção é uma identidade dentro da sua Conta da AWS que tem permissões específicas. É semelhante a um IAM usuário, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma IAM função no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma AWS API operação AWS CLI or ou usando uma personalizadaURL. Para obter mais informações sobre métodos de uso de funções, consulte Métodos para assumir uma função no Guia IAM do usuário.

IAMfunções com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter informações sobre funções para federação, consulte Criação de uma função para um provedor de identidade terceirizado no Guia IAM do usuário. Se você usa o IAM Identity Center, configura um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a uma função em. IAM Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Manual do Usuário do AWS IAM Identity Center .
- Permissões temporárias IAM de IAM usuário — Um usuário ou função pode assumir uma IAM função para assumir temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas — Você pode usar uma IAM função para permitir que alguém (um diretor confiável) em uma conta diferente acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso

usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.

- Sessões de acesso direto (FAS) — Quando você usa um IAM usuário ou uma função para realizar ações em AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.
- Função de serviço — Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a uma AWS service (Serviço da AWS) no Guia do IAM usuário.
- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.

- Aplicativos em execução na Amazon EC2 — Você pode usar uma IAM função para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo AWS CLI AWS API solicitações. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Como usar uma IAM função para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia IAM do usuário.

Para saber se usar IAM funções ou IAM usuários, consulte Quando criar uma IAM função (em vez de um usuário) no Guia do IAM usuário.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas

permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como JSON documentos. Para obter mais informações sobre a estrutura e o conteúdo dos documentos de JSON política, consulte Visão geral das JSON políticas no Guia IAM do usuário.

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

IAMas políticas definem permissões para uma ação, independentemente do método usado para realizar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função do AWS Management Console AWS CLI, do ou do AWS API.

Políticas baseadas em identidade

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolha entre políticas gerenciadas e políticas em linha no Guia do IAMusuário.

Políticas baseadas no recurso

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são as políticas de confiança de IAM funções e as

políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas de uma política baseada IAM em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Amazon S3, AWS WAF, e Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber maisACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões — Um limite de permissões é um recurso avançado no qual você define as permissões máximas que uma política baseada em identidade pode conceder a uma IAM entidade (IAMusuário ou função). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para IAM entidades no Guia IAM do usuário.
- Políticas de controle de serviço (SCPs) — SCPs são JSON políticas que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as

suas contas. Os SCP limites de permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations eSCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.

- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia IAM do usuário.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de política estão envolvidos, consulte Lógica de avaliação de políticas no Guia IAM do usuário.

IAMcom a Amazon EMR

Você pode usar IAM com EMR a Amazon para definir usuários, AWS recursos, grupos, funções e políticas. Você também pode controlar Serviços da AWS quais usuários e funções podem acessar.

Para obter mais informações sobre como usar IAM com a AmazonEMR, consulte AWS Identity and Access Management para a Amazon EMR.

Validação de conformidade para AMIs de deep learning da AWS

Auditores terceirizados avaliam a segurança e a conformidade AMIs de deep learning da AWS como parte de vários programas de AWS conformidade. Para obter informações sobre os programas de conformidade suportados, consulte Validação de conformidade para a Amazon EC2.

Para obter uma lista do Serviços da AWS escopo de programas de conformidade específicos, consulte AWS Serviços no escopo do programa de conformidade AWS . Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixando relatórios no AWS Artifact em. AWS Artifact

Sua responsabilidade de conformidade ao usar DLAMI é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido de segurança e compatibilidade: estes guias de implantação abordam as considerações de arquitetura e fornecem etapas para implantação de ambientes de linha de base focados em compatibilidade e segurança na AWS.
- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- Avaliação de recursos com AWS Config regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar seus AWS recursos e verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

Resiliência em AMIs de deep learning da AWS

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data centers tradicionais.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte Infraestrutura AWS global.

Para obter informações sobre os EC2 recursos da Amazon para ajudar a suportar suas necessidades de resiliência e backup de dados, consulte Resiliência EC2 na Amazon no Guia EC2 do usuário da Amazon.

Segurança da infraestrutura em AMIs de deep learning da AWS

A segurança da infraestrutura do AMIs de deep learning da AWS é apoiada pela AmazonEC2. Para obter mais informações, consulte Segurança da infraestrutura na Amazon EC2 no Guia EC2 do usuário da Amazon.

AMIs de deep learning da AWS Instâncias de monitoramento

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho da sua AMIs de deep learning da AWS instância e de suas outras AWS soluções. Sua DLAMI instância vem com várias ferramentas de GPU monitoramento, incluindo um utilitário que reporta estatísticas GPU de uso para a Amazon CloudWatch. Para obter mais informaçõesGPUMonitoramento e otimização, consulte e consulte Monitorar EC2 recursos da Amazon no Guia EC2 do usuário da Amazon.

Optar por não acompanhar o uso de instâncias DLAMI

As distribuições de sistema AMIs de deep learning da AWS operacional a seguir incluem código que permite AWS coletar informações sobre o tipo de instância, o ID da instância, o DLAMI tipo e o sistema operacional.

> **Note**
>
> AWS não coleta nem retém nenhuma outra informação sobre oDLAMI, como os comandos que você usa noDLAMI.

- Amazon Linux 2
- Ubuntu 20.04
- Ubuntu 18.04
- Ubuntu 16.04

Para desativar o rastreamento de uso

Se preferir, você pode desativar o rastreamento de uso de uma nova DLAMI instância. Para optar por não participar, você deve adicionar uma tag à sua EC2 instância da Amazon durante o lançamento. A tag deve usar a chave `OPT_OUT_TRACKING` com o valor associado definido como `true`. Para

obter mais informações, consulte Marcar seus EC2 recursos da Amazon no Guia EC2 do usuário da Amazon.

DLAMIpolítica de suporte estrutural

Aqui você pode encontrar detalhes da política de suporte para AMIs de deep learning da AWS (DLAMI) estruturas.

Para obter uma lista das DLAMI estruturas que AWS atualmente suporta, consulte a página DLAMIFramework Support Policy. Nas tabelas dessa página, lembre-se do seguinte:

- A versão atual especifica a versão da estrutura no formato x.y.z. Nesse formato, x se refere à versão principal, y se refere à versão secundária e z se refere à versão do patch. Por exemplo, para TensorFlow 2.10.1, a versão principal é 2, a versão secundária é 10 e a versão do patch é 1.
- O fim do patch especifica por quanto tempo AWS suporta essa versão da estrutura.

Para obter informações detalhadas sobre questões específicasDLAMIs, consulteNotas da versão da DLAMIs.

DLAMIsuporte de estrutura FAQs

- Quais versões da estrutura recebem patches de segurança?
- O que as imagens fazem AWS publicar quando novas versões do framework forem lançadas?
- Quais imagens ficam novas SageMaker/AWS características?
- Como a versão atual é definida na tabela de Estruturas compatíveis?
- E se eu executar uma versão que não está na tabela Estruturas compatíveis?
- Oferece DLAMIs suporte às versões anteriores do TensorFlow?
- Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?
- Com que frequência novas imagens são lançadas?
- Minha instância receberá um patch enquanto a workload estiver em execução?
- O que acontece quando uma nova versão com patch aplicado ou atualizada da estrutura está disponível?
- As dependências são atualizadas sem alterar a versão da estrutura?
- Quando o suporte ativo para minha versão da estrutura termina?

- As imagens com versões de estrutura que não são mais mantidas ativamente receberão um patch?
- Como usar uma versão de estrutura mais antiga?
- Como faço para up-to-date acompanhar as mudanças de suporte nas estruturas e suas versões?
- Preciso de uma licença comercial para usar o repositório Anaconda?

Quais versões da estrutura recebem patches de segurança?

Se a versão da estrutura estiver rotulada como Compatível no AMIs de deep learning da AWS Tabela de políticas de suporte do Framework, ela recebe patches de segurança.

O que as imagens fazem AWS publicar quando novas versões do framework forem lançadas?

Publicamos novas versões DLAMIs logo após o lançamento TensorFlow e o lançamento PyTorch de novas versões. Isso inclui versões principais, versões principais e secundárias e major-minor-patch versões de estruturas. Também atualizamos as imagens quando novas versões de drivers e bibliotecas são disponibilizadas. Para obter mais informações sobre a manutenção da imagem, consulte Quando o suporte ativo para minha versão da estrutura termina?

Quais imagens ficam novas SageMaker/AWS características?

Os novos recursos geralmente são lançados na versão mais recente do DLAMIs for PyTorch TensorFlow e. Consulte as notas de lançamento de uma imagem específica para obter detalhes sobre novas SageMaker ou AWS características. Para obter uma lista dos disponíveisDLAMIs, consulte as Notas de versão doDLAMI. Para obter mais informações sobre a manutenção da imagem, consulte Quando o suporte ativo para minha versão da estrutura termina?

Como a versão atual é definida na tabela de Estruturas compatíveis?

A versão atual no AMIs de deep learning da AWS A tabela Framework Support Policy se refere à versão mais recente da estrutura que AWS disponibiliza em GitHub. Cada versão mais recente inclui atualizações para os drivers, bibliotecas e pacotes relevantes noDLAMI. Para obter informações sobre a manutenção de imagens, consulte Quando o suporte ativo para minha versão da estrutura termina?

E se eu executar uma versão que não está na tabela Estruturas compatíveis?

Se você estiver executando uma versão que não está no AMIs de deep learning da AWS Na tabela Framework Support Policy, talvez você não tenha os drivers, bibliotecas e pacotes relevantes mais atualizados. Para uma up-to-date versão adicional, recomendamos que você atualize para uma das estruturas suportadas disponíveis usando a versão mais recente DLAMI de sua escolha. Para obter uma lista dos disponíveisDLAMIs, consulte as Notas de versão doDLAMI.

Oferece DLAMIs suporte às versões anteriores do TensorFlow?

Não. Oferecemos suporte à versão de patch mais recente da versão principal mais recente de cada estrutura, lançada 365 dias após o GitHub lançamento inicial, conforme declarado na AMIs de deep learning da AWS Tabela de políticas de suporte do Framework. Para ter mais informações, consulte E se eu executar uma versão que não está na tabela Estruturas compatíveis?

Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?

Para usar a DLAMI com a versão mais recente da estrutura, recupere o DLAMIID e use-o para iniciá-lo DLAMI usando o EC2console. Para amostra AWS CLIcomandos para recuperar o AMIs de deep learning da AWS ID, consulte as notas de DLAMI lançamento da página de notas de DLAMIlançamento de estrutura única. A versão da estrutura que você escolher deve ser rotulada como Suportada no AMIs de deep learning da AWS Tabela de políticas de suporte do Framework.

Com que frequência novas imagens são lançadas?

Nossa maior prioridade é fornecer versões de patch atualizadas. Criamos rotineiramente imagens com patch aplicado na primeira oportunidade. Monitoramos as versões recém-corrigidas da estrutura (ex. TensorFlow 2.9 a TensorFlow 2.9.1) e novas versões secundárias (ex. TensorFlow 2.9 a TensorFlow 2.10) e disponibilize-os o mais rápido possível. Quando uma versão existente do TensorFlow é lançada com uma nova versão doCUDA, lançamos uma nova DLAMI para essa versão do TensorFlow com suporte para a nova CUDA versão.

Minha instância receberá um patch enquanto a workload estiver em execução?

Não. As atualizações de patch para não DLAMI são atualizações “no local”.

Você deve ativar uma nova EC2 instância, migrar suas cargas de trabalho e scripts e, em seguida, desativar sua instância anterior.

O que acontece quando uma nova versão com patch aplicado ou atualizada da estrutura está disponível?

Verifique regularmente a imagem na página das notas de versão. É recomendado atualizar para novas estruturas com patch aplicado ou mais recentes quando estiverem disponíveis. Para obter uma lista dos disponíveisDLAMIs, consulte as Notas de versão doDLAMI.

As dependências são atualizadas sem alterar a versão da estrutura?

Atualizamos as dependências sem alterar a versão da estrutura. No entanto, se uma atualização de dependência causar uma incompatibilidade, criaremos uma imagem com uma versão diferente. Certifique-se de verificar as Notas de versão DLAMI para obter informações atualizadas sobre dependências.

Quando o suporte ativo para minha versão da estrutura termina?

DLAMIas imagens são imutáveis. Depois de criadas, elas não mudam. Há quatro motivos principais para o fim do suporte ativo de uma versão da estrutura:

- Atualizações da versão da estrutura (patch)
- AWS patches de segurança
- Data de fim do patch (descontinuação)
- Dependência end-of-support

> **Note**
>
> Devido à frequência de atualizações de patches de versão e patches de segurança, recomendamos verificar a página de notas de lançamento DLAMI com frequência e atualizar quando forem feitas alterações.

Atualizações da versão da estrutura (patch)

Se você tem uma DLAMI carga de trabalho baseada na TensorFlow 2.7.0 e TensorFlow libera a versão 2.7.1 ativada, então GitHub AWS lança um novo DLAMI com TensorFlow 2.7.1. As imagens

anteriores com 2.7.0 não são mais mantidas ativamente depois que a nova imagem com TensorFlow 2.7.1 é lançada. O DLAMI with TensorFlow 2.7.0 não recebe mais patches. A página DLAMI de notas de lançamento do TensorFlow 2.7 é então atualizada com as informações mais recentes. Não há uma página individual das notas de versão para cada patch secundário.

Os novos DLAMIs criados devido a atualizações de patches são designados com um novo AMIID.

AWS patches de segurança

Se você tiver uma carga de trabalho baseada em uma imagem com TensorFlow 2.7.0 e AWS faz um patch de segurança e, em seguida, uma nova versão do DLAMI é lançada para a TensorFlow 2.7.0. A versão anterior das imagens com TensorFlow 2.7.0 não é mais mantida ativamente. Para obter mais informações, consulte Minha instância receberá um patch enquanto a workload estiver em execução? Para obter as etapas para encontrar as mais recentesDLAMI, consulte Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?

Os novos DLAMIs criados devido a atualizações de patches são designados com um novo AMIID.

Data de fim do patch (descontinuação)

DLAMIsatingiram a data final do patch 365 dias após a data GitHub de lançamento.

Para várias estruturas DLAMIs, quando uma das versões da estrutura é atualizada, é necessária uma nova DLAMI com a versão atualizada. A versão DLAMI com a estrutura antiga não é mais mantida ativamente.

> **Important**
>
> A exceção é quando há uma grande atualização da estrutura. Por exemplo, se a versão TensorFlow 1.15 for atualizada para TensorFlow 2.0, continuaremos a oferecer suporte à versão mais recente da TensorFlow 1.15 por um período de dois anos a partir da data de GitHub lançamento ou seis meses após a equipe de manutenção da estrutura de origem cancelar o suporte, qualquer que seja a data anterior.

Dependência end-of-support

Se você estiver executando uma carga de trabalho em uma imagem TensorFlow 2.7.0 DLAMI com o Python 3.6 e essa versão do Python estiver marcada como sim, end-of-support todas as imagens

DLAMI baseadas no Python 3.6 não serão mais mantidas ativamente. Da mesma forma, se uma versão do sistema operacional como o Ubuntu 16.04 estiver marcada para end-of-support, todas as DLAMI imagens que dependem do Ubuntu 16.04 não serão mais mantidas ativamente.

As imagens com versões de estrutura que não são mais mantidas ativamente receberão um patch?

Não. As imagens que não são mais mantidas ativamente não terão novos lançamentos.

Como usar uma versão de estrutura mais antiga?

Para usar um DLAMI com uma versão mais antiga da estrutura, recupere o DLAMIID e use-o para iniciá-lo DLAMI usando o EC2console. Para AWS CLIcomandos para recuperar o AMI ID, consulte a página de notas de versão nas notas de DLAMIversão de estrutura única.

Como faço para up-to-date acompanhar as mudanças de suporte nas estruturas e suas versões?

Fique up-to-date com DLAMI estruturas e versões usando o AMIs de deep learning da AWS Tabela de políticas de suporte do Framework, as notas de DLAMI lançamento.

Preciso de uma licença comercial para usar o repositório Anaconda?

O Anaconda mudou para um modelo de licenciamento comercial com foco em determinados usuários. DLAMIsAs mantidas ativamente foram migradas para a versão de código aberto disponível ao público do Conda (conda-forge) do canal Anaconda.

Alterações importantes NVIDIA do driver em DLAMIs

Em 15 de novembro de 2023, AWS fez mudanças importantes em AMIs de deep learning da AWS (DLAMI) relacionado ao NIVIDA driver que DLAMIs usa. Para obter informações sobre o que mudou e se isso afeta seu uso deDLAMIs, consulteDLAMINVIDIAmudança de motorista FAQs.

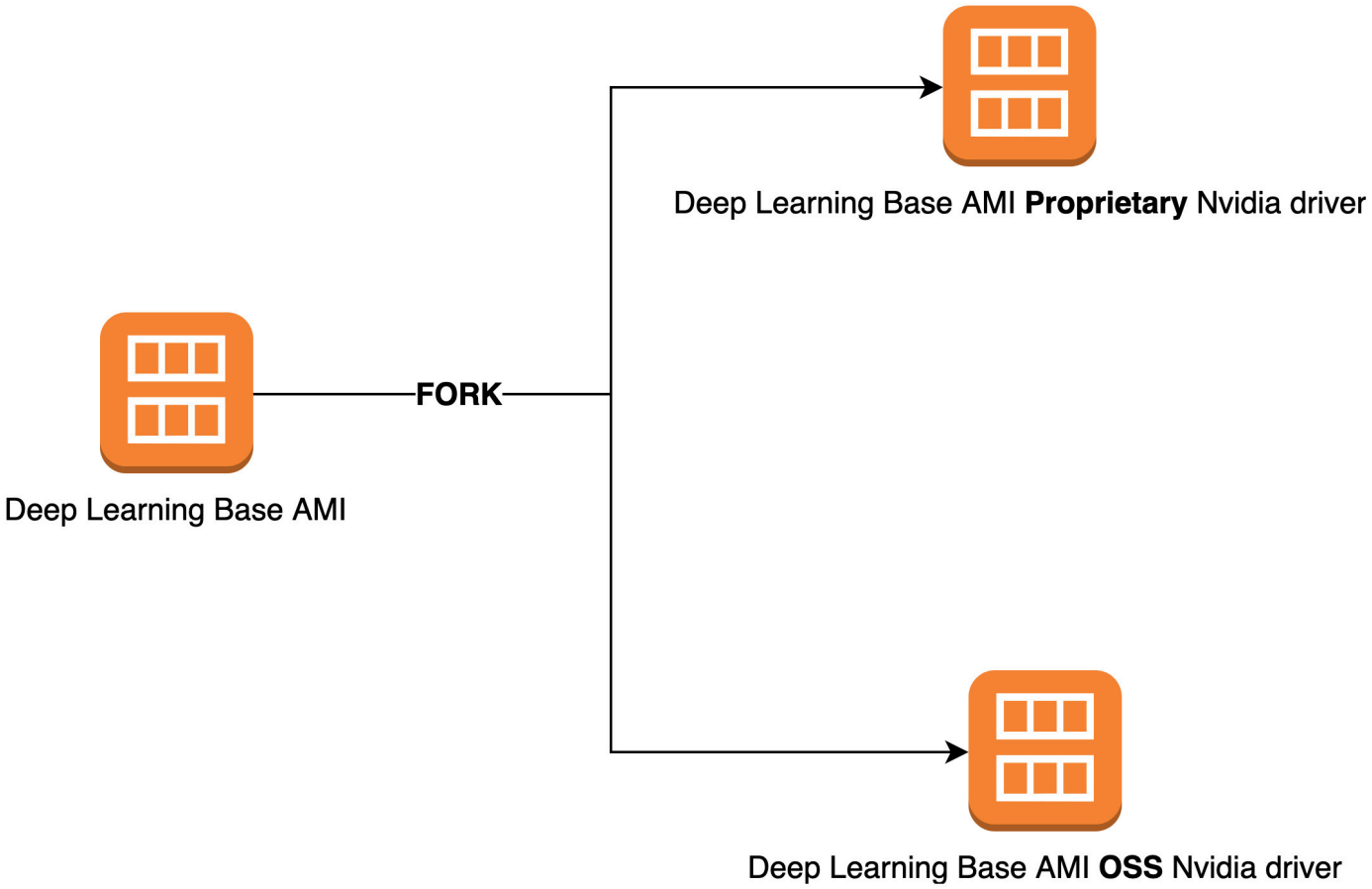
DLAMINVIDIAmudança de motorista FAQs

O que mudou?

Nós nos DLAMIs dividimos em dois grupos separados:

- DLAMIsque usam driver NVIDIA proprietário (para suportar P3, P3dn, G3)
- DLAMIsque usam NVIDIA OSS driver (para suportar G4dn, G5, P4, P5)

Como resultado, criamos novos DLAMIs para cada uma das duas categorias com novos nomes e novos AMIIDs. Eles não DLAMIs são intercambiáveis. Ou seja, DLAMIs de um grupo não oferecem suporte a instâncias que o outro grupo suporta. Por exemplo, o DLAMI que suporta P5 não suporta G3, e o DLAMI que suporta G3 não suporta P5.



Por que essa mudança foi necessária?

Anteriormente, DLAMIs NVIDIA GPUs incluía um driver de kernel proprietário do. NVIDIA No entanto, a comunidade upstream do kernel Linux aceitou uma mudança que isola os drivers proprietários do kernel, como o NVIDIA GPU driver, da comunicação com outros drivers do kernel. Essa alteração é desativada GPUDirect RDMA nas instâncias das séries P4 e P5, que é o mecanismo que permite o uso eficiente GPUs para treinamento distribuídoEFA. Como resultado, DLAMIs agora use o driver OpenRM (driver de código NVIDIA aberto), vinculado aos EFA drivers de código aberto para suportar G4dn, G5, P4 e P5. No entanto, esse driver OpenRM não oferece suporte a instâncias mais antigas (como P3 e G3). Portanto, para garantir que continuemos fornecendo suporte atual, seguro DLAMIs e de alto desempenho para os dois tipos de instância, DLAMIs dividimos em dois grupos: um com o driver OpenRM (compatível com G4dn, G5, P4 e P5) e outro com o driver proprietário mais antigo (compatível com P3, P3dn e G3).

O DLAMIs que essa mudança afetou?

Essa mudança afetou a todosDLAMIs.

O que isso significa para você?

Todos DLAMIs continuarão fornecendo funcionalidade, desempenho e segurança, desde que você os execute em um tipo de instância compatível do Amazon Elastic Compute Cloud (AmazonEC2). Para determinar os tipos de EC2 instância DLAMI compatíveis com a, verifique as notas de versão eDLAMI, em seguida, procure por EC2Instâncias suportadas. Para obter uma lista das DLAMI opções atualmente suportadas e links para suas notas de versão, consulteNotas da versão da DLAMIs.

Além disso, você deve usar o correto AWS Command Line Interface (AWS CLI) comandos para invocar a correnteDLAMIs.

Para bases DLAMIs que suportam P3, P3dn e G3, use este comando:

```
aws ec2 describe-images --region us-east-1 --owners amazon \
--filters 'Name=name,Values=Deep Learning Base Proprietary Nvidia Driver AMI (Amazon
 Linux 2) Version ??.?' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' --output text
```

Para bases DLAMIs que suportam G4dn, G5, P4 e P5, use este comando:

```
aws ec2 describe-images --region us-east-1 --owners amazon \
--filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2)
 Version ??.?' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' --output text
```

Há alguma perda de funcionalidade com o mais novoDLAMIs?

Não, não há perda de funcionalidade. As atuais DLAMIs fornecem todas as funcionalidades, desempenho e segurança das anterioresDLAMIs, desde que você as execute em um tipo de EC2 instância compatível.

Essa mudança afetou os Deep Learning Containers?

Não, essa mudança não afetou AWS Deep Learning Containers, porque eles não incluem o NVIDIA driver. No entanto, certifique-se de executar Deep Learning Containers AMIs que sejam compatíveis com as instâncias subjacentes.

Informações relacionadas sobre DLAMI

Você pode encontrar outros recursos com informações relacionadas DLAMI fora do AMIs de deep learning da AWS Guia do desenvolvedor Ativado AWS re:Post, confira as perguntas DLAMI de outros clientes ou faça suas próprias perguntas. Sobre o AWS Blog de Machine Learning e outros AWS blogs, leia postagens oficiais sobreDLAMI.

AWS re:Post

Etiqueta: AMIs de deep learning da AWS

AWS Blog

- AWS Blog de Machine Learning | Categoria: AMIs de deep learning da AWS
- AWS Blog de Machine Learning | Treinamento mais rápido com TensorFlow 1.6 otimizado nas instâncias EC2 C5 e P3 da Amazon
- AWS Blog de Machine Learning | Novo AMIs de deep learning da AWS para profissionais de Machine Learning
- AWS Partner Network (APN) Blog | Novos cursos de treinamento disponíveis: Introdução ao Machine Learning e ao Deep Learning em AWS
- AWS Blog de notícias | Viagem ao aprendizado profundo com AWS

Notas da versão da DLAMIs

Aqui você pode encontrar notas de lançamento detalhadas para todas as opções atualmente suportadas AMIs de deep learning da AWS (DLAMI).

Para ver as notas de lançamento de DLAMI estruturas para as quais não oferecemos mais suporte, consulte a seção Arquivo de notas de lançamento do Unsupported Framework na página Framework DLAMISupport Policy.

> **Note**
>
> Eles AMIs de deep learning da AWS têm uma cadência de lançamento noturno de patches de segurança. Não incluímos esses patches de segurança incrementais nas notas de lançamento oficiais.

Base DLAMIs

GPU

- X86
  - AWS Base de aprendizado profundo AMI (Amazon Linux 2)
  - AWS Base de aprendizado profundo AMI (Ubuntu 22.04)
  - AWS Base de aprendizado profundo AMI (Ubuntu 20.04)
- ARM64
  - AWS Base de aprendizado profundo ARM64 AMI (Ubuntu 22.04)
  - AWS Base de aprendizado profundo ARM64 AMI (Amazon Linux 2)

AWS Neuron

- X86
  - AWS AMINeurônio básico de aprendizado profundo (Amazon Linux 2)
  - AWS AMINeurônio básico de aprendizado profundo (Ubuntu 20.04)

Qualcomm

- X86
  - AWS Base de aprendizado profundo Qualcomm (AMIAmazon Linux 2)

Estrutura única DLAMIs

PyTorch-específico AMIs

GPU

- X86
  - AWS Aprendizado profundo AMI GPU PyTorch 2.4 (Ubuntu 22.04)
  - AWS Aprendizado profundo AMI GPU PyTorch 2.3 (Ubuntu 20.04)
  - AWS Aprendizado profundo AMI GPU PyTorch 2.3 (Amazon Linux 2)
  - AWS Aprendizado profundo AMI GPU PyTorch 2.2 (Ubuntu 20.04)
  - AWS Aprendizado profundo AMI GPU PyTorch 2.2 (Amazon Linux 2)
  - AWS Aprendizado profundo AMI GPU PyTorch 1.13 (Amazon Linux 2)
  - AWS Aprendizado profundo AMI GPU PyTorch 1.13 (Ubuntu 20.04)
- ARM64
  - AWS Aprendizado profundo ARM64 AMI GPU PyTorch 2.4 (Ubuntu 22.04)
  - AWS Aprendizado profundo ARM64 AMI GPU PyTorch 2.3 (Ubuntu 22.04)
  - AWS Aprendizado profundo ARM64 AMI GPU PyTorch 2.2 (Ubuntu 20.04)

AWS Neuron

- X86
  - AWS Deep Learning AMI Neuron PyTorch 1.13 (Amazon Linux 2)
  - AWS AMINeurônio de aprendizagem profunda PyTorch 1.13 (Ubuntu 20.04)

TensorFlow-específico AMIs

GPU

- X86
  - AWS Aprendizado profundo AMI GPU TensorFlow 2.17 (Ubuntu 22.04)

- AWS Aprendizado profundo AMI GPU TensorFlow 2.16 (Amazon Linux 2)
- AWS Aprendizado profundo AMI GPU TensorFlow 2.16 (Ubuntu 20.04)
- AWS Aprendizado profundo AMI GPU TensorFlow 2.15 (Amazon Linux 2)
- AWS Aprendizado profundo AMI GPU TensorFlow 2.15 (Ubuntu 20.04)

AWS Neuron

- X86
  - AWS Deep Learning AMI Neuron TensorFlow 2.10 (Amazon Linux 2)
  - AWS AMINeurônio de aprendizagem profunda TensorFlow 2.10 (Ubuntu 20.04)

Estrutura múltipla DLAMIs

> **Tip**
>
> Se você usa apenas uma estrutura de aprendizado de máquina, recomendamos uma estrutura única DLAMI.

GPU

- X86
  - AWS Aprendizado profundo AMI (Amazon Linux 2)

AWS Neuron

- X86
  - AWS AMINeurônio de aprendizado profundo (Amazon Linux 2023)
  - AWS AMINeurônio de aprendizado profundo (Ubuntu 22.04)

Recursos obsoletos do DLAMI

A tabela a seguir lista os recursos obsoletos do AMIs de deep learning da AWS (DLAMI), a data em que os descontinuamos e detalhes sobre por que os descontinuamos.

Atributo	Data	Detalhes
Ubuntu 16.04	10/07/2021	O Ubuntu Linux 16.04 LTS chegou ao fim de sua LTS janela de cinco anos em 30 de abril de 2021 e não é mais suportado por seu fornecedor. Não há mais atualizações na Base de Aprendizado Profundo AMI (Ubuntu 16.04) em novas versões a partir de outubro de 2021. As versões anteriores continuarão disponíveis.
Amazon Linux	10/07/2021	O Amazon Linux está end-of-lifeem dezembro de 2020. Não há mais atualizações para o Deep Learning AMI (Amazon Linux) em novas versões a partir de outubro de 2021. As versões anteriore s do Deep Learning AMI (Amazon Linux) continuarão disponíveis.
Chainer	01/07/2020	A Chainer anunciou o fim dos grandes lançamentos a partir de dezembro de 2019. Consequentemente,

Atributo	Data	Detalhes
		não incluiremos mais os ambientes Chainer Conda no DLAMI início de julho de 2020. As versões anteriores do DLAMI que contêm esses ambientes continuarão disponíveis. Nós só forneceremos atualizações para esses ambientes se houver correções de segurança publicadas pela comunidade de código aberto para essas estruturas de trabalho.
Python 3.6	15/06/2020	Devido a solicitações dos clientes, estamos mudando para o Python 3.7 para as novas versões TF/MX/PT.
Python 2	01/01/2020	A comunidade de código aberto Python encerrou oficialmente o suporte a Python 2. As MXNet comunidades TensorFlow, PyTorch, e também anunciaram que as versões TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4 e MXNet 1.6.0 serão as últimas a oferecer suporte ao Python 2.

Histórico da documentação do DLAMI

A tabela a seguir fornece um histórico dos DLAMI lançamentos recentes e das alterações relacionadas ao Guia do AMIs de deep learning da AWS desenvolvedor.

Mudanças recentes

Alteração	Descrição	Data
ARM64 DLAMI	O AMIs de deep learning da AWS agora suporta imagens baseadas no processador GPUs Arm64.	29 de novembro de 2021
TensorFlow 2	O Deep Learning AMI com Conda agora vem com TensorFlow 2 com CUDA 10.	3 de dezembro de 2019
AWS Inferência	O Deep Learning AMI agora oferece suporte ao hardware AWS Inferentia e ao AWS SDK Neuron.	3 de dezembro de 2019
Usando o TensorFlow Serving com um modelo Inception	Um exemplo de uso de inferência com um modelo Inception foi adicionado para TensorFlow Serving, tanto com quanto sem Elastic Inference.	28 de novembro de 2018
Instalando PyTorch a partir de uma compilação noturna	Foi adicionado um tutorial que aborda como você pode desinstalar e PyTorch, em seguida, instalar uma versão noturna do PyTorch seu Deep Learning AMI com o Conda.	25 de setembro de 2018

Tutorial do Conda	O exemplo MOTD foi atualizado para refletir uma versão mais recente.	23 de julho de 2018

Alterações anteriores

A tabela a seguir fornece um histórico de DLAMI versões anteriores e alterações relacionadas antes de julho de 2018.

Alteração	Descrição	Data
TensorFlow com Horovod	Foi adicionado um tutorial para treinar ImageNet com TensorFlow e Horovod.	6 de junho de 2018
Guia de atualização	Adicionado o guia de atualização.	15 de maio de 2018
Novas regiões e novo tutorial de 10 minutos	Novas regiões adicionadas: Oeste dos EUA (Norte da Califórnia), América do Sul, Canadá (Central), UE (Londres) e UE (Paris). Além disso, a primeira versão de um tutorial de 10 minutos intitulado: “Introdução ao aprendizado profundoAMI”.	26 de abril de 2018
Tutorial do Chainer	Um tutorial para usar o Chainer nos CPU modos múltiplo GPUGPU, único e foi adicionado. CUDAa integração foi atualizada de CUDA 8 para CUDA 9 para várias estruturas.	28 de fevereiro de 2018

Alteração	Descrição	Data
Linux AMIs v3.0, além da introdução do MXNet Model Server, TensorFlow Serving e TensorBoard	Tutoriais adicionados para o Conda AMIs com novos recursos de exibição de modelos e visualizações usando o MXNet Model Server v0.1.5, TensorFlow Serving v1.4.0 e v0.4.0. TensorBoard AMIe CUDA recursos de estrutura descritos em Conda e visões CUDA gerais. Notas de versão mais recentes movidas para https://aws.amazon.com/releasenotes/	25 de janeiro de 2018
Linux AMIs v2.0	Base, Source e Conda AMIs atualizados com NCCL 2.1. Source e Conda AMIs atualizados com MXNet v1.0, PyTorch 0.3.0 e Keras 2.0.9.	11 de dezembro de 2017
Duas AMI opções do Windows adicionadas	Windows 2012 R2 e 2016 AMIs lançados: adicionados ao guia AMI de seleção e adicionados às notas de versão.	30 de novembro de 2017
Versão da documentação inicial	Descrição detalhada da alteração com o link para o tópico ou a seção que sofreu a alteração.	15 de novembro de 2017

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.