aws

Panduan Developer

AWS Deep Learning AMIs

AWS Deep Learning AMIs: Panduan Developer

Table of Contents

Apa itu AWS Deep Learning AMIs?

AWS Deep Learning AMIs (DLAMI) menyediakan gambar mesin khusus yang dapat Anda gunakan untuk pembelajaran mendalam di cloud. Yang DLAMIs tersedia di sebagian besar Wilayah AWS untuk berbagai jenis instans Amazon Elastic Compute Cloud (AmazonEC2), dari instans kecil CPU saja hingga instans multi berdaya tinggi terbaru. GPU Yang DLAMIs datang telah dikonfigurasikan dengan NVIDIACUDADNNdan NVIDIAcu dan rilis terbaru dari kerangka pembelajaran mendalam yang paling populer.

Tentang panduan ini

Konten di dapat membantu Anda meluncurkan dan menggunakanDLAMIs. Panduan ini mencakup beberapa kasus penggunaan pembelajaran mendalam yang umum, baik untuk pelatihan maupun inferensi. Ini juga mencakup bagaimana memilih yang tepat AMI untuk tujuan Anda dan jenis contoh yang mungkin Anda sukai.

Selain itu, DLAMIs termasuk beberapa tutorial yang disediakan oleh kerangka kerja yang didukung mereka. Panduan ini dapat menunjukkan cara mengaktifkan setiap kerangka kerja dan menemukan tutorial yang sesuai untuk memulai. Ini juga memiliki tutorial tentang pelatihan terdistribusi, debugging, menggunakan AWS Inferensia dan AWS Trainium, dan konsep kunci lainnya. Untuk petunjuk tentang cara mengatur server notebook Jupyter untuk menjalankan tutorial di browser Anda, lihat. Menyiapkan server Jupyter Notebook pada sebuah instance DLAMI

Prasyarat

Agar berhasil menjalankanDLAMIs, kami sarankan Anda terbiasa dengan alat baris perintah dan Python dasar.

Contoh kasus DLAMI penggunaan

Berikut ini adalah contoh dari beberapa kasus penggunaan umum untuk AWS Deep Learning AMIs (DLAMI).

Belajar tentang pembelajaran mendalam - DLAMI adalah pilihan tepat untuk belajar atau mengajar pembelajaran mesin dan kerangka pembelajaran mendalam. DLAMIsMenghilangkan sakit kepala dari pemecahan masalah instalasi setiap kerangka kerja dan membuat mereka bermain bersama di komputer yang sama. DLAMIsTermasuk notebook Jupyter dan membuatnya mudah untuk

menjalankan tutorial yang disediakan kerangka kerja bagi orang-orang yang baru mengenal pembelajaran mesin dan pembelajaran mendalam.

Pengembangan aplikasi — Jika Anda seorang pengembang aplikasi yang tertarik menggunakan pembelajaran mendalam untuk membuat aplikasi Anda memanfaatkan kemajuan terbaru dalam AI, maka DLAMI adalah tempat uji yang sempurna untuk Anda. Setiap kerangka kerja dilengkapi dengan tutorial tentang cara memulai pembelajaran mendalam, dan banyak dari mereka memiliki kebun binatang model yang membuatnya mudah untuk mencoba pembelajaran mendalam tanpa harus membuat jaringan saraf sendiri atau melakukan pelatihan model apa pun. Beberapa contoh menunjukkan cara membuat aplikasi deteksi gambar hanya dalam beberapa menit, atau cara membuat aplikasi pengenalan suara untuk chatbot Anda sendiri.

Pembelajaran mesin dan analitik data — Jika Anda seorang ilmuwan data atau Anda tertarik untuk memproses data Anda dengan pembelajaran mendalam, maka Anda akan menemukan bahwa banyak kerangka kerja memiliki dukungan untuk R dan Spark. Anda akan menemukan tutorial tentang cara melakukan regresi sederhana, hingga membangun sistem pemrosesan data yang dapat diskalakan untuk sistem personalisasi dan prediksi.

Penelitian — Jika Anda seorang peneliti yang ingin mencoba kerangka kerja baru, menguji model baru, atau melatih model baru, lalu DLAMI dan AWS kemampuan untuk skala dapat mengurangi rasa sakit dari instalasi yang membosankan dan pengelolaan beberapa node pelatihan.

> **Note**
>
> Meskipun pilihan awal Anda mungkin memutakhirkan jenis instans Anda ke instance yang lebih besar dengan lebih banyak GPUs (hingga 8), Anda juga dapat menskalakan secara horizontal dengan membuat sekelompok DLAMI instance. Lihat Informasi terkait tentang DLAMI untuk informasi lebih lanjut tentang build cluster.

Fitur dari DLAMI

Fitur-fitur dari AWS Deep Learning AMIs (DLAMI) termasuk kerangka kerja pembelajaran mendalam yang telah diinstal sebelumnya, GPU perangkat lunak, server model, dan alat visualisasi model.

Kerangka kerja terinstal

Saat ini ada dua rasa utama DLAMI dengan variasi lain yang terkait dengan sistem operasi (OS) dan versi perangkat lunak:

- Pembelajaran Mendalam AMI dengan Conda— Kerangka kerja diinstal secara terpisah menggunakan `conda` paket dan lingkungan Python terpisah.
- Deep Learning Base AMI— Tidak ada kerangka kerja yang diinstal; hanya NVIDIACUDAdan dependensi lainnya.

Pembelajaran Mendalam AMI dengan Conda menggunakan `conda` lingkungan untuk mengisolasi setiap kerangka kerja, sehingga Anda dapat beralih di antara mereka sesuka hati dan tidak khawatir tentang dependensi mereka yang bertentangan. Pembelajaran Mendalam AMI dengan Conda mendukung kerangka kerja berikut:

- PyTorch
- TensorFlow 2

> **Note**
>
> DLAMItidak lagi mendukung kerangka pembelajaran mendalam berikut: Apache, MXNet Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Chainer, dan Keras.

Perangkat lunak yang sudah diinstal sebelumnya GPU

Bahkan jika Anda menggunakan instance CPU -only, DLAMIs will have NVIDIACUDAdan NVIDIAcuDNN. Perangkat lunak yang diinstal adalah sama terlepas dari jenis instancenya. Perlu diingat bahwa alat GPU -spesifik hanya berfungsi pada instance yang memiliki setidaknya satuGPU. Untuk informasi selengkapnya tentang jenis instance, lihatMemilih jenis DLAMI instance.

Untuk informasi lebih lanjut tentangCUDA, lihatInstalasi CUDA dan Framework Bindings.

Penyajian model dan visualisasi

Deep Learning AMI with Conda sudah diinstal sebelumnya dengan server model untuk TensorFlow, serta TensorBoard untuk visualisasi model. Untuk informasi selengkapnya, lihat TensorFlow Melayani.

Memulai dengan DLAMI

Panduan ini mencakup tips tentang memilih DLAMI yang tepat untuk Anda, memilih jenis instance yang sesuai dengan kasus penggunaan dan anggaran Anda, dan Informasi terkait tentang DLAMI yang menjelaskan pengaturan khusus yang mungkin menarik.

Jika Anda baru menggunakan AWS atau menggunakan AmazonEC2, mulailah denganPembelajaran Mendalam AMI dengan Conda. Jika Anda terbiasa dengan Amazon EC2 dan AWS layanan lain seperti AmazonEMR, AmazonEFS, atau Amazon S3, dan tertarik untuk mengintegrasikan layanan tersebut untuk proyek yang memerlukan pelatihan atau inferensi terdistribusi, periksa Informasi terkait tentang DLAMI untuk melihat apakah sesuai dengan kasus penggunaan Anda.

Kami menyarankan Anda memeriksa Memilih DLAMI untuk mendapatkan gambaran tentang jenis instans mana yang terbaik untuk aplikasi Anda.

Langkah selanjutnya

Memilih DLAMI

Memilih DLAMI

Kami menawarkan berbagai DLAMI pilihan. Untuk membantu Anda memilih yang benar DLAMI untuk kasus penggunaan Anda, kami mengelompokkan gambar berdasarkan jenis perangkat keras atau fungsionalitas yang dikembangkan. Pengelompokan tingkat atas kami adalah:

- DLAMIJenis: CUDA versus Basis versus Kerangka Tunggal versus Multi-Kerangka (Conda) DLAMI
- Arsitektur Komputasi: Graviton berbasis x86 versus ARM64 AWS
- Jenis Prosesor: GPUhttps://docs.aws.amazon.com/dlami/latest/devguide/gpu versus Inferentia CPUversus Trainium
- SDK: CUDAversus AWS Neuron
- OS: Amazon Linux versus Ubuntu

Topik lainnya dalam panduan ini membantu memberi tahu Anda lebih lanjut dan masuk ke detail lebih lanjut.

Topik

- Instalasi CUDA dan Framework Bindings

- Deep Learning Base AMI
- Pembelajaran Mendalam AMI dengan Conda
- Pilihan Arsitektur DLAMI
- Opsi Sistem Operasi DLAMI

Selanjutnya

Pembelajaran Mendalam AMI dengan Conda

Instalasi CUDA dan Framework Bindings

Sementara pembelajaran mendalam semuanya cukup canggih, setiap kerangka kerja menawarkan versi “stabil”. Versi stabil ini mungkin tidak berfungsi dengan implementasi dan fitur CUDA atau cuDNN terbaru. Kasus penggunaan Anda dan fitur yang Anda butuhkan dapat membantu Anda memilih kerangka kerja. Jika Anda tidak yakin, maka gunakan AMI Pembelajaran Mendalam terbaru dengan Conda. Ini memiliki `pip` binari resmi untuk semua kerangka kerja dengan CUDA, menggunakan versi terbaru mana pun yang didukung oleh setiap kerangka kerja. Jika Anda menginginkan versi terbaru, dan untuk menyesuaikan lingkungan pembelajaran mendalam Anda, gunakan AMI Dasar Pembelajaran Mendalam.

Lihat panduan kami Kandidat Stabil Versus Rilis untuk panduan lebih lanjut.

Pilih DLAMI dengan CUDA

Deep Learning Base AMIMemiliki semua seri versi CUDA yang tersedia

Pembelajaran Mendalam AMI dengan CondaMemiliki semua seri versi CUDA yang tersedia

> **Note**
>
> Kami tidak lagi menyertakan lingkungan MXNet, CNTK, Caffe, Caffe2, Theano, Chainer, atau Keras Conda di. AWS Deep Learning AMIs

Untuk nomor versi kerangka kerja tertentu, lihat Catatan rilis untuk DLAMIs

Pilih jenis DLAMI ini atau pelajari lebih lanjut tentang DLAMI yang berbeda dengan opsi Next Up.

Pilih salah satu versi CUDA dan tinjau daftar lengkap DLAMI yang memiliki versi tersebut di Lampiran, atau pelajari lebih lanjut tentang DLAMI yang berbeda dengan opsi Next Up.

Selanjutnya

Deep Learning Base AMI

Topik Terkait

- Untuk petunjuk tentang beralih antara versi CUDA, lihat Menggunakan Basis Pembelajaran Mendalam AMI tutorial.

Deep Learning Base AMI

Deep Learning Base AMI seperti kanvas kosong untuk pembelajaran mendalam. Muncul dengan semua yang Anda butuhkan sampai titik instalasi kerangka tertentu, dan memiliki pilihan Anda versi CUDA.

Mengapa Memilih Dasar DLAMI

Grup AMI ini berguna untuk kontributor proyek yang ingin membuat proyek pembelajaran mendalam dan membangun yang terbaru. Ini untuk seseorang yang ingin menggulung lingkungannya sendiri dengan keyakinan bahwa perangkat lunak NVIDIA terbaru diinstal dan berfungsi sehingga mereka dapat fokus memilih kerangka kerja dan versi mana yang ingin mereka instal.

Pilih jenis DLAMI ini atau pelajari lebih lanjut tentang DLAM yang berbeda dengan opsi Next Up.

Selanjutnya Up

DLAMI dengan Conda

Topik Terkait

- Menggunakan AMI Deep Learning Base

Pembelajaran Mendalam AMI dengan Conda

Conda DLAMI menggunakan lingkungan `conda` virtual, mereka hadir baik multi-framework atau kerangka kerja tunggal. DLAMIs Lingkungan ini dikonfigurasi untuk menjaga instalasi kerangka kerja yang berbeda terpisah dan merampingkan peralihan antar kerangka kerja. Ini bagus untuk belajar dan bereksperimen dengan semua kerangka kerja yang DLAMI ditawarkan. Sebagian besar pengguna menemukan bahwa Deep Learning baru AMI dengan Conda sangat cocok untuk mereka.

Mereka sering diperbarui dengan versi terbaru dari kerangka kerja, dan memiliki GPU driver dan perangkat lunak terbaru. Mereka umumnya disebut sebagai AWS Deep Learning AMIs dalam sebagian besar dokumen. Ini DLAMIs mendukung sistem Operasi Ubuntu 20.04, Amazon Linux 2. Dukungan sistem operasi tergantung pada dukungan dari OS upstream.

Kandidat Stabil Versus Rilis

Conda AMIs menggunakan binari yang dioptimalkan dari rilis formal terbaru dari setiap kerangka kerja. Kandidat rilis dan fitur eksperimental tidak diharapkan. Pengoptimalan tergantung pada dukungan kerangka kerja untuk teknologi akselerasi seperti Intel MKLDNN, yang mempercepat pelatihan dan inferensi pada jenis instans C5 dan C4. CPU Binari juga dikompilasi untuk mendukung set instruksi Intel tingkat lanjut termasuk tetapi tidak terbatas padaAVX, AVX -2, SSE4 .1, dan SSE4 .2. Ini mempercepat operasi vektor dan floating point pada CPU arsitektur Intel. Selain itu, GPU misalnya jenis, CUDA dan cu DNN diperbarui dengan versi mana pun yang didukung rilis resmi terbaru.

Pembelajaran Mendalam AMI dengan Conda secara otomatis menginstal versi kerangka kerja yang paling dioptimalkan untuk EC2 instans Amazon Anda pada aktivasi pertama kerangka kerja. Untuk informasi lebih lanjut, lihatMenggunakan Deep Learning AMI dengan Conda.

Jika Anda ingin menginstal dari sumber, menggunakan opsi build khusus atau yang dioptimalkan, Deep Learning Base AMI s mungkin merupakan opsi yang lebih baik untuk Anda.

Pengakhiran Python 2

Komunitas open source Python telah secara resmi mengakhiri dukungan untuk Python 2 pada 1 Januari 2020. PyTorch Komunitas TensorFlow dan telah mengumumkan bahwa rilis TensorFlow 2.1 dan PyTorch 1.4 adalah yang terakhir mendukung Python 2. Rilis sebelumnya dari DLAMI (v26, v25, dll) yang berisi lingkungan Python 2 Conda terus tersedia. Namun, kami menyediakan pembaruan untuk lingkungan Python 2 Conda pada DLAMI versi yang diterbitkan sebelumnya hanya jika ada perbaikan keamanan yang diterbitkan oleh komunitas sumber terbuka untuk versi tersebut. DLAMIrilis dengan versi terbaru dari TensorFlow dan PyTorch kerangka kerja tidak mengandung lingkungan Python 2 Conda.

CUDASupport

Nomor CUDA versi tertentu dapat ditemukan di catatan GPU DLAMI rilis.

Selanjutnya

Pilihan Arsitektur DLAMI

Topik Terkait

- Untuk tutorial tentang menggunakan Deep Learning AMI dengan Conda, lihat Menggunakan Deep Learning AMI dengan Conda tutorialnya.

Pilihan Arsitektur DLAMI

AWS Deep Learning AMIss ditawarkan dengan arsitektur Graviton2 berbasis x86 atau berbasis AWS ARM64.

Untuk informasi tentang memulai dengan DLAMI GPU ARM64, lihat. The ARM64 DLAMI Untuk detail selengkapnya tentang jenis instans yang tersedia, lihatMemilih jenis DLAMI instance.

Selanjutnya

Opsi Sistem Operasi DLAMI

Opsi Sistem Operasi DLAMI

DLAMI ditawarkan dalam sistem operasi berikut.

- Amazon Linux 2
- Ubuntu 20.04
- Ubuntu 22.04

Versi sistem operasi yang lebih lama tersedia pada DLAMI yang tidak digunakan lagi. Untuk informasi selengkapnya tentang penghentian DLAMI, lihat Deprecations for DLAMI

Sebelum memilih DLAMI, nilai jenis instans apa yang Anda butuhkan dan identifikasi Wilayah Anda. AWS

Selanjutnya

Memilih jenis DLAMI instance

Memilih jenis DLAMI instance

Secara lebih umum, pertimbangkan hal berikut ketika memilih jenis instance untuk aDLAMI.

- Jika Anda baru mengenal pembelajaran mendalam, maka contoh dengan satu GPU mungkin sesuai dengan kebutuhan Anda.
- Jika Anda sadar anggaran, maka Anda dapat menggunakan instance CPU -only.
- Jika Anda ingin mengoptimalkan kinerja tinggi dan efisiensi biaya untuk inferensi model pembelajaran mendalam, maka Anda dapat menggunakan instance dengan chip AWS Inferentia.
- Jika Anda mencari GPU instans berkinerja tinggi dengan CPU arsitektur berbasis ARM64, maka Anda dapat menggunakan tipe instans G5G.
- Jika Anda tertarik untuk menjalankan model terlatih untuk inferensi dan prediksi, Anda dapat melampirkan Amazon Elastic Inference ke instans Amazon Anda. EC2 Amazon Elastic Inference memberi Anda akses ke akselerator dengan fraksi. GPU
- Untuk layanan inferensi volume tinggi, satu CPU instance dengan banyak memori, atau sekelompok instance semacam itu, mungkin merupakan solusi yang lebih baik.
- Jika Anda menggunakan model besar dengan banyak data atau ukuran batch tinggi, maka Anda memerlukan instance yang lebih besar dengan lebih banyak memori. Anda juga dapat mendistribusikan model Anda ke sekelompokGPUs. Anda mungkin menemukan bahwa menggunakan instance dengan memori lebih sedikit adalah solusi yang lebih baik untuk Anda jika Anda mengurangi ukuran batch Anda. Ini dapat memengaruhi akurasi dan kecepatan pelatihan Anda.
- Jika Anda tertarik untuk menjalankan aplikasi pembelajaran mesin menggunakan NVIDIA Collective Communications Library (NCCL) yang membutuhkan komunikasi antar-simpul tingkat tinggi dalam skala besar, Anda mungkin ingin menggunakan Elastic Fabric Adapter () EFA.

Untuk detail selengkapnya tentang instance, lihat .

Topik berikut memberikan informasi tentang pertimbangan jenis instance.

> **Important**
>
> Deep Learning AMIs mencakup driver, perangkat lunak, atau toolkit yang dikembangkan, dimiliki, atau disediakan oleh NVIDIA Corporation. Anda setuju untuk menggunakan NVIDIA driver, perangkat lunak, atau toolkit ini hanya pada EC2 instans Amazon yang menyertakan perangkat keras. NVIDIA

Topik

Harga untuk DLAMI

Kerangka kerja pembelajaran mendalam yang DLAMI disertakan di dalamnya gratis, dan masing-masing memiliki lisensi sumber terbuka sendiri. Meskipun perangkat lunak yang disertakan di DLAMI dalamnya gratis, Anda masih harus membayar untuk perangkat keras EC2 instans Amazon yang mendasarinya.

Beberapa jenis EC2 instans Amazon diberi label gratis. Dimungkinkan untuk menjalankan salah DLAMI satu contoh gratis ini. Ini berarti bahwa menggunakan DLAMI sepenuhnya gratis ketika Anda hanya menggunakan kapasitas instance itu. Jika Anda membutuhkan instance yang lebih kuat dengan lebih banyak CPU core, lebih banyak ruang diskRAM, lebih banyak, atau satu atau lebihGPUs, maka Anda memerlukan instance yang tidak ada di kelas instance free-tier.

Untuk informasi selengkapnya tentang pilihan dan harga instans, lihat EC2harga Amazon.

Ketersediaan Wilayah DLAMI

Setiap Wilayah mendukung berbagai jenis instans yang berbeda dan seringkali jenis instans memiliki biaya yang sedikit berbeda di Wilayah yang berbeda. DLAMIstidak tersedia di setiap Wilayah, tetapi dimungkinkan untuk menyalin DLAMIs ke Wilayah pilihan Anda. Lihat Menyalin AMI untuk informasi lebih lanjut. Perhatikan daftar pilihan Wilayah dan pastikan Anda memilih Wilayah yang dekat dengan Anda atau pelanggan Anda. Jika Anda berencana untuk menggunakan lebih dari satu DLAMI dan berpotensi membuat cluster, pastikan untuk menggunakan Region yang sama untuk semua node di cluster.

Untuk info lebih lanjut tentang Wilayah, kunjungi titik akhir EC2 layanan Amazon.

Selanjutnya

GPUContoh yang Direkomendasikan

GPUContoh yang Direkomendasikan

Kami merekomendasikan GPU contoh untuk sebagian besar tujuan pembelajaran mendalam. Melatih model baru lebih cepat pada GPU instance daripada CPU instance. Anda dapat menskalakan sub-linier ketika Anda memiliki GPU multi-instance atau jika Anda menggunakan pelatihan terdistribusi di banyak instance dengan. GPUs

Jenis contoh berikut mendukung fileDLAMI. Untuk informasi tentang opsi tipe GPU instans dan kegunaannya, lihat dan pilih Accelerated Computing.

> **Note**
>
> Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi instans yang tersediaRAM, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- Instans Amazon EC2 P5e memiliki hingga 8 NVIDIA Tesla H200. GPUs
- Instans Amazon EC2 P5 memiliki hingga 8 NVIDIA Tesla H100. GPUs
- Instans Amazon EC2 P4 memiliki hingga 8 NVIDIA Tesla A100. GPUs
- Instans Amazon EC2 P3 memiliki hingga 8 NVIDIA Tesla V100. GPUs
- Instans Amazon EC2 G3 memiliki hingga 4 NVIDIA Tesla M60. GPUs
- Instans Amazon EC2 G4 memiliki hingga 4 T4NVIDIA. GPUs
- Instans Amazon EC2 G5 memiliki hingga 8 NVIDIA A10G. GPUs
- Instans Amazon EC2 G6 memiliki hingga 8 NVIDIA L4. GPUs
- Instans Amazon EC2 G6e memiliki hingga 8 Inti Tensor NVIDIA L40S. GPUs
- Instans Amazon EC2 G5G memiliki prosesor Graviton2 berbasis ARM64 AWS .

DLAMIinstans menyediakan perkakas untuk memantau dan mengoptimalkan proses AndaGPU. Untuk informasi selengkapnya tentang memantau GPU proses Anda, lihatGPUPemantauan dan Optimalisasi.

Untuk tutorial khusus tentang bekerja dengan instans G5G, lihat. The ARM64 DLAMI

Selanjutnya

CPUContoh yang Direkomendasikan

CPUContoh yang Direkomendasikan

Baik Anda memiliki anggaran terbatas, belajar tentang pembelajaran mendalam, atau hanya ingin menjalankan layanan prediksi, Anda memiliki banyak opsi terjangkau dalam CPU kategori tersebut. Beberapa kerangka kerja memanfaatkan Intel MKLDNN, yang mempercepat pelatihan dan inferensi pada jenis instans C5 (tidak tersedia di semua Wilayah)CPU. Untuk informasi tentang tipe CPU instans, lihat dan pilih Compute Optimized.

> **Note**
>
> Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi instans yang tersediaRAM, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- Instans Amazon EC2 C5 memiliki hingga 72 Intel. vCPUs Instans C5 unggul dalam pemodelan ilmiah, pemrosesan batch, analitik terdistribusi, komputasi kinerja tinggi (HPC), dan inferensi pembelajaran mesin dan mendalam.

Selanjutnya

Contoh Inferensia yang Direkomendasikan

Contoh Inferensia yang Direkomendasikan

AWS Instance inferensia dirancang untuk memberikan kinerja tinggi dan efisiensi biaya untuk beban kerja inferensi model pembelajaran mendalam. Secara khusus, jenis instans Inf2 menggunakan chip AWS Inferentia dan AWS Neuron SDK, yang terintegrasi dengan kerangka kerja pembelajaran mesin populer seperti dan. TensorFlow PyTorch

Pelanggan dapat menggunakan instans Inf2 untuk menjalankan aplikasi inferensi pembelajaran mesin skala besar seperti pencarian, mesin rekomendasi, visi komputer, pengenalan suara, pemrosesan bahasa alami, personalisasi, dan deteksi penipuan, dengan biaya terendah di cloud.

> **Note**
>
> Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi instans yang tersediaRAM, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- Instans Amazon EC2 Inf2 memiliki hingga 16 chip AWS Inferentia dan throughput jaringan 100 Gbps.

Untuk informasi lebih lanjut tentang memulai dengan AWS InferensiaDLAMIs, lihatChip AWS Inferensia Dengan DLAMI.

Selanjutnya

Instans Trainium yang Direkomendasikan

Instans Trainium yang Direkomendasikan

AWS Instans Trainium dirancang untuk memberikan kinerja tinggi dan efisiensi biaya untuk beban kerja inferensi model pembelajaran mendalam. Secara khusus, jenis instans Trn1 menggunakan chip AWS Trainium dan AWS Neuron SDK, yang terintegrasi dengan kerangka kerja pembelajaran mesin populer seperti dan. TensorFlow PyTorch

Pelanggan dapat menggunakan instans Trn1 untuk menjalankan aplikasi inferensi pembelajaran mesin skala besar seperti pencarian, mesin rekomendasi, visi komputer, pengenalan suara, pemrosesan bahasa alami, personalisasi, dan deteksi penipuan, dengan biaya terendah di cloud.

> **Note**
>
> Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi instans yang tersediaRAM, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- Instans Amazon EC2 Trn1 memiliki hingga 16 chip AWS Trainium dan throughput jaringan 100 Gbps.

Menyiapkan sebuah DLAMI instance

Setelah Anda memilih DLAMI dan memilih jenis instans Amazon Elastic Compute Cloud (AmazonEC2) yang ingin Anda gunakan, maka Anda siap untuk menyiapkan DLAMI instans baru Anda.

Jika Anda belum memilih tipe DLAMI dan EC2 instance, lihatMemulai dengan DLAMI.

Topik

- Menemukan ID dari DLAMI
- Meluncurkan sebuah DLAMI instance
- Menghubungkan ke sebuah DLAMI instance
- Menyiapkan server Jupyter Notebook pada sebuah instance DLAMI
- Membersihkan sebuah DLAMI instance

Menemukan ID dari DLAMI

Masing-masing DLAMI memiliki pengenal unik (ID). Saat meluncurkan DLAMI instance menggunakan EC2 konsol Amazon, Anda dapat menggunakan DLAMI ID secara opsional untuk mencari DLAMI yang ingin Anda gunakan. Saat Anda meluncurkan DLAMI instance menggunakan AWS Command Line Interface (AWS CLI), ID ini diperlukan.

Anda dapat menemukan ID untuk DLAMI pilihan Anda dengan menggunakan AWS CLI perintah untuk Amazon EC2 atau Parameter Store, kemampuan AWS Systems Manager. Untuk petunjuk tentang menginstal dan mengonfigurasi AWS CLI, lihat Memulai dengan AWS CLI di Panduan AWS Command Line Interface Pengguna.

Using Parameter Store

Untuk menemukan DLAMI ID menggunakan ssm get-parameter

Dalam ssm get-parameterperintah berikut, untuk `--name` opsi, format nama parameter adalah */aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id*. Dalam format nama ini, *`architecture`* bisa salah satu **`x86_64`** atau**`arm64`**. Tentukan *`ami_type`* dengan mengambil DLAMI nama dan menghapus kata kunci “mendalam”, “belajar”, dan “ami”. AMINama dapat ditemukan di Catatan rilis untuk DLAMIs

> **Important**
>
> Untuk menggunakan perintah ini, prinsipal AWS Identity and Access Management (IAM) yang Anda gunakan harus memiliki `ssm:GetParameter` izin. Untuk informasi selengkapnya tentang IAM kepala sekolah, lihat bagian Sumber daya tambahan IAM peran di Panduan Pengguna. IAM

- ```
  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
  ```

  Outputnya akan serupa dengan yang berikut ini:

  ```
  ami-09ee1a996ac214ce7
  ```

  > **Tip**
  >
  > Untuk beberapa DLAMI kerangka kerja yang saat ini didukung, Anda dapat menemukan ssm get-parameter perintah contoh yang lebih spesifik diCatatan rilis untuk DLAMIs. Pilih tautan ke catatan rilis yang Anda pilihDLAMI, lalu temukan kueri ID-nya di catatan rilis.

Using Amazon EC2 CLI

Untuk menemukan DLAMI ID menggunakan ec2 describe-images

Dalam ec2 describe-imagesperintah berikut, untuk nilai filterName=name, masukkan DLAMI nama. Anda dapat menentukan versi rilis untuk kerangka kerja tertentu, atau Anda bisa mendapatkan rilis terbaru dengan mengganti nomor versi dengan tanda tanya (?).

- ```
  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
  ```

  Outputnya akan serupa dengan yang berikut ini:

```
ami-09ee1a996ac214ce7
```

> **Tip**
>
> Untuk contoh ec2 describe-images perintah yang khusus untuk DLAMI pilihan Anda, lihatCatatan rilis untuk DLAMIs. Pilih tautan ke catatan rilis yang Anda pilihDLAMI, lalu temukan kueri ID-nya di catatan rilis.

Langkah selanjutnya

Meluncurkan sebuah DLAMI instance

Meluncurkan sebuah DLAMI instance

Setelah Anda menemukan ID DLAMI yang ingin Anda gunakan untuk meluncurkan DLAMI instance, Anda siap untuk meluncurkan instance. Untuk meluncurkannya, Anda dapat menggunakan EC2 konsol Amazon atau AWS Command Line Interface (AWS CLI).

> **Note**
>
> Untuk panduan ini, kami mungkin membuat referensi khusus untuk Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04). Bahkan jika Anda memilih yang berbedaDLAMI, Anda harus dapat mengikuti panduan ini.

EC2 console

> **Note**
>
> Untuk mempercepat aplikasi komputasi (HPC) dan pembelajaran mesin berkinerja tinggi, Anda dapat meluncurkan DLAMI instans Anda dengan Elastic Fabric Adapter (EFA). Untuk instruksi khusus, lihatMeluncurkan AWS Deep Learning AMIs Instance Dengan EFA.

1. Buka EC2Konsol.

2. Perhatikan saat ini Wilayah AWS di navigasi paling atas. Jika ini bukan Wilayah yang Anda inginkan, maka ubah opsi ini sebelum Anda melanjutkan. Untuk informasi selengkapnya, lihat Titik akhir EC2 layanan Amazon EC2 di. Referensi Umum Amazon Web Services
3. Pilih Luncurkan Instans.
4. Masukkan nama untuk instans Anda dan pilih DLAMI yang tepat untuk Anda.

   a. Temukan yang sudah ada DLAMI di My AMIs atau pilih Quick Start.

   b. Cari berdasarkan DLAMI ID. Jelajahi opsi lalu pilih pilihan Anda.
5. Pilih jenis instance. Anda dapat menemukan keluarga contoh yang direkomendasikan untuk Anda DLAMICatatan rilis untuk DLAMIs. Untuk rekomendasi umum tentang jenis DLAMI instans, lihatMemilih jenis DLAMI instance.
6. Pilih Luncurkan Instans.

AWS CLI

- Untuk menggunakan AWS CLI, Anda harus memiliki ID DLAMI yang ingin Anda gunakan, jenis Wilayah AWS dan EC2 instance, dan informasi token keamanan Anda. Kemudian, Anda dapat meluncurkan instance menggunakan ec2 run-instances AWS CLI perintah.

  Untuk petunjuk tentang menginstal dan mengonfigurasi AWS CLI, lihat Memulai dengan AWS CLI di Panduan AWS Command Line Interface Pengguna. Untuk informasi selengkapnya, termasuk contoh perintah, lihat Meluncurkan, mencantumkan, dan menutup EC2 instans Amazon untuk. AWS CLI

Setelah meluncurkan instans menggunakan EC2 konsol Amazon atau AWS CLI, tunggu hingga instans siap. Ini biasanya hanya memakan waktu beberapa menit. Anda dapat memverifikasi status instans di EC2konsol Amazon. Untuk informasi selengkapnya, lihat Pemeriksaan status untuk EC2 instans Amazon di Panduan EC2 Pengguna Amazon.

Langkah selanjutnya

Menghubungkan ke sebuah DLAMI instance

Menghubungkan ke sebuah DLAMI instance

Setelah meluncurkan DLAMI instance dan instance sedang berjalan, Anda dapat menghubungkannya dari klien (Windows, macOS, atau Linux) menggunakan. SSH Untuk petunjuknya, lihat Connect ke instans Linux Anda menggunakan SSH Panduan EC2 Pengguna Amazon.

Simpan salinan perintah SSH login jika Anda ingin menyiapkan server Jupyter Notebook setelah Anda masuk. Untuk terhubung ke halaman web Jupyter, Anda menggunakan variasi dari perintah itu.

Langkah selanjutnya

Menyiapkan server Jupyter Notebook pada sebuah instance DLAMI

Menyiapkan server Jupyter Notebook pada sebuah instance DLAMI

Dengan server Jupyter Notebook, Anda dapat membuat dan menjalankan notebook Jupyter dari instance Anda. DLAMI Dengan notebook Jupyter, Anda dapat melakukan eksperimen pembelajaran mesin (ML) untuk pelatihan dan inferensi saat menggunakan AWS infrastruktur dan mengakses paket yang ada di dalamnya. DLAMI Untuk informasi selengkapnya tentang notebook Jupyter, lihat Notebook Jupyter di situs web Dokumentasi Pengguna Jupyter.

Untuk menyiapkan server Jupyter Notebook, Anda harus:

- Konfigurasikan server Jupyter Notebook pada instans AndaDLAMI.
- Konfigurasikan klien Anda untuk terhubung ke server Jupyter Notebook. Kami menyediakan instruksi konfigurasi untuk klien Windows, macOS, dan Linux.
- Uji penyiapan dengan masuk ke server Jupyter Notebook.

Untuk menyelesaikan langkah-langkah ini, ikuti instruksi dalam topik berikut. Setelah menyiapkan server Jupyter Notebook, Anda dapat menjalankan contoh tutorial notebook yang dikirimkan di file. DLAMIs Untuk informasi selengkapnya, lihat Menjalankan Tutorial Notebook Jupyter.

Topik

- Mengamankan server Jupyter Notebook pada sebuah instance DLAMI
- Memulai server Jupyter Notebook pada sebuah instance DLAMI

- Menghubungkan klien ke server Jupyter Notebook pada sebuah instance DLAMI
- Masuk ke server Jupyter Notebook pada sebuah instance DLAMI

Mengamankan server Jupyter Notebook pada sebuah instance DLAMI

Agar server Jupyter Notebook Anda tetap aman, kami sarankan untuk menyiapkan kata sandi dan membuat SSL sertifikat untuk server. Untuk mengonfigurasi kata sandi danSSL, pertama-tama sambungkan ke DLAMI instans Anda, lalu ikuti petunjuk ini.

Untuk mengamankan server Notebook Jupyter

1. Jupyter menyediakan utilitas kata sandi. Jalankan perintah berikut dan masukkan kata sandi pilihan Anda di prompt.

   ```
   $ jupyter notebook password
   ```

   Outputnya akan terlihat seperti ini:

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

2. Buat SSL sertifikat yang ditandatangani sendiri. Ikuti petunjuk untuk mengisi wilayah Anda sesuai keinginan Anda. Anda harus masuk `.` jika Anda ingin membiarkan prompt kosong. Jawaban Anda tidak akan memengaruhi fungsionalitas sertifikat.

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

> **Note**
>
> Anda mungkin tertarik untuk membuat SSL sertifikat reguler yang ditandatangani oleh pihak ketiga dan tidak menyebabkan browser memberi Anda peringatan keamanan. Proses ini

jauh lebih terlibat. Untuk informasi selengkapnya, lihat Mengamankan server notebook di dokumentasi pengguna Jupyter Notebook.

Langkah selanjutnya

Memulai server Jupyter Notebook pada sebuah instance DLAMI

Memulai server Jupyter Notebook pada sebuah instance DLAMI

Setelah Anda mengamankan server Jupyter Notebook Anda dengan kata sandi dan SSL, Anda dapat memulai server. Masuk ke DLAMI instans Anda dan jalankan perintah berikut yang menggunakan SSL sertifikat yang Anda buat sebelumnya.

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

Dengan server dimulai, Anda sekarang dapat terhubung ke sana melalui SSH terowongan dari komputer klien Anda. Ketika server berjalan, Anda akan melihat beberapa output dari Jupyter yang mengonfirmasi bahwa server sedang berjalan. Pada titik ini, abaikan callout bahwa Anda dapat mengakses server melalui host lokalURL, karena itu tidak akan berfungsi sampai Anda membuat terowongan.

> **Note**
>
> Jupyter akan menangani lingkungan switching untuk Anda saat Anda mengganti kerangka kerja menggunakan antarmuka web Jupyter. Untuk informasi selengkapnya, lihat Beralih Lingkungan dengan Jupyter.

Langkah selanjutnya

Menghubungkan klien ke server Jupyter Notebook pada sebuah instance DLAMI

Menghubungkan klien ke server Jupyter Notebook pada sebuah instance DLAMI

Setelah Anda memulai server Jupyter Notebook pada DLAMI instans Anda, konfigurasikan klien Windows, macOS, atau Linux Anda untuk terhubung ke server. Saat Anda terhubung, Anda dapat

membuat dan mengakses notebook Jupyter di server di ruang kerja Anda dan menjalankan kode pembelajaran mendalam Anda di server.

Prasyarat

Pastikan Anda memiliki yang berikut ini, yang Anda butuhkan untuk mengatur SSH terowongan:

- DNSNama publik EC2 instans Amazon Anda. Untuk informasi selengkapnya, lihat jenis nama host EC2 instans Amazon di Panduan EC2 Pengguna Amazon.
- Key pair untuk file kunci pribadi. Untuk informasi selengkapnya tentang mengakses key pair, lihat pasangan EC2 kunci Amazon dan EC2 instans Amazon di EC2Panduan Pengguna Amazon.

Connect dari klien Windows, macOS, atau Linux

Untuk terhubung ke DLAMI instans Anda dari klien Windows, macOS, atau Linux, ikuti instruksi untuk sistem operasi klien Anda.

Windows

Untuk terhubung ke DLAMI instans Anda dari klien Windows menggunakan SSH

1. Gunakan SSH klien untuk Windows, seperti PuTTY. Untuk petunjuknya, lihat Connect ke instans Linux menggunakan Pu TTY di Panduan EC2 Pengguna Amazon. Untuk opsi SSH koneksi lainnya, lihat Connect to your Linux instance using SSH.
2. (Opsional) Buat SSH terowongan ke server Jupyter yang sedang berjalan. Instal Git Bash di klien Windows Anda, lalu ikuti instruksi koneksi untuk klien macOS dan Linux.

macOS or Linux

Untuk terhubung ke DLAMI instans Anda dari klien macOS atau Linux menggunakan SSH

1. Buka terminal.
2. Jalankan perintah berikut untuk meneruskan semua permintaan pada port lokal 8888 ke port 8888 pada instans Amazon EC2 jarak jauh Anda. Perbarui perintah dengan mengganti lokasi kunci Anda untuk mengakses EC2 instans Amazon dan DNS nama publik EC2 instans Amazon Anda. Catatan, untuk Amazon LinuxAMI, nama pengguna `ec2-user` bukan`ubuntu`.

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

Perintah ini membuka terowongan antara klien Anda dan EC2 instans Amazon jarak jauh yang menjalankan server Jupyter Notebook.

Langkah selanjutnya

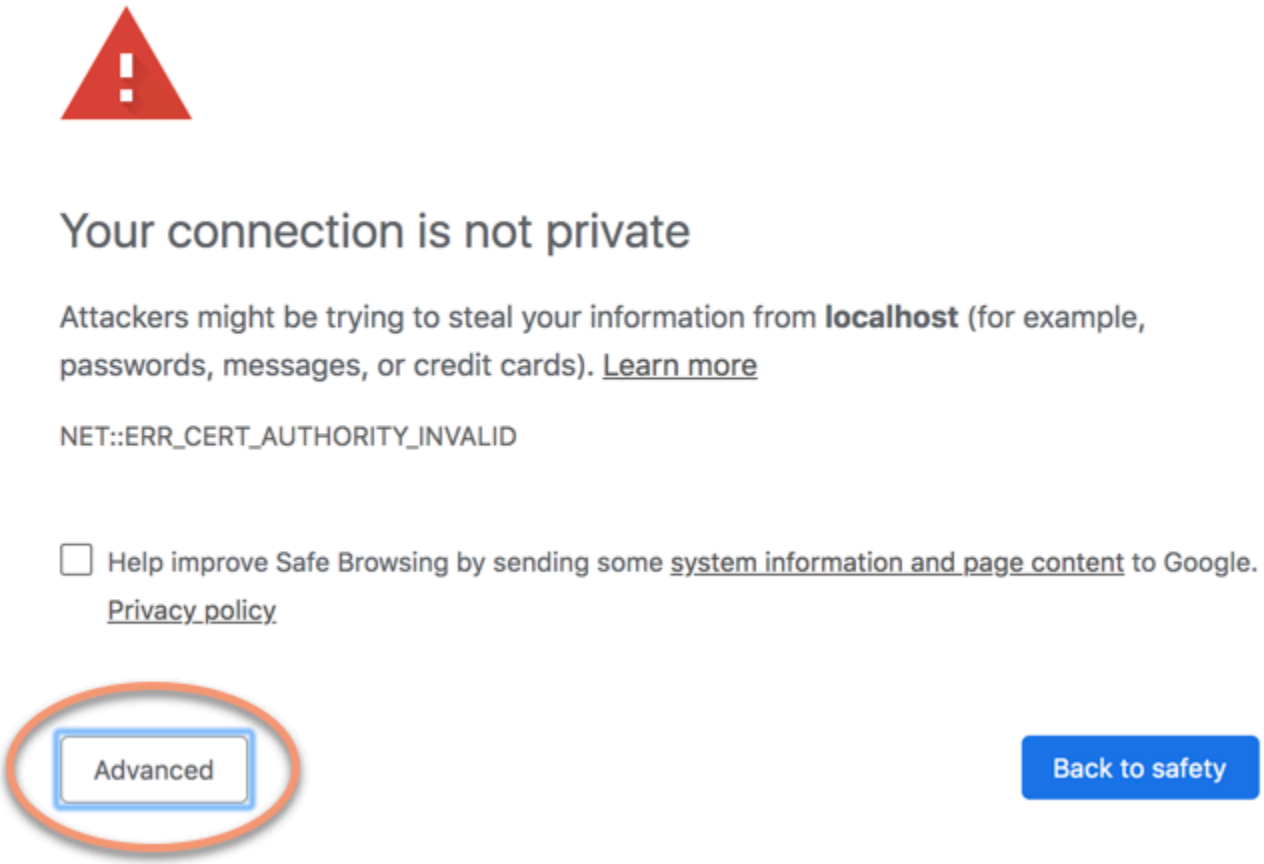
Masuk ke server Jupyter Notebook pada sebuah instance DLAMI

Masuk ke server Jupyter Notebook pada sebuah instance DLAMI

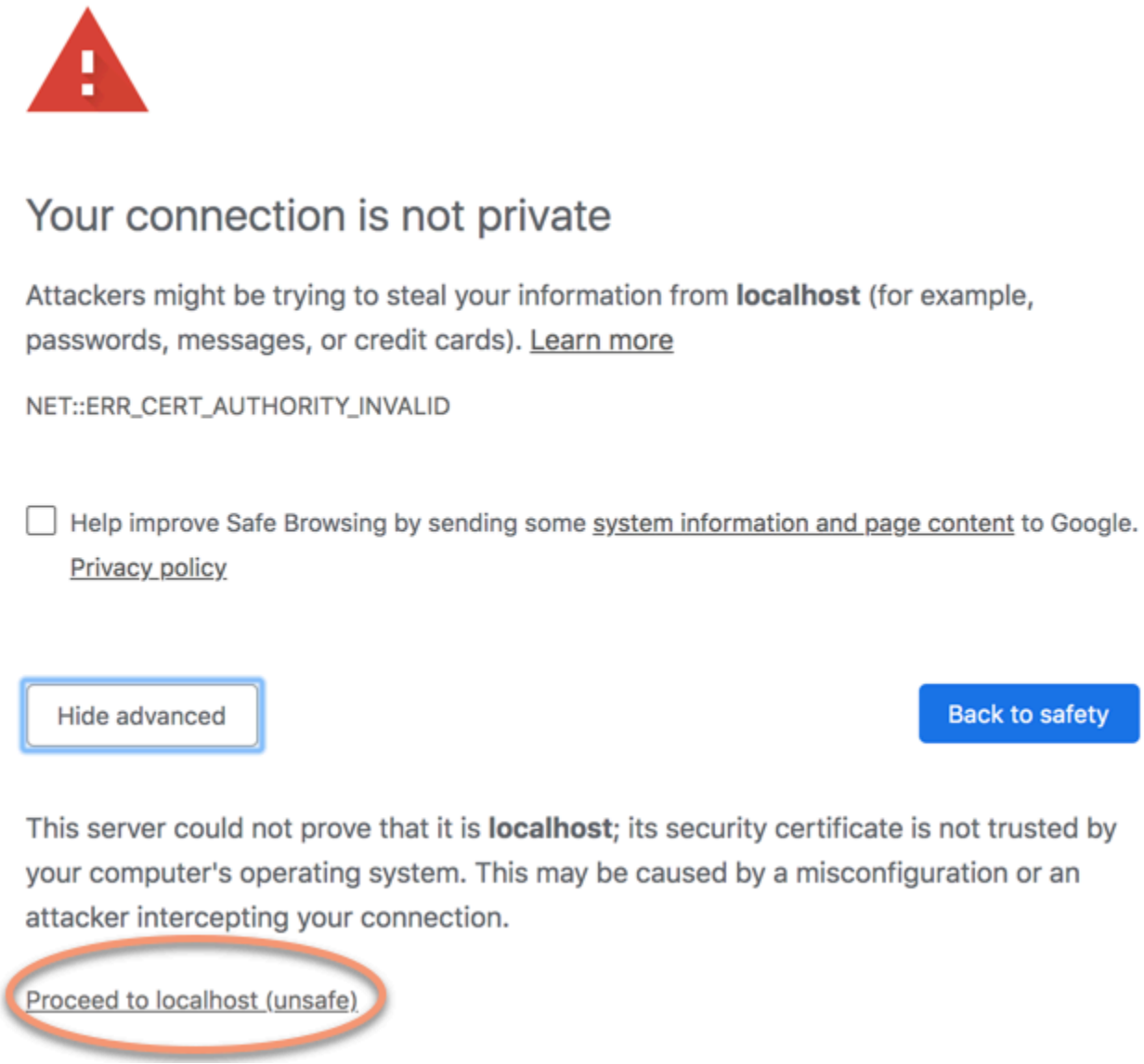
Setelah Anda menghubungkan klien Anda ke server Jupyter Notebook pada DLAMI instans Anda, Anda dapat masuk ke server.

Untuk masuk ke server di browser Anda

1. Di bilah alamat browser Anda, masukkan yang berikut iniURL, atau klik tautan ini: https://localhost:8888
2. Dengan SSL sertifikat yang ditandatangani sendiri, browser Anda akan memperingatkan Anda dan meminta Anda untuk menghindari terus mengunjungi situs web.



Karena Anda mengatur ini sendiri, aman untuk melanjutkan. Bergantung pada browser Anda, Anda akan mendapatkan tombol “lanjutan”, “tampilkan detail”, atau serupa.



Klik ini, lalu klik tautan “lanjutkan ke localhost”. Jika koneksi berhasil, Anda melihat halaman web server Jupyter Notebook. Pada titik ini, Anda akan diminta kata sandi yang sebelumnya Anda atur.

Sekarang Anda memiliki akses ke server Jupyter Notebook yang berjalan pada instance. DLAMI Anda dapat membuat notebook baru atau menjalankan yang disediakanTutorial.

Membersihkan sebuah DLAMI instance

Ketika Anda tidak lagi membutuhkan DLAMI instans Anda, Anda dapat menghentikannya atau menghentikannya di Amazon EC2 untuk menghindari biaya yang tidak terduga.

Jika Anda menghentikan sebuah instance, Anda dapat menyimpannya dan memulainya nanti ketika Anda ingin menggunakannya lagi. Konfigurasi, file, dan informasi non-volatile lainnya disimpan dalam volume di Amazon Simple Storage Service (Amazon S3). Saat instans dihentikan, Anda dikenakan biaya S3 untuk mempertahankan volume, tetapi Anda tidak dikenakan biaya untuk sumber daya komputasi. Ketika Anda memulai instance lagi, itu akan memasang volume penyimpanan itu dengan data Anda.

Jika Anda menghentikan sebuah instance, itu hilang, dan Anda tidak dapat memulainya lagi. Tentu saja, Anda tidak akan dikenakan biaya lagi untuk sumber daya komputasi dengan instance yang dihentikan. Namun, data Anda masih berada di Amazon S3, dan Anda dapat terus dikenakan biaya S3. Untuk mencegah semua biaya lebih lanjut yang terkait dengan instans yang dihentikan, Anda juga harus menghapus volume penyimpanan di Amazon S3. Untuk petunjuk, lihat Menghentikan EC2 instans Amazon di EC2Panduan Pengguna Amazon.

Untuk informasi selengkapnya tentang status EC2 instans Amazon, seperti `stopped` dan`terminated`, lihat perubahan status EC2 instans Amazon di Panduan EC2 Pengguna Amazon.

Menggunakan DLAMI

Topik

Bagian berikut menjelaskan bagaimana Deep Learning AMI with Conda dapat digunakan untuk beralih lingkungan, menjalankan kode sampel dari masing-masing kerangka kerja, dan menjalankan Jupyter sehingga Anda dapat mencoba tutorial notebook yang berbeda.

Menggunakan Deep Learning AMI dengan Conda

Topik

Pengantar Pembelajaran Mendalam AMI dengan Conda

Conda adalah sistem manajemen paket open source dan sistem manajemen lingkungan yang berjalan di Windows, macOS, dan Linux. Conda dengan cepat menginstal, menjalankan, dan memperbarui paket dan dependensinya. Conda dengan mudah membuat, menyimpan, memuat, dan beralih antar lingkungan di komputer lokal Anda.

Pembelajaran Mendalam AMI dengan Conda telah dikonfigurasi agar Anda dapat dengan mudah beralih di antara lingkungan pembelajaran yang mendalam. Instruksi berikut memandu Anda pada beberapa perintah dasar dengan`conda`. Mereka juga membantu Anda memverifikasi bahwa impor dasar kerangka kerja berfungsi, dan Anda dapat menjalankan beberapa operasi sederhana dengan

kerangka kerja. Anda kemudian dapat beralih ke tutorial yang lebih menyeluruh yang disediakan dengan DLAMI atau contoh kerangka kerja yang ditemukan di setiap situs proyek kerangka kerja.

Masuk ke Anda DLAMI

Setelah Anda masuk ke server Anda, Anda akan melihat server “message of the day” (MOTD) menjelaskan berbagai perintah Conda yang dapat Anda gunakan untuk beralih di antara kerangka kerja pembelajaran mendalam yang berbeda. Di bawah ini adalah contohMOTD. Spesifik Anda MOTD dapat bervariasi saat versi DLAMI baru dirilis.

```
=============================================================================
        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
    =============================================================================
```

Mulai TensorFlow Lingkungan

> **Note**
>
> Saat Anda meluncurkan lingkungan Conda pertama Anda, harap bersabar saat memuat. Pembelajaran Mendalam AMI dengan Conda secara otomatis menginstal versi kerangka

kerja yang paling dioptimalkan untuk EC2 instans Anda pada aktivasi pertama kerangka kerja. Anda seharusnya tidak mengharapkan penundaan berikutnya.

1. Aktifkan lingkungan TensorFlow virtual untuk Python 3.

```
$ source activate tensorflow2_p310
```

2. Mulai iPython terminal.

```
(tensorflow2_p310)$ ipython
```

3. Jalankan TensorFlow program cepat.

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

Anda akan melihat “Halo, Tensorflow!”

Selanjutnya

Menjalankan Tutorial Notebook Jupyter

Beralih ke Lingkungan PyTorch Python 3

Jika Anda masih di iPython konsol, gunakan`quit()`, lalu bersiaplah untuk beralih lingkungan.

- Aktifkan lingkungan PyTorch virtual untuk Python 3.

```
$ source activate pytorch_p310
```

Uji Beberapa PyTorch Kode

Untuk menguji instalasi Anda, gunakan Python untuk menulis PyTorch kode yang membuat dan mencetak array.

1. Mulai iPython terminal.

```
(pytorch_p310)$ ipython
```

2. Impor PyTorch.

```
import torch
```

Anda mungkin melihat pesan peringatan tentang paket pihak ketiga. Anda dapat mengabaikannya.

3. Buat matriks 5x3 dengan elemen yang diinisialisasi secara acak. Cetak array.

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

Verifikasi hasilnya.

```
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]])
```

Menghapus Lingkungan

Jika Anda kehabisan ruang diDLAMI, Anda dapat memilih untuk menghapus paket Conda yang tidak Anda gunakan:

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

Menggunakan Basis Pembelajaran Mendalam AMI

Menggunakan Basis Pembelajaran Mendalam AMI

Base AMI dilengkapi dengan platform dasar GPU driver dan pustaka akselerasi untuk menerapkan lingkungan pembelajaran mendalam Anda sendiri yang disesuaikan. Secara default AMI dikonfigurasi

dengan salah satu lingkungan NVIDIA CUDA versi. Anda juga dapat beralih di antara berbagai versiCUDA. Lihat instruksi berikut untuk cara melakukan ini.

Mengkonfigurasi Versi CUDA

Anda dapat memverifikasi CUDA versi dengan menjalankan NVIDIA `nvcc` program.

```
nvcc --version
```

Anda dapat memilih dan memverifikasi CUDA versi tertentu dengan perintah bash berikut:

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

Untuk informasi selengkapnya, lihat Catatan DLAMI rilis dasar.

Menjalankan Tutorial Notebook Jupyter

Tutorial dan contoh dikirimkan dengan masing-masing sumber proyek pembelajaran mendalam dan dalam banyak kasus mereka akan berjalan di mana saja. DLAMI Jika Anda memilihPembelajaran Mendalam AMI dengan Conda, Anda mendapatkan manfaat tambahan dari beberapa tutorial pilihan yang sudah disiapkan dan siap untuk dicoba.

> **Important**
>
> Untuk menjalankan tutorial notebook Jupyter yang diinstal padaDLAMI, Anda harus melakukannya. Menyiapkan server Jupyter Notebook pada sebuah instance DLAMI

Setelah server Jupyter berjalan, Anda dapat menjalankan tutorial melalui browser web Anda. Jika Anda menjalankan Deep Learning AMI dengan Conda atau jika Anda telah menyiapkan lingkungan Python, Anda dapat mengganti kernel Python dari antarmuka notebook Jupyter. Pilih kernel yang sesuai sebelum mencoba menjalankan tutorial khusus kerangka kerja. Contoh lebih lanjut dari ini disediakan untuk pengguna Deep Learning AMI with Conda.

> **Note**
>
> Banyak tutorial memerlukan modul Python tambahan yang mungkin tidak diatur pada Anda. DLAMI Jika Anda mendapatkan kesalahan seperti`"xyz module not found"`, masuk keDLAMI, aktifkan lingkungan seperti dijelaskan di atas, lalu instal modul yang diperlukan.

> **Tip**
>
> Tutorial dan contoh pembelajaran mendalam sering bergantung pada satu atau lebihGPUs. Jika tipe instans Anda tidak memiliki aGPU, Anda mungkin perlu mengubah beberapa kode contoh untuk menjalankannya.

Menavigasi Tutorial yang Diinstal

Setelah Anda masuk ke server Jupyter dan dapat melihat direktori tutorial (pada Deep Learning AMI with Conda saja), Anda akan disajikan dengan folder tutorial dengan masing-masing nama kerangka kerja. Jika Anda tidak melihat kerangka kerja yang terdaftar, maka tutorial tidak tersedia untuk kerangka kerja itu pada Anda saat iniDLAMI. Klik pada nama framework untuk melihat tutorial yang terdaftar, lalu klik tutorial untuk meluncurkannya.

Pertama kali Anda menjalankan notebook di Deep Learning AMI dengan Conda, itu akan ingin tahu lingkungan mana yang ingin Anda gunakan. Ini akan meminta Anda untuk memilih dari daftar. Setiap lingkungan diberi nama sesuai dengan pola ini:

`Environment (conda_framework_python-version)`

Misalnya, Anda mungkin melihat`Environment (conda_mxnet_p36)`, yang menandakan bahwa lingkungan memiliki MXNet dan Python 3. Variasi lain dari ini adalah`Environment (conda_mxnet_p27)`, yang menandakan bahwa lingkungan memiliki MXNet dan Python 2.

> **Tip**
>
> Jika Anda khawatir tentang versi mana CUDA yang aktif, salah satu cara untuk melihatnya adalah MOTD saat Anda pertama kali masuk ke fileDLAMI.

Beralih Lingkungan dengan Jupyter

Jika Anda memutuskan untuk mencoba tutorial untuk kerangka kerja yang berbeda, pastikan untuk memverifikasi kernel yang sedang berjalan. Info ini dapat dilihat di kanan atas antarmuka Jupyter, tepat di bawah tombol logout. Anda dapat mengubah kernel pada notebook yang terbuka dengan mengklik item menu Jupyter Kernel, lalu Change Kernel, dan kemudian mengklik lingkungan yang sesuai dengan notebook yang sedang Anda jalankan.

Pada titik ini Anda harus menjalankan ulang sel apa pun karena perubahan pada kernel akan menghapus status apa pun yang telah Anda jalankan sebelumnya.

> **Tip**
>
> Beralih antar kerangka kerja bisa menyenangkan dan mendidik, namun Anda bisa kehabisan memori. Jika Anda mulai mengalami kesalahan, lihat jendela terminal yang menjalankan server Jupyter. Ada pesan bermanfaat dan pencatatan kesalahan di sini, dan Anda mungkin melihat out-of-memory kesalahan. Untuk memperbaikinya, Anda dapat pergi ke halaman beranda server Jupyter Anda, klik tab Running, lalu klik Shutdown untuk setiap tutorial yang mungkin masih berjalan di latar belakang dan memakan semua memori Anda.

Tutorial

Berikut ini adalah tutorial tentang cara menggunakan Deep Learning AMI dengan perangkat lunak Conda.

Topik

- Mengaktifkan Kerangka
- Pelatihan terdistribusi menggunakan Adaptor Kain Elastis
- GPUPemantauan dan Optimalisasi
- Chip AWS Inferensia Dengan DLAMI
- The ARM64 DLAMI
- Inferensi
- Penyajian Model

Mengaktifkan Kerangka

Berikut ini adalah kerangka kerja pembelajaran mendalam yang diinstal pada Deep Learning AMI with Conda. Klik pada kerangka kerja untuk mempelajari cara mengaktifkannya.

Topik

PyTorch

Mengaktifkan PyTorch

Ketika paket Conda yang stabil dari kerangka kerja dirilis, itu diuji dan diinstal sebelumnya pada file. DLAMI Jika Anda ingin menjalankan build malam terbaru yang belum teruji, Anda dapat secara manual. Instal PyTorch Nightly Build (eksperimental)

Untuk mengaktifkan kerangka kerja yang saat ini diinstal, ikuti petunjuk ini di Deep Learning Anda AMI dengan Conda.

Untuk PyTorch pada Python 3 dengan CUDA dan MKL -DNN, jalankan perintah ini:

```
$ source activate pytorch_p310
```

Mulai iPython terminal.

```
(pytorch_p310)$ ipython
```

Jalankan PyTorch program cepat.

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

Anda akan melihat array acak awal dicetak, kemudian ukurannya, dan kemudian penambahan array acak lainnya.

Instal PyTorch Nightly Build (eksperimental)

Cara menginstal PyTorch dari build malam

Anda dapat menginstal PyTorch build terbaru ke salah satu atau kedua lingkungan PyTorch Conda di Deep Learning Anda AMI dengan Conda.

1. • (Opsi untuk Python 3) - Aktifkan lingkungan Python 3: PyTorch

   ```
   $ source activate pytorch_p310
   ```

2. Langkah-langkah yang tersisa menganggap Anda menggunakan `pytorch_p310` lingkungan. Hapus yang saat ini diinstal PyTorch:

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

3. • (Opsi untuk GPU instance) - Instal build malam terbaru dengan.0: PyTorch CUDA

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

   • (Opsi untuk CPU instance) - Instal build malam terbaru PyTorch untuk instance tanpa: GPUs

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

4. Untuk memverifikasi bahwa Anda telah berhasil menginstal build nightly terbaru, mulai IPython terminal dan periksa versi. PyTorch

   ```
   (pytorch_p310)$ ipython
   ```

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

   Output harus mencetak sesuatu yang mirip dengan `1.0.0.dev20180922`

5. Untuk memverifikasi bahwa PyTorch nightly build berfungsi dengan baik dengan MNIST contoh, Anda dapat menjalankan skrip pengujian dari repositori contoh PyTorch:

   ```
   (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
```

Lebih Banyak Tutorial

Untuk tutorial dan contoh lebih lanjut, lihat dokumen resmi kerangka kerja, PyTorch dokumentasi, dan PyTorchsitus web.

TensorFlow 2

Tutorial ini menunjukkan cara mengaktifkan TensorFlow 2 pada instance yang menjalankan Deep Learning AMI dengan Conda (DLAMIdi Conda) dan menjalankan program TensorFlow 2.

Ketika paket Conda yang stabil dari kerangka kerja dirilis, itu diuji dan diinstal sebelumnya pada file. DLAMI

Mengaktifkan 2 TensorFlow

Untuk TensorFlow menjalankan DLAMI dengan Conda

1. Untuk mengaktifkan TensorFlow 2, buka instance Amazon Elastic Compute Cloud (AmazonEC2) DLAMI dengan Conda.
2. Untuk TensorFlow 2 dan Keras 2 pada Python 3 dengan CUDA 10.1 dan MKL -DNN, jalankan perintah ini:

   ```
   $ source activate tensorflow2_p310
   ```
3. Mulai iPython terminal:

   ```
   (tensorflow2_p310)$ ipython
   ```
4. Jalankan program TensorFlow 2 untuk memverifikasi bahwa itu berfungsi dengan baik:

   ```


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

   `Hello, TensorFlow!`akan muncul di layar Anda.

Lebih Banyak Tutorial

Untuk tutorial dan contoh lainnya, lihat TensorFlow dokumentasi untuk TensorFlow Python API atau lihat situs webnya. TensorFlow

Pelatihan terdistribusi menggunakan Adaptor Kain Elastis

Elastic Fabric Adapter (EFA) adalah perangkat jaringan yang dapat Anda lampirkan ke DLAMI instans Anda untuk mempercepat aplikasi High Performance Computing (HPC). EFAmemungkinkan Anda mencapai kinerja aplikasi HPC klaster lokal, dengan skalabilitas, fleksibilitas, dan elastisitas yang disediakan oleh Cloud. AWS

Topik berikut menunjukkan kepada Anda bagaimana untuk mulai menggunakan EFA denganDLAMI.

> **Note**
>
> Pilih DLAMI dari GPUDLAMIdaftar Dasar ini

Topik

- Meluncurkan AWS Deep Learning AMIs Instance Dengan EFA
- Menggunakan EFA pada DLAMI

Meluncurkan AWS Deep Learning AMIs Instance Dengan EFA

Base terbaru DLAMI siap digunakan dengan EFA dan dilengkapi dengan driver yang diperlukan, modul kernel, libfabric, openmpi dan NCCLOFIplugin untuk instance. GPU

Anda dapat menemukan CUDA versi Base yang didukung DLAMI di catatan rilis.

Catatan:

- Saat menjalankan NCCL Aplikasi menggunakan `mpirun` onEFA, Anda harus menentukan jalur lengkap ke instalasi yang EFA didukung sebagai:

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

- Untuk mengaktifkan aplikasi Anda untuk digunakanEFA, tambahkan `FI_PROVIDER="efa"` ke `mpirun` perintah seperti yang ditunjukkan padaMenggunakan EFA pada DLAMI.

Topik

Siapkan Grup Keamanan yang EFA Diaktifkan

EFAmembutuhkan grup keamanan yang memungkinkan semua lalu lintas masuk dan keluar ke dan dari grup keamanan itu sendiri. Untuk informasi selengkapnya, lihat EFADokumentasi.

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Di panel navigasi, pilih Grup Keamanan lalu pilih Buat Grup Keamanan.
3. Di jendela Buat Grup Keamanan, lakukan hal berikut:
   - Untuk Nama grup keamanan, masukkan nama deskriptif untuk grup keamanan, seperti `EFA-enabled security group`.
   - (Opsional) Untuk Deskripsi, masukkan deskripsi singkat grup keamanan.
   - Untuk VPC, pilih VPC ke mana Anda ingin meluncurkan instance EFA -enabled Anda.
   - Pilih Buat.
4. Pilih grup keamanan yang Anda buat, dan pada tab Deskripsi, salin ID Grup.
5. Pada tab Inbound dan Outbound, lakukan hal berikut:
   - Pilih Edit.
   - Untuk Jenis, pilih Semua lalu lintas.
   - Untuk Sumber, pilih Kustom.
   - Rekatkan ID grup keamanan yang Anda salin ke bidang.
   - Pilih Simpan.
6. Aktifkan lalu lintas masuk yang mengacu pada Otorisasi Lalu Lintas Masuk untuk Instans Linux Anda. Jika Anda melewati langkah ini, Anda tidak akan dapat berkomunikasi dengan DLAMI instans Anda.

Luncurkan Instance Anda

EFAon saat AWS Deep Learning AMIs ini didukung dengan jenis instance dan sistem operasi berikut:

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

Bagian berikut menunjukkan cara meluncurkan DLAMI instance yang EFA diaktifkan. Untuk informasi selengkapnya tentang meluncurkan instance yang EFA diaktifkan, lihat Meluncurkan Instans EFA yang Diaktifkan ke dalam Grup Penempatan Cluster.

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Pilih Luncurkan Instans.
3. Pada Pilih AMI halaman, pilih yang didukung yang DLAMI ditemukan di Halaman Catatan DLAMI Rilis
4. Pada halaman Pilih Jenis Instance, pilih salah satu jenis instans yang didukung berikut, lalu pilih Berikutnya: Konfigurasi Detail Instance. Lihat tautan ini untuk daftar instance yang didukung: Memulai EFA dan MPI
5. Pada halaman Konfigurasi Detail Instans, lakukan langkah berikut:
   - Untuk Jumlah instans, masukkan jumlah instans EFA -enabled yang ingin Anda luncurkan.
   - Untuk Network dan Subnet, pilih subnet VPC dan untuk meluncurkan instance.
   - [Opsional] Untuk grup Penempatan, pilih Tambahkan instance ke grup penempatan. Untuk performa terbaik, luncurkan instance dalam grup penempatan.
   - [Opsional] Untuk nama grup Penempatan, pilih Tambahkan ke grup penempatan baru, masukkan nama deskriptif untuk grup penempatan, lalu untuk strategi grup Penempatan, pilih klaster.
   - Pastikan untuk mengaktifkan “Adaptor Kain Elastis” di halaman ini. Jika opsi ini dinonaktifkan, ubah subnet menjadi subnet yang mendukung jenis instans yang Anda pilih.
   - Di bagian Antarmuka Jaringan, untuk perangkat eth0, pilih Antarmuka jaringan baru. Anda dapat secara opsional menentukan IPv4 alamat utama dan satu atau lebih IPv4 alamat sekunder. Jika Anda meluncurkan instance ke subnet yang memiliki IPv6 CIDR blok terkait, Anda dapat secara opsional menentukan IPv6 alamat utama dan satu atau beberapa alamat sekunderIPv6.
   - Pilih Berikutnya: Tambahkan Penyimpanan.

6. Pada halaman Tambahkan Penyimpanan, tentukan volume yang akan dilampirkan ke instance selain volume yang ditentukan oleh AMI (seperti volume perangkat root), lalu pilih Berikutnya: Tambahkan Tag.
7. Di halaman Tambahkan Tanda, tentukan tanda untuk instans, seperti nama yang mudah digunakan, lalu pilih Selanjutnya: Konfigurasikan Grup Keamanan.
8. Pada halaman Konfigurasi Grup Keamanan, untuk Menetapkan grup keamanan, pilih Pilih grup keamanan yang ada, lalu pilih grup keamanan yang Anda buat sebelumnya.
9. Pilih Tinjau dan Luncurkan.
10. Di halaman Tinjau Peluncuran Instans, tinjau pengaturannya, lalu pilih Luncurkan untuk memilih pasangan kunci dan meluncurkan instans Anda.

Verifikasi EFA Lampiran

Dari Konsol

Setelah meluncurkan instance, periksa detail instance di AWS Console. Untuk melakukan ini, pilih instance di EC2 konsol dan lihat Tab Deskripsi di panel bawah pada halaman. Temukan parameter 'Network Interfaces: eth0' dan klik eth0 yang membuka pop-up. Pastikan 'Adaptor Kain Elastis' diaktifkan.

Jika tidak EFA diaktifkan, Anda dapat memperbaikinya dengan:

- Mengakhiri EC2 instance dan meluncurkan yang baru dengan langkah yang sama. Pastikan EFA terpasang.
- Lampirkan EFA ke instance yang ada.

  1. Di EC2 konsol, buka Network Interfaces.
  2. Klik Buat Antarmuka Jaringan.
  3. Pilih subnet yang sama dengan instans Anda.
  4. Pastikan untuk mengaktifkan 'Adaptor Kain Elastis' dan klik Buat.
  5. Kembali ke Tab EC2 Instances dan pilih instance Anda.
  6. Buka Actions: Instance State dan hentikan instance sebelum Anda melampirkanEFA.
  7. Dari Tindakan, pilih Jaringan: Lampirkan Antarmuka Jaringan.
  8. Pilih antarmuka yang baru saja Anda buat dan klik lampirkan.
  9. Mulai ulang instans Anda.

Dari Instance

Skrip pengujian berikut sudah ada di fileDLAMI. Jalankan untuk memastikan bahwa modul kernel dimuat dengan benar.

```
$ fi_info -p efa
```

Output-nya semestinya mirip dengan yang berikut.

```
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
```

Verifikasi Konfigurasi Grup Keamanan

Skrip pengujian berikut sudah ada di fileDLAMI. Jalankan untuk memastikan bahwa grup keamanan yang Anda buat dikonfigurasi dengan benar.

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

Output-nya semestinya mirip dengan yang berikut.

```
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
```

Jika berhenti merespons atau tidak selesai, pastikan bahwa grup keamanan Anda memiliki aturan masuk/keluar yang benar.

Menggunakan EFA pada DLAMI

Bagian berikut menjelaskan cara menggunakan EFA untuk menjalankan aplikasi multi-node pada. AWS Deep Learning AMIs

Menjalankan Aplikasi Multi-Node dengan EFA

Untuk menjalankan aplikasi di seluruh cluster node konfigurasi berikut diperlukan

Topik

- Aktifkan Tanpa Kata Sandi SSH
- Buat File Host
- NCCLTes

Aktifkan Tanpa Kata Sandi SSH

Pilih satu node di cluster Anda sebagai node pemimpin. Node yang tersisa disebut sebagai node anggota.

1. Pada node pemimpin, buat RSA keypair.

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

2. Ubah izin kunci privat pada simpul pemimpin.

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

3. Salin kunci publik `~/.ssh/id_rsa.pub` ke dan tambahkan ke `~/.ssh/authorized_keys` node anggota di cluster.

4. Anda sekarang harus dapat langsung masuk ke node anggota dari node pemimpin menggunakan ip pribadi.

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

5. Nonaktifkan strictHostKey Memeriksa dan mengaktifkan penerusan agen pada node pemimpin dengan menambahkan yang berikut ini ke file ~/.ssh/config pada node pemimpin:

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

6. Pada instans Amazon Linux 2, jalankan perintah berikut pada node pemimpin untuk memberikan izin yang benar ke file konfigurasi:

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

Buat File Host

Pada node pemimpin, buat file host untuk mengidentifikasi node di cluster. File host harus memiliki entri untuk setiap node di cluster. Buat file ~/hosts dan tambahkan setiap node menggunakan ip pribadi sebagai berikut:

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

NCCLTes

> **Note**
>
> Tes ini telah dijalankan menggunakan EFA versi 1.30.0 dan OFI NCCL Plugin 1.7.4.

Di bawah ini adalah bagian dari NCCL Pengujian yang disediakan oleh Nvidia untuk menguji fungsionalitas dan kinerja melalui beberapa node komputasi

Instans yang Didukung: P3dn, P4, P5

Tes Fungsionalitas

NCCLTransfer Pesan Tes Multi Node

Nccl_message_transfer adalah tes sederhana untuk memastikan bahwa Plugin berfungsi seperti yang diharapkan. NCCL OFI Tes memvalidasi fungsionalitas pembentukan NCCL koneksi dan transfer APIs data. Pastikan Anda menggunakan path lengkap ke mpirun seperti yang ditunjukkan pada contoh saat menjalankan NCCL aplikasi dengan. EFA Ubah parameter np dan N berdasarkan jumlah instance dan GPUs di cluster Anda. Untuk informasi lebih lanjut, lihat AWS OFINCCLdokumentasi.

Tes nccl_message_transfer berikut adalah untuk versi xx.x generik. CUDA Anda dapat menjalankan perintah untuk CUDA versi apa pun yang tersedia di EC2 instans Amazon Anda dengan mengganti CUDA versi dalam skrip.

```
$/opt/amazon/openmpi/bin/mpirun -n 2 -N 1 --hostfile hosts \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:$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
```

Output Anda akan terlihat seperti berikut ini. Anda dapat memeriksa output untuk melihat yang EFA sedang digunakan sebagai OFI penyedia.

```
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
```

Tes Kinerja

Uji NCCL Kinerja Multi-node pada P4D.24xlarge

Untuk memeriksa NCCL Kinerja denganEFA, jalankan uji NCCL Kinerja standar yang tersedia di Repo NCCL -Tests resmi. DLAMIDilengkapi dengan tes ini yang sudah dibangun untuk CUDA XX.X. Anda juga dapat menjalankan skrip Anda sendiri. EFA

Saat membuat skrip Anda sendiri, lihat panduan berikut:

- Gunakan jalur lengkap ke mpirun seperti yang ditunjukkan pada contoh saat menjalankan NCCL aplikasi dengan. EFA
- Ubah params np dan N berdasarkan jumlah instance dan GPUs di cluster Anda.
- Tambahkan INFO tanda NCCL _ DEBUG = dan pastikan bahwa log menunjukkan EFA penggunaan sebagai “Penyedia Terpilih adalahEFA”.
- Mengatur Lokasi Log Pelatihan untuk mengurai validasi

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

Gunakan perintah `watch nvidia-smi` pada salah satu node anggota untuk memantau GPU penggunaan. `watch nvidia-smi`Perintah berikut adalah untuk versi CUDA xx.x generik dan bergantung pada Sistem Operasi instance Anda. Anda dapat menjalankan perintah untuk CUDA versi apa pun yang tersedia di EC2 instans Amazon Anda dengan mengganti CUDA versi dalam skrip.

- 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}
```

Output Anda akan terlihat seperti berikut:

```
# 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
```

Tes Validasi

Untuk memvalidasi bahwa EFA tes mengembalikan hasil yang valid, silakan gunakan tes berikut untuk mengonfirmasi:

- Dapatkan jenis instance menggunakan EC2 Metadata Instance:

```
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)
```

- Jalankan Tes Kinerja
- Mengatur Parameter Berikut

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Validasi Hasil seperti yang ditunjukkan:

```
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
```

- Untuk mengakses data benchmark, kita dapat mengurai baris terakhir dari output tabel dari tes 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}"
```

GPUPemantauan dan Optimalisasi

Bagian berikut akan memandu Anda melalui opsi GPU pengoptimalan dan pemantauan. Bagian ini diatur seperti alur kerja biasa dengan pemantauan mengawasi prapemrosesan dan pelatihan.

- Pemantauan
  - Monitor GPUs dengan CloudWatch
- Pengoptimalan
  - Pra-pemrosesan
  - Pelatihan

Pemantauan

Anda sudah DLAMI diinstal sebelumnya dengan beberapa alat GPU pemantauan. Panduan ini juga menyebutkan alat yang tersedia untuk diunduh dan dipasang.

- Monitor GPUs dengan CloudWatch- utilitas prainstal yang melaporkan statistik GPU penggunaan ke Amazon CloudWatch.
- nvidia-smi CLI - utilitas untuk memantau GPU komputasi keseluruhan dan pemanfaatan memori. Ini sudah diinstal sebelumnya di AWS Deep Learning AMIs (DLAMI) Anda.
- NVMLC perpustakaan - berbasis C API untuk langsung mengakses fungsi GPU pemantauan dan manajemen. Ini digunakan oleh nvidia-smi CLI di bawah kap dan sudah diinstal sebelumnya pada Anda. DLAMI Ini juga memiliki ikatan Python dan Perl untuk memfasilitasi pengembangan dalam bahasa-bahasa tersebut. Utilitas gpumon.py yang sudah diinstal sebelumnya pada Anda DLAMI menggunakan paket pynvml. dari. nvidia-ml-py
- NVIDIADCGM- Alat manajemen cluster. Kunjungi halaman pengembang untuk mempelajari cara menginstal dan mengkonfigurasi alat ini.

> **Tip**
>
> Lihat blog NVIDIA pengembang untuk info terbaru tentang penggunaan CUDA alat yang diinstal AndaDLAMI:
>
> - Pemantauan TensorCore pemanfaatan menggunakan Nsight IDE dan nvprof.

Monitor GPUs dengan CloudWatch

Ketika Anda menggunakan DLAMI dengan Anda, GPU Anda mungkin menemukan bahwa Anda mencari cara untuk melacak penggunaannya selama pelatihan atau inferensi. Ini dapat berguna untuk mengoptimalkan pipeline data Anda, dan menyetel jaringan pembelajaran mendalam Anda.

Ada dua cara untuk mengonfigurasi GPU metrik dengan CloudWatch:

- Konfigurasikan metrik dengan AWS CloudWatch agen (Disarankan)
- Konfigurasikan metrik dengan skrip yang sudah diinstal sebelumnya `gpumon.py`

Konfigurasikan metrik dengan AWS CloudWatch agen (Disarankan)

Integrasikan Anda DLAMI dengan CloudWatch agen terpadu untuk mengonfigurasi GPU metrik dan memantau pemanfaatan GPU proses bersama dalam instans akselerasi AmazonEC2.

Ada empat cara untuk mengonfigurasi GPUmetrik dengan AndaDLAMI:

- Konfigurasikan GPU metrik minimal
- Konfigurasikan GPU metrik paral
- Konfigurasikan semua GPU metrik yang tersedia
- Konfigurasikan GPU metrik khusus

Untuk informasi tentang pembaruan dan patch keamanan, lihat Penambalan keamanan untuk agen AWS CloudWatch

Prasyarat

Untuk memulai, Anda harus mengonfigurasi IAM izin EC2 instans Amazon yang memungkinkan instans Anda mendorong metrik. CloudWatch Untuk langkah-langkah mendetail, lihat Membuat IAM peran dan pengguna untuk digunakan dengan CloudWatch agen.

Konfigurasikan GPU metrik minimal

Konfigurasikan GPU metrik minimal menggunakan `dlami-cloudwatch-agent@minimal` `systemd` layanan. Layanan ini mengonfigurasi metrik berikut:

- `utilization_gpu`
- `utilization_memory`

Anda dapat menemukan `systemd` layanan untuk GPU metrik minimal yang telah dikonfigurasi sebelumnya di lokasi berikut:

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

Aktifkan dan mulai `systemd` layanan dengan perintah berikut:

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

Konfigurasikan GPU metrik paral

Konfigurasikan GPU metrik sebagian menggunakan `dlami-cloudwatch-agent@partial` `systemd` layanan. Layanan ini mengonfigurasi metrik berikut:

- `utilization_gpu`

- utilization_memory
- memory_total
- memory_used
- memory_free

Anda dapat menemukan systemd layanan untuk GPU metrik sebagian yang telah dikonfigurasi sebelumnya di lokasi berikut:

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

Aktifkan dan mulai systemd layanan dengan perintah berikut:

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

Konfigurasikan semua GPU metrik yang tersedia

Konfigurasikan semua GPU metrik yang tersedia menggunakan dlami-cloudwatch-agent@all systemd layanan. Layanan ini mengonfigurasi metrik berikut:

- 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`

Anda dapat menemukan `systemd` layanan untuk semua GPU metrik yang telah dikonfigurasi sebelumnya yang tersedia di lokasi berikut:

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

Aktifkan dan mulai `systemd` layanan dengan perintah berikut:

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

Konfigurasikan GPU metrik khusus

Jika metrik yang telah dikonfigurasi sebelumnya tidak memenuhi persyaratan Anda, Anda dapat membuat file konfigurasi CloudWatch agen kustom.

Buat file konfigurasi khusus

Untuk membuat file konfigurasi khusus, lihat langkah-langkah terperinci di Buat atau edit file konfigurasi CloudWatch agen secara manual.

Untuk contoh ini, asumsikan bahwa definisi skema terletak di`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`.

Konfigurasikan metrik dengan file kustom Anda

Jalankan perintah berikut untuk mengkonfigurasi CloudWatch agen sesuai dengan file kustom Anda:

```
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
```

Penambalan keamanan untuk agen AWS CloudWatch

Yang baru dirilis DLAMIs dikonfigurasi dengan patch keamanan AWS CloudWatch agen terbaru yang tersedia. Lihat bagian berikut untuk memperbarui Anda saat ini DLAMI dengan patch keamanan terbaru tergantung pada sistem operasi pilihan Anda.

Amazon Linux 2

Gunakan yum untuk mendapatkan patch keamanan AWS CloudWatch agen terbaru untuk Amazon Linux 2DLAMI.

```
sudo yum update
```

Ubuntu

Untuk mendapatkan patch AWS CloudWatch keamanan terbaru untuk DLAMI dengan Ubuntu, perlu menginstal ulang AWS CloudWatch agen menggunakan tautan unduhan Amazon S3.

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

Untuk informasi selengkapnya tentang menginstal AWS CloudWatch agen menggunakan tautan unduhan Amazon S3, lihat Menginstal dan menjalankan CloudWatch agen di server Anda.

Konfigurasikan metrik dengan skrip yang sudah diinstal sebelumnya **gpumon.py**

Sebuah utilitas yang disebut gpumon.py sudah diinstal pada AndaDLAMI. Ini terintegrasi dengan CloudWatch dan mendukung pemantauan per GPU penggunaan: GPU memori, GPU suhu, dan GPU Daya. Script secara berkala mengirimkan data yang dipantau ke CloudWatch. Anda dapat mengonfigurasi tingkat granularitas untuk data yang dikirim CloudWatch dengan mengubah beberapa pengaturan dalam skrip. Namun, sebelum memulai skrip, Anda harus mengatur CloudWatch untuk menerima metrik.

Cara mengatur dan menjalankan GPU pemantauan dengan CloudWatch

1. Buat IAM pengguna, atau ubah pengguna yang sudah ada agar memiliki kebijakan untuk memublikasikan metrik ke CloudWatch. Jika Anda membuat pengguna baru, harap perhatikan kredensialnya karena Anda akan membutuhkannya di langkah berikutnya.

   IAMKebijakan untuk mencari adalah “cloudwatch:PutMetricData”. Kebijakan yang ditambahkan adalah sebagai berikut:

   ```
   {
      "Version": "2012-10-17",
      "Statement": [
           {
               "Action": [
   ```

```
                "cloudwatch:PutMetricData"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Tip**
>
> Untuk informasi selengkapnya tentang membuat IAM pengguna dan menambahkan kebijakan CloudWatch, lihat CloudWatch dokumentasi.

2. Pada AndaDLAMI, jalankan AWS configure dan tentukan kredensi IAM pengguna.

```
$ aws configure
```

3. Anda mungkin perlu membuat beberapa modifikasi pada utilitas gpumon sebelum menjalankannya. Anda dapat menemukan utilitas gpumon dan README di lokasi yang ditentukan dalam blok kode berikut. Untuk informasi selengkapnya tentang `gpumon.py` skrip, lihat lokasi skrip Amazon S3.

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

   Opsi:

   - Ubah wilayah di gpumon.py jika instance Anda ada NOT di us-east-1.
   - Ubah parameter lain seperti CloudWatch `namespace` atau periode pelaporan dengan`store_reso`.

4. Saat ini skrip hanya mendukung Python 3. Aktifkan lingkungan Python 3 kerangka kerja pilihan Anda atau aktifkan lingkungan DLAMI Python 3 umum.

```
$ source activate python3
```

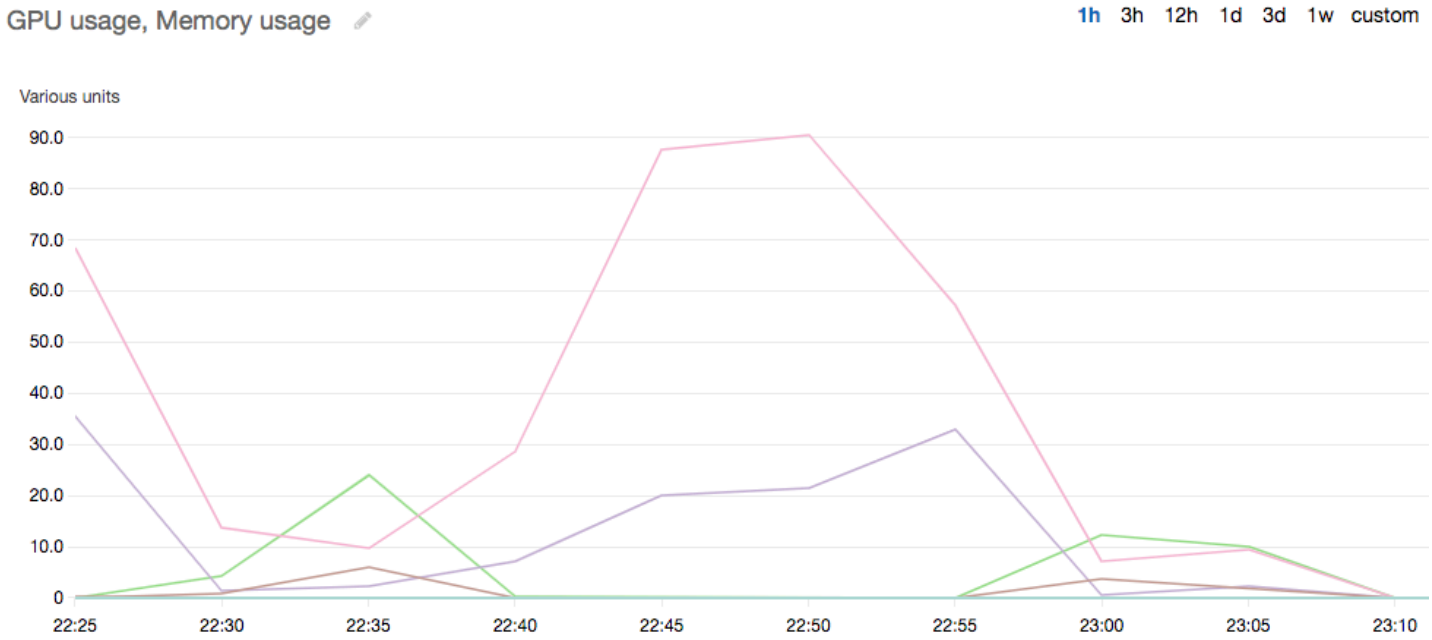
5. Jalankan utilitas gpumon di latar belakang.

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

6. Buka browser Anda ke metrik https://console.aws.amazon.com/cloudwatch/lalu pilih. Ini akan memiliki namespace ". DeepLearningTrain

> **Tip**
>
> Anda dapat mengubah namespace dengan memodifikasi gpumon.py. Anda juga dapat mengubah interval pelaporan dengan menyesuaikan`store_reso`.

Berikut ini adalah contoh CloudWatch bagan pelaporan pada menjalankan gpumon.py memantau pekerjaan pelatihan pada instance p2.8xlarge.



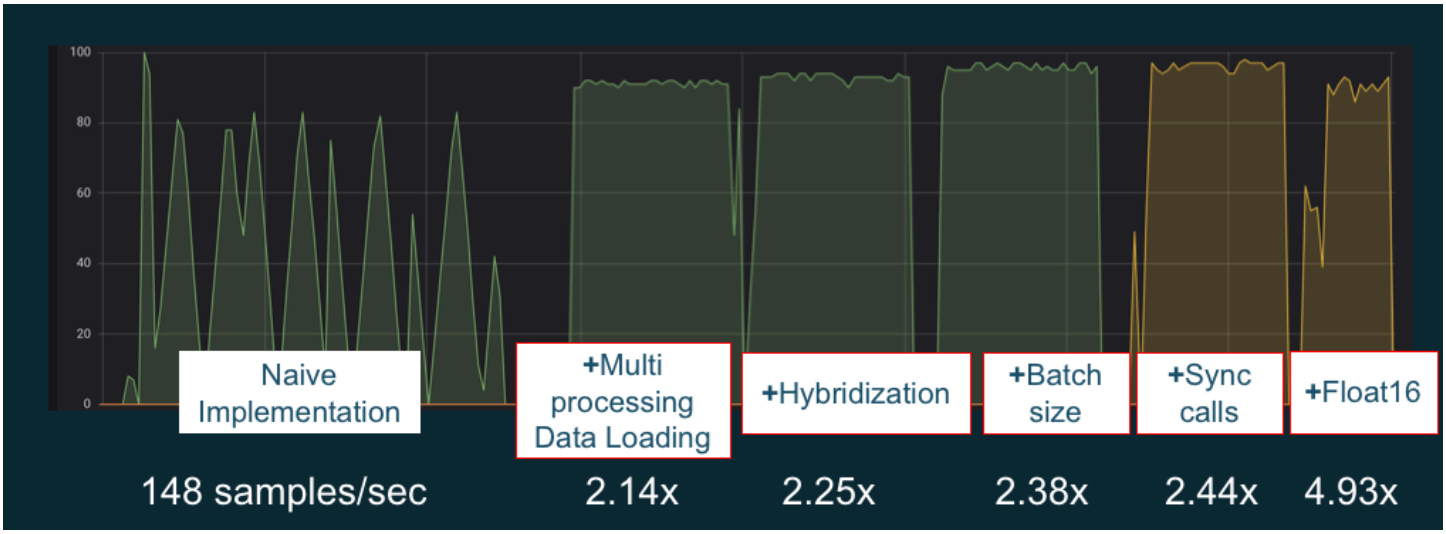
Anda mungkin tertarik dengan topik lain tentang GPU pemantauan dan pengoptimalan:

- Pemantauan
  - Monitor GPUs dengan CloudWatch
- Pengoptimalan
  - Pra-pemrosesan
  - Pelatihan

Pengoptimalan

Untuk memaksimalkanGPUs, Anda dapat mengoptimalkan pipeline data dan menyetel jaringan pembelajaran mendalam Anda. Seperti yang dijelaskan bagan berikut, implementasi naif atau dasar dari jaringan saraf mungkin menggunakan potensi yang tidak GPU konsisten dan tidak sepenuhnya. Ketika Anda mengoptimalkan preprocessing dan pemuatan data Anda, Anda dapat mengurangi hambatan dari Anda ke Anda. CPU GPU Anda dapat menyesuaikan jaringan saraf itu sendiri, dengan menggunakan hibridisasi (bila didukung oleh kerangka kerja), menyesuaikan ukuran batch, dan menyinkronkan panggilan. Anda juga dapat menggunakan pelatihan presisi ganda (float16 atau int8) di sebagian besar kerangka kerja, yang dapat memiliki efek dramatis pada peningkatan throughput.

Bagan berikut menunjukkan peningkatan kinerja kumulatif saat menerapkan pengoptimalan yang berbeda. Hasil Anda akan tergantung pada data yang Anda proses dan jaringan yang Anda optimalkan.



Contoh GPU optimasi kinerja. Sumber bagan: Trik Kinerja dengan MXNet Gluon

Panduan berikut memperkenalkan opsi yang akan bekerja dengan Anda DLAMI dan membantu Anda meningkatkan GPU kinerja.

Topik

- Pra-pemrosesan
- Pelatihan

Pra-pemrosesan

Preprocessing data melalui transformasi atau augmentasi seringkali bisa menjadi proses yang CPU terikat, dan ini bisa menjadi hambatan dalam keseluruhan pipeline Anda. Kerangka kerja memiliki operator bawaan untuk pemrosesan gambar, tetapi DALI (Data Augmentation Library) menunjukkan peningkatan kinerja dibandingkan opsi bawaan kerangka kerja.

- NVIDIAData Augmentation Library (DALI): DALI membongkar augmentasi data ke file. GPU Ini tidak diinstal sebelumnya diDLAMI, tetapi Anda dapat mengaksesnya dengan menginstalnya atau memuat wadah kerangka kerja yang didukung pada instans Amazon Elastic Compute Cloud Anda DLAMI atau lainnya. Lihat halaman DALI proyek di NVIDIA situs web untuk detailnya. Untuk contoh kasus penggunaan dan untuk mengunduh sampel kode, lihat contoh Kinerja Pelatihan SageMaker Preprocessing.
- nvJPEG: perpustakaan JPEG decoder GPU -percepatan untuk programmer C. Ini mendukung decoding gambar tunggal atau batch serta operasi transformasi berikutnya yang umum dalam pembelajaran mendalam. nv JPEG dilengkapi denganDALI, atau Anda dapat mengunduh dari halaman nvjpeg NVIDIAsitus web dan menggunakannya secara terpisah.

Anda mungkin tertarik dengan topik lain tentang GPU pemantauan dan pengoptimalan:

- Pemantauan
  - Monitor GPUs dengan CloudWatch
- Pengoptimalan
  - Pra-pemrosesan
  - Pelatihan

Pelatihan

Dengan pelatihan presisi campuran, Anda dapat menggunakan jaringan yang lebih besar dengan jumlah memori yang sama, atau mengurangi penggunaan memori dibandingkan dengan jaringan presisi tunggal atau ganda Anda, dan Anda akan melihat peningkatan kinerja komputasi. Anda juga mendapatkan manfaat dari transfer data yang lebih kecil dan lebih cepat, faktor penting dalam pelatihan terdistribusi beberapa node. Untuk memanfaatkan pelatihan presisi campuran, Anda perlu menyesuaikan pengecoran data dan penskalaan kerugian. Berikut ini adalah panduan yang menjelaskan cara melakukan ini untuk kerangka kerja yang mendukung presisi campuran.

- NVIDIADeep Learning SDK - dokumen di NVIDIA situs web yang menjelaskan implementasi presisi campuran untukMXNet,, dan. PyTorch TensorFlow

> **Tip**
>
> Pastikan untuk memeriksa situs web untuk kerangka pilihan Anda, dan cari “presisi campuran” atau “fp16" untuk teknik pengoptimalan terbaru. Berikut adalah beberapa panduan presisi campuran yang mungkin berguna bagi Anda:
>
> - Pelatihan presisi campuran dengan TensorFlow (video) - di NVIDIA situs blog.
> - Pelatihan presisi campuran menggunakan float16 dengan MXNet - FAQ artikel di situs web. MXNet
> - NVIDIAApex: alat untuk pelatihan presisi campuran yang mudah dengan PyTorch - artikel blog di situs web. NVIDIA

Anda mungkin tertarik dengan topik lain tentang GPU pemantauan dan pengoptimalan:

- Pemantauan
  - Monitor GPUs dengan CloudWatch
- Pengoptimalan
  - Pra-pemrosesan
  - Pelatihan

Chip AWS Inferensia Dengan DLAMI

AWS Inferentia adalah chip pembelajaran mesin khusus yang dirancang oleh AWS yang dapat Anda gunakan untuk prediksi inferensi kinerja tinggi. Untuk menggunakan chip, siapkan instans Amazon Elastic Compute Cloud dan gunakan kit pengembangan perangkat lunak AWS Neuron (SDK) untuk memanggil chip Inferentia. Untuk memberikan pengalaman Inferensia terbaik kepada pelanggan, Neuron telah dibangun ke dalam AWS Deep Learning AMIs (DLAMI).

Topik berikut menunjukkan kepada Anda bagaimana memulai menggunakan Inferentia dengan. DLAMI

Daftar Isi

- Meluncurkan DLAMI Instance dengan AWS Neuron
- Menggunakan DLAMI AWS Neuron

Meluncurkan DLAMI Instance dengan AWS Neuron

Yang terbaru DLAMI siap digunakan dengan AWS Inferentia dan dilengkapi dengan API paket AWS Neuron. Untuk meluncurkan DLAMI instance, lihat Meluncurkan dan Mengkonfigurasi. DLAMI Setelah Anda memilikinyaDLAMI, gunakan langkah-langkah di sini untuk memastikan bahwa chip AWS Inferentia dan sumber daya AWS Neuron Anda aktif.

Daftar Isi

Verifikasi Instance Anda

Sebelum menggunakan instance Anda, verifikasi bahwa itu diatur dan dikonfigurasi dengan benar dengan Neuron.

Mengidentifikasi Perangkat AWS Inferensia

Untuk mengidentifikasi jumlah perangkat Inferentia pada instans Anda, gunakan perintah berikut:

```
neuron-ls
```

Jika instans Anda memiliki perangkat Inferentia yang terpasang padanya, output Anda akan terlihat mirip dengan yang berikut ini:

```
+--------+--------+--------+-----------+--------------+
| 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 |
+--------+--------+--------+-----------+--------------+
```

Output yang disediakan diambil dari instance INF1.6xLarge dan menyertakan kolom berikut:

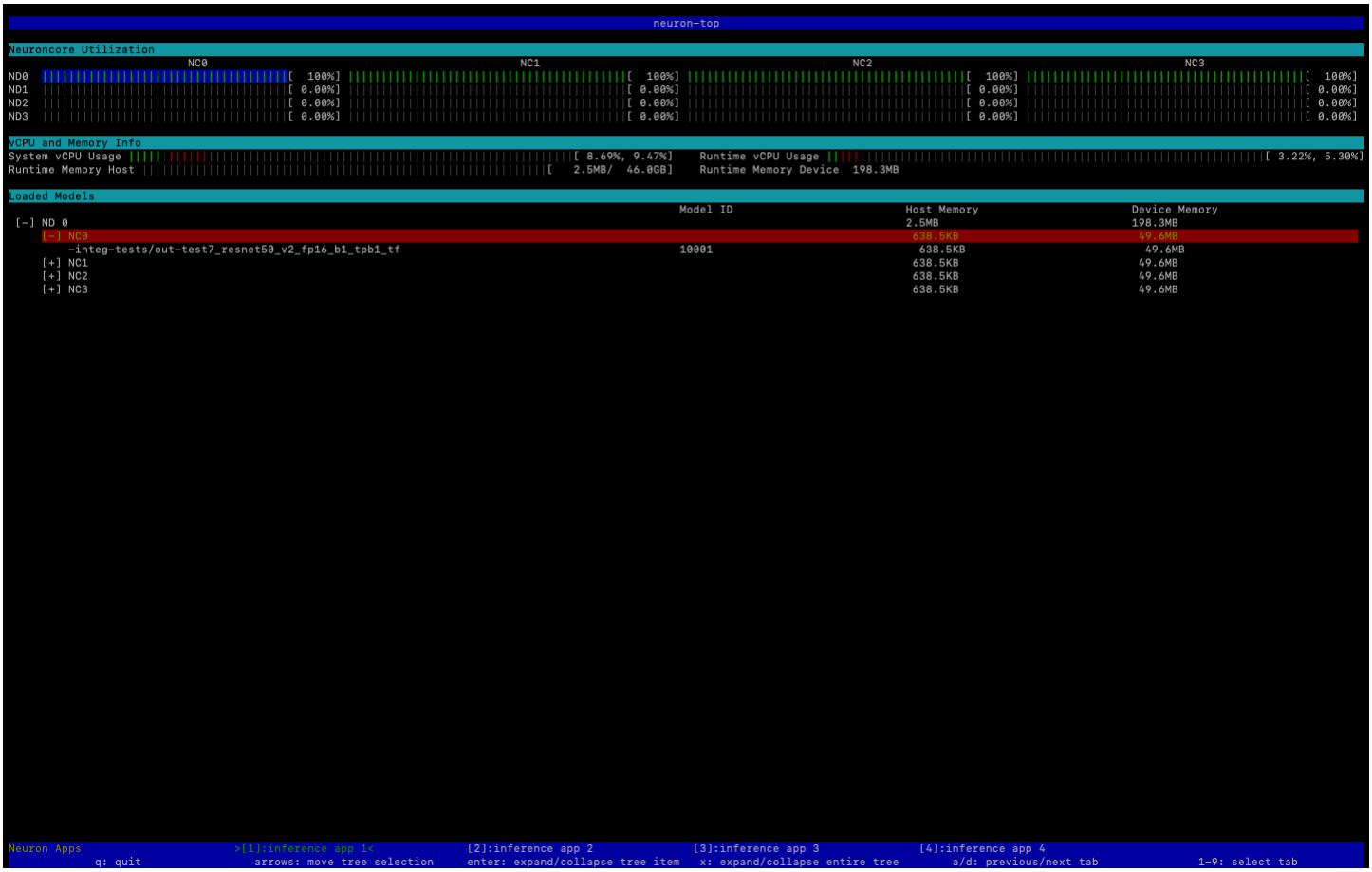
- NEURONDEVICE: ID logis yang ditetapkan untuk NeuronDevice. ID ini digunakan saat mengonfigurasi beberapa runtime untuk menggunakan yang berbeda. NeuronDevices
- NEURONCORES: Jumlah NeuronCores hadiah di NeuronDevice.
- NEURONMEMORY: Jumlah DRAM memori di NeuronDevice.
- CONNECTEDDEVICES: Lainnya NeuronDevices terhubung ke NeuronDevice.
- PCIBDF: Fungsi Perangkat PCI Bus (BDF) ID dari NeuronDevice.

Lihat Penggunaan Sumber Daya

Lihat informasi yang berguna tentang NeuronCore dan v CPU pemanfaatan, penggunaan memori, model yang dimuat, dan aplikasi Neuron dengan `neuron-top` perintah. `neuron-top`Peluncuran tanpa argumen akan menampilkan data untuk semua aplikasi pembelajaran mesin yang memanfaatkan NeuronCores.

```
neuron-top
```

Ketika aplikasi menggunakan empat NeuronCores, output akan terlihat mirip dengan gambar berikut:



Untuk informasi lebih lanjut tentang sumber daya untuk memantau dan mengoptimalkan aplikasi inferensi berbasis Neuron, lihat Alat Neuron.

Menggunakan Neuron Monitor (neuron-monitor)

Neuron Monitor mengumpulkan metrik dari runtime Neuron yang berjalan di sistem dan mengalirkan data yang dikumpulkan ke stdout dalam format. JSON Metrik ini diatur ke dalam grup metrik yang Anda konfigurasikan dengan menyediakan file konfigurasi. Untuk informasi lebih lanjut tentang Monitor Neuron, lihat Panduan Pengguna untuk Monitor Neuron.

Meningkatkan Perangkat Lunak Neuron

Untuk informasi tentang cara memperbarui SDK perangkat lunak Neuron di dalamnyaDLAMI, lihat Panduan Pengaturan AWS Neuron.

Langkah Selanjutnya

Menggunakan DLAMI AWS Neuron

Menggunakan DLAMI AWS Neuron

Alur kerja khas dengan AWS Neuron SDK adalah mengkompilasi model pembelajaran mesin yang dilatih sebelumnya di server kompilasi. Setelah ini, distribusikan artefak ke instance Inf1 untuk dieksekusi. AWS Deep Learning AMIs (DLAMI) datang pra-instal dengan semua yang Anda butuhkan untuk mengkompilasi dan menjalankan inferensi dalam instance Inf1 yang menggunakan Inferentia.

Bagian berikut menjelaskan cara menggunakan DLAMI Inferentia.

Daftar Isi

Menggunakan TensorFlow -Neuron dan Kompiler AWS Neuron

Tutorial ini menunjukkan cara menggunakan kompiler AWS Neuron untuk mengkompilasi model Keras ResNet -50 dan mengekspornya sebagai model yang disimpan dalam format. SavedModel Format ini adalah format TensorFlow model yang dapat dipertukarkan khas. Anda juga belajar cara menjalankan inferensi pada instance Inf1 dengan input contoh.

Untuk informasi lebih lanjut tentang NeuronSDK, lihat SDKdokumentasi AWS Neuron.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan DLAMI Instance dengan AWS Neuron. Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan. DLAMI

Aktifkan lingkungan Conda

Aktifkan lingkungan conda TensorFlow -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_tensorflow_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan perintah berikut:

```
source deactivate
```

Resnet50 Kompilasi

Buat skrip Python yang disebut **`tensorflow_compile_resnet50.py`** yang memiliki konten berikut. Skrip Python ini mengkompilasi model Keras ResNet 50 dan mengekspornya sebagai model yang disimpan.

```
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')
```

Kompilasi model menggunakan perintah berikut:

```
python tensorflow_compile_resnet50.py
```

Proses kompilasi akan memakan waktu beberapa menit. Ketika selesai, output Anda akan terlihat seperti berikut:

```
...
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
...
```

Setelah kompilasi, model yang disimpan di-zip di**`ws_resnet50/resnet50_neuron.zip`**. Buka zip model dan unduh gambar sampel untuk inferensi menggunakan perintah berikut:

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

ResNet50 Inferensi

Buat skrip Python yang disebut **`tensorflow_infer_resnet50.py`** yang memiliki konten berikut. Skrip ini menjalankan inferensi pada model yang diunduh menggunakan model inferensi yang dikompilasi sebelumnya.

```
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])
```

Jalankan inferensi pada model menggunakan perintah berikut:

```
python tensorflow_infer_resnet50.py
```

Output Anda akan terlihat seperti berikut:

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

Langkah Selanjutnya

[Menggunakan TensorFlow Penyajian AWS Neuron]

Menggunakan TensorFlow Penyajian AWS Neuron

Tutorial ini menunjukkan cara membuat grafik dan menambahkan langkah kompilasi AWS Neuron sebelum mengekspor model yang disimpan untuk digunakan dengan TensorFlow Serving. TensorFlow Melayani adalah sistem penyajian yang memungkinkan Anda meningkatkan inferensi di seluruh jaringan. Neuron TensorFlow Serving menggunakan sama dengan API TensorFlow Serving normal. Satu-satunya perbedaan adalah bahwa model yang disimpan harus dikompilasi untuk AWS Inferentia dan titik masuknya adalah biner yang berbeda bernama`tensorflow_model_server_neuron`. Biner ditemukan di `/usr/local/bin/tensorflow_model_server_neuron` dan sudah diinstal sebelumnya di. DLAMI

Untuk informasi lebih lanjut tentang NeuronSDK, lihat SDKdokumentasi AWS Neuron.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan DLAMI Instance dengan AWS Neuron. Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan. DLAMI

Aktifkan lingkungan Conda

Aktifkan lingkungan conda TensorFlow -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_tensorflow_p36
```

Jika Anda perlu keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Kompilasi dan Ekspor Model Tersimpan

Buat skrip Python yang disebut `tensorflow-model-server-compile.py` dengan konten berikut. Skrip ini membuat grafik dan mengkompilasinya menggunakan Neuron. Kemudian mengekspor grafik yang dikompilasi sebagai model yang disimpan.

```
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)
```

Kompilasi model menggunakan perintah berikut:

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

Output Anda akan terlihat seperti berikut:

```
...
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
```

Melayani Model Tersimpan

Setelah model dikompilasi, Anda dapat menggunakan perintah berikut untuk menyajikan model yang disimpan dengan biner tensorflow_model_server_neuron:

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

Output Anda akan terlihat seperti berikut ini. Model yang dikompilasi dipentaskan di perangkat Inferentia DRAM oleh server untuk mempersiapkan inferensi.

```
...
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 ...
```

Hasilkan permintaan inferensi ke server model

Membuat skrip Python yang disebut `tensorflow-model-server-infer.py` dengan konten berikut. Skrip ini menjalankan inferensi melalui gRPC, yang merupakan kerangka kerja layanan.

```
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))
```

Jalankan inferensi pada model dengan menggunakan g RPC dengan perintah berikut:

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

Output Anda akan terlihat seperti berikut:

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

Menggunakan MXNet -Neuron dan Kompiler AWS Neuron

Kompilasi MXNet -Neuron API menyediakan metode untuk mengkompilasi grafik model yang dapat Anda jalankan pada perangkat AWS Inferentia.

Dalam contoh ini, Anda menggunakan API untuk mengkompilasi model ResNet -50 dan menggunakannya untuk menjalankan inferensi.

Untuk informasi lebih lanjut tentang NeuronSDK, lihat SDKdokumentasi AWS Neuron.

Daftar Isi

- ResNet50 Inferensi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan DLAMI Instance dengan AWS Neuron. Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan. DLAMI

Aktifkan Lingkungan Conda

Aktifkan lingkungan conda MXNet -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_mxnet_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Resnet50 Kompilasi

Membuat skrip Python yang disebut **`mxnet_compile_resnet50.py`** dengan konten berikut. Skrip ini menggunakan kompilasi MXNet -Neuron API Python untuk mengkompilasi ResNet model -50.

```
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)
```

Kompilasi model menggunakan perintah berikut:

```
python mxnet_compile_resnet50.py
```

Kompilasi akan memakan waktu beberapa menit. Ketika kompilasi telah selesai, file-file berikut akan berada di direktori Anda saat ini:

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

ResNet50 Inferensi

Membuat skrip Python yang disebut **`mxnet_infer_resnet50.py`** dengan konten berikut. Skrip ini mengunduh gambar sampel dan menggunakannya untuk menjalankan inferensi dengan model yang dikompilasi.

```
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]))
```

Jalankan inferensi dengan model yang dikompilasi menggunakan perintah berikut:

```
python mxnet_infer_resnet50.py
```

Output Anda akan terlihat seperti berikut:

```
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
```

Langkah Selanjutnya

Menggunakan Penyajian Model MXNet -Neuron

Menggunakan Penyajian Model MXNet -Neuron

Dalam tutorial ini, Anda belajar menggunakan MXNet model pra-terlatih untuk melakukan klasifikasi gambar real-time dengan Multi Model Server (MMS). MMSadalah easy-to-use alat yang fleksibel dan untuk melayani model pembelajaran mendalam yang dilatih menggunakan pembelajaran mesin atau

kerangka pembelajaran mendalam. Tutorial ini mencakup langkah kompilasi menggunakan AWS Neuron dan implementasi MMS penggunaanMXNet.

Untuk informasi lebih lanjut tentang NeuronSDK, lihat SDKdokumentasi AWS Neuron.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan DLAMI Instance dengan AWS Neuron. Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan. DLAMI

Aktifkan Lingkungan Conda

Aktifkan lingkungan conda MXNet -Neuron dengan menggunakan perintah berikut:

```
source activate aws_neuron_mxnet_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Unduh Kode Contoh

Untuk menjalankan contoh ini, unduh kode contoh menggunakan perintah berikut:

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

Kompilasi Model

Membuat skrip Python yang disebut `multi-model-server-compile.py` dengan konten berikut. Skrip ini mengkompilasi model ResNet 50 ke target perangkat 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)
```

Untuk mengkompilasi model, gunakan perintah berikut:

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

Output Anda akan terlihat seperti berikut:

```
...
[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!
```

Buat file bernama `signature.json` dengan konten berikut untuk mengkonfigurasi nama input dan bentuk:

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

Download `synset.txt` file dengan menggunakan perintah berikut. File ini adalah daftar nama untuk kelas ImageNet prediksi.

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

Buat kelas layanan kustom mengikuti template di `model_server_template` folder. Salin template ke direktori kerja Anda saat ini dengan menggunakan perintah berikut:

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

Edit `mxnet_model_service.py` modul untuk mengganti `mx.cpu()` konteks dengan `mx.neuron()` konteks sebagai berikut. Anda juga perlu mengomentari salinan data yang tidak perlu `model_input` karena MXNet -Neuron tidak mendukung NDArray dan APIs Gluon.

```
...
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]
```

Package model dengan model-archiver menggunakan perintah berikut:

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

Jalankan Inferensi

Mulai Multi Model Server dan muat model yang menggunakan RESTful API dengan menggunakan perintah berikut. Pastikan neuron-rtd itu berjalan dengan pengaturan default.

```
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
```

Jalankan inferensi menggunakan contoh gambar dengan perintah berikut:

```
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
```

Output Anda akan terlihat seperti berikut:

```
[
  {
    "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"
  }
]
```

Untuk membersihkan setelah pengujian, keluarkan perintah hapus melalui RESTful API dan hentikan server model menggunakan perintah berikut:

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

multi-model-server --stop
```

Anda akan melihat output berikut:

```
{
  "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
```

Menggunakan PyTorch -Neuron dan Kompiler AWS Neuron

Kompilasi PyTorch -Neuron API menyediakan metode untuk mengkompilasi grafik model yang dapat Anda jalankan pada perangkat AWS Inferentia.

Model terlatih harus dikompilasi ke target Inferentia sebelum dapat digunakan pada instance Inf1. Tutorial berikut mengkompilasi model torchvision ResNet 50 dan mengekspornya sebagai modul yang disimpan. TorchScript Model ini kemudian digunakan untuk menjalankan inferensi.

Untuk kenyamanan, tutorial ini menggunakan instance Inf1 untuk kompilasi dan inferensi. Dalam praktiknya, Anda dapat mengkompilasi model Anda menggunakan tipe instance lain, seperti keluarga instance c5. Anda kemudian harus menerapkan model yang dikompilasi ke server inferensi Inf1. Untuk informasi lebih lanjut, lihat PyTorch SDKDokumentasi AWS Neuron.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan DLAMI Instance dengan AWS Neuron. Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan. DLAMI

Aktifkan Lingkungan Conda

Aktifkan lingkungan conda PyTorch -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_pytorch_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Resnet50 Kompilasi

Membuat skrip Python yang disebut **`pytorch_trace_resnet50.py`** dengan konten berikut. Skrip ini menggunakan kompilasi PyTorch -Neuron API Python untuk mengkompilasi ResNet model -50.

> **Note**
>
> Ada ketergantungan antara versi torchvision dan paket obor yang harus Anda ketahui saat mengkompilasi model torchvision. Aturan ketergantungan ini dapat dikelola melalui pip. Torchvision==0.6.1 cocok dengan rilis torch==1.5.1, sedangkan torchvision==0.8.2 cocok dengan rilis 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")
```

Jalankan skrip kompilasi.

```
python pytorch_trace_resnet50.py
```

Kompilasi akan memakan waktu beberapa menit. Ketika kompilasi telah selesai, model yang dikompilasi disimpan seperti `resnet50_neuron.pt` di direktori lokal.

ResNet50 Inferensi

Membuat skrip Python yang disebut **`pytorch_infer_resnet50.py`** dengan konten berikut. Skrip ini mengunduh gambar sampel dan menggunakannya untuk menjalankan inferensi dengan model yang dikompilasi.

```
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) )
```

Jalankan inferensi dengan model yang dikompilasi menggunakan perintah berikut:

```
python pytorch_infer_resnet50.py
```

Output Anda akan terlihat seperti berikut:

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

The ARM64 DLAMI

AWS ARM64GPUDLAMIsdirancang untuk memberikan kinerja tinggi dan efisiensi biaya untuk beban kerja pembelajaran mendalam. Secara khusus, tipe instans G5G menampilkan prosesor AWS Graviton2 berbasis ARM64, yang dibangun dari bawah ke atas AWS dan dioptimalkan untuk bagaimana pelanggan menjalankan beban kerja mereka di cloud. AWS ARM64GPUDLAMIstelah dikonfigurasi sebelumnya dengan Docker, NVIDIA Docker, NVIDIA DriverCUDA,, Cu DNNNCCL,, serta kerangka kerja pembelajaran mesin populer seperti dan. TensorFlow PyTorch

Dengan tipe instans G5G, Anda dapat memanfaatkan manfaat harga dan kinerja Graviton2 untuk menerapkan model pembelajaran mendalam yang GPU dipercepat dengan biaya yang jauh lebih rendah jika dibandingkan dengan instans berbasis x86 dengan akselerasi. GPU

Pilih ARM64 DLAMI

Luncurkan instans G5G dengan ARM64 DLAMI pilihan Anda.

Untuk step-by-step petunjuk tentang meluncurkanDLAMI, lihat Meluncurkan dan Mengonfigurasi file. DLAMI

Untuk daftar yang terbaru ARM64DLAMIs, lihat Catatan Rilis untuk DLAMI.

Mulai

Topik berikut menunjukkan kepada Anda bagaimana untuk mulai menggunakan ARM64DLAMI.

Daftar Isi

Menggunakan ARM64 GPU PyTorch DLAMI

AWS Deep Learning AMIs Ini siap digunakan dengan berbasis prosesor Arm64GPUs, dan dioptimalkan untuk. PyTorch ARM64GPU PyTorch DLAMITermasuk lingkungan Python yang

telah dikonfigurasi sebelumnya dengan PyTorch, TorchVision, dan TorchServeuntuk pelatihan pembelajaran mendalam dan kasus penggunaan inferensi.

Daftar Isi

Verifikasi PyTorch Lingkungan Python

Hubungkan ke instans G5G Anda dan aktifkan lingkungan dasar Conda dengan perintah berikut:

```
source activate base
```

Prompt perintah Anda harus menunjukkan bahwa Anda bekerja di lingkungan dasar Conda, yang berisi PyTorch TorchVision, dan pustaka lainnya.

```
(base) $
```

Verifikasi jalur alat default PyTorch lingkungan:

```
(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)
```

Jalankan Sampel Pelatihan dengan PyTorch

Jalankan contoh pekerjaan MNIST pelatihan:

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

Output-nya semestinya mirip dengan yang berikut:

```
...
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%)
```

Jalankan Sampel Inferensi dengan PyTorch

Gunakan perintah berikut untuk mengunduh model densenet161 yang telah dilatih sebelumnya dan jalankan inferensi menggunakan: 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
```

Output-nya semestinya mirip dengan yang berikut:

```
{
  "tiger_cat": 0.4693363308906555,
  "tabby": 0.4633873701095581,
  "Egyptian_cat": 0.06456123292446136,
  "lynx": 0.0012828150065615773,
  "plastic_bag": 0.00023522898778133094
}
```

Gunakan perintah berikut untuk membatalkan pendaftaran model densenet161 dan menghentikan server:

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

Output-nya semestinya mirip dengan yang berikut:

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

Inferensi

Bagian ini memberikan tutorial tentang cara menjalankan inferensi menggunakan kerangka kerja dan alat. DLAMI

Alat Inferensi

- TensorFlow Melayani

Penyajian Model

Berikut ini adalah opsi penyajian model yang diinstal pada Deep Learning AMI with Conda. Klik salah satu opsi untuk mempelajari cara menggunakannya.

Topik

TensorFlow Melayani

TensorFlow Melayani adalah sistem penyajian yang fleksibel dan berkinerja tinggi untuk model pembelajaran mesin.

`tensorflow-serving-api`Ini sudah diinstal sebelumnya dengan Deep Learning AMI dengan Conda! Anda akan menemukan contoh skrip untuk melatih, mengekspor, dan melayani MNIST model. `~/examples/tensorflow-serving/`

Untuk menjalankan salah satu contoh ini, pertama-tama hubungkan ke Deep Learning Anda AMI dengan Conda dan aktifkan TensorFlow lingkungan.

```
$ source activate tensorflow2_p310
```

Sekarang ubah direktori ke folder skrip contoh penyajian.

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

Melayani Model Inception yang Terlatih

Berikut ini adalah contoh yang dapat Anda coba untuk melayani model yang berbeda seperti Inception. Sebagai aturan umum, Anda memerlukan model servable dan skrip klien untuk sudah diunduh ke Anda. DLAMI

Melayani dan Menguji Inferensi dengan Model Inception

1. Unduh modelnya.

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

2. Untar modelnya.

   ```
   $ unzip INCEPTION.zip
   ```

3. Unduh gambar husky.

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

4. Luncurkan server. Perhatikan, bahwa untuk Amazon Linux, Anda harus mengubah direktori yang digunakan untuk`model_base_path`, dari `/home/ubuntu` ke`/home/ec2-user`.

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

5. Dengan server berjalan di latar depan, Anda perlu meluncurkan sesi terminal lain untuk melanjutkan. Buka terminal baru dan aktifkan TensorFlow dengan`source activate tensorflow2_p310`. Kemudian gunakan editor teks pilihan Anda untuk membuat skrip yang memiliki konten berikut. Namai itu`inception_client.py`. Skrip ini akan mengambil nama file gambar sebagai parameter, dan mendapatkan hasil prediksi dari model yang telah dilatih sebelumnya.

```
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. Sekarang jalankan skrip melewati lokasi server dan port dan nama file foto husky sebagai parameter.

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

Melatih dan Melayani MNIST Model

Untuk tutorial ini kita akan mengekspor model kemudian menyajikannya dengan `tensorflow_model_server` aplikasi. Akhirnya, Anda dapat menguji server model dengan contoh skrip klien.

Jalankan skrip yang akan melatih dan mengekspor MNIST model. Sebagai satu-satunya argumen skrip, Anda perlu menyediakan lokasi folder untuk menyimpan model. Untuk saat ini kita bisa memasukkannya`mnist_model`. Script akan membuat folder untuk Anda.

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

Bersabarlah, karena skrip ini mungkin memakan waktu beberapa saat sebelum memberikan output apa pun. Ketika pelatihan selesai dan model akhirnya diekspor, Anda akan melihat yang berikut:

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

Langkah Anda selanjutnya adalah menjalankan `tensorflow_model_server` untuk melayani model yang diekspor.

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

Skrip klien disediakan bagi Anda untuk menguji server.

Untuk mengujinya, Anda harus membuka jendela terminal baru.

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

Lebih Banyak Fitur dan Contoh

Jika Anda tertarik untuk mempelajari lebih lanjut tentang TensorFlow Melayani, lihat TensorFlow situs webnya.

Anda juga dapat menggunakan TensorFlow Serving dengan Amazon Elastic Inference. Lihat panduan tentang cara Menggunakan Elastic Inference with TensorFlow Serving untuk info lebih lanjut.

TorchServe

TorchServe adalah alat yang fleksibel untuk melayani model pembelajaran mendalam yang telah diekspor dari PyTorch. TorchServe datang pra-instal dengan Deep Learning AMI dengan Conda.

Untuk informasi selengkapnya tentang penggunaan TorchServe, lihat Server Model untuk PyTorch Dokumentasi.

Topik

Sajikan Model Klasifikasi Gambar pada TorchServe

Tutorial ini menunjukkan cara menyajikan model klasifikasi gambar dengan TorchServe. Ini menggunakan model DenseNet -161 yang disediakan oleh. PyTorch Setelah server berjalan, ia mendengarkan permintaan prediksi. Saat Anda mengunggah gambar, dalam hal ini, gambar anak kucing, server mengembalikan prediksi 5 kelas pencocokan teratas dari kelas tempat model dilatih.

Untuk menyajikan contoh model klasifikasi gambar pada TorchServe

1. Connect ke instans Amazon Elastic Compute Cloud (AmazonEC2) dengan Deep Learning AMI dengan Conda v34 atau versi lebih baru.
2. Aktifkan `pytorch_p310` lingkungan.

```
source activate pytorch_p310
```

3. Kloning TorchServe repositori, lalu buat direktori untuk menyimpan model Anda.

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

4. Arsipkan model menggunakan pengarsipan model. `extra-files`Param menggunakan file dari `TorchServe` repo, jadi perbarui jalur jika perlu. Untuk informasi selengkapnya tentang pengarsipan model, lihat Pengarsip Model Obor untuk. 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. Jalankan TorchServe untuk memulai titik akhir. Menambahkan `> /dev/null` menenangkan output log.

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

6. Unduh gambar anak kucing dan kirimkan ke titik akhir TorchServe prediksi:

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

Titik akhir prediksi mengembalikan prediksi yang JSON mirip dengan lima prediksi teratas berikut, di mana gambar memiliki probabilitas 47% mengandung kucing Mesir, diikuti oleh kemungkinan 46% memiliki kucing kucing kucing.

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

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

7. Setelah Anda selesai menguji, hentikan server:

```
torchserve --stop
```

Contoh Lain

TorchServe memiliki berbagai contoh yang dapat Anda jalankan pada DLAMI instance Anda. Anda dapat melihatnya di halaman contoh repositori TorchServe proyek.

Info Lebih Lanjut

Untuk TorchServe dokumentasi lebih lanjut, termasuk cara mengatur TorchServe dengan Docker dan TorchServe fitur terbaru, lihat halaman TorchServe proyek di GitHub.

Upgrade Anda DLAMI

Di sini Anda akan menemukan informasi tentang meningkatkan DLAMI dan tips memperbarui perangkat lunak pada AndaDLAMI.

Selalu perbarui sistem operasi Anda dan perangkat lunak terinstal lainnya dengan menerapkan tambalan dan pembaruan segera setelah tersedia.

Jika Anda menggunakan Amazon Linux atau Ubuntu, ketika Anda masuk ke AndaDLAMI, Anda diberi tahu jika pembaruan tersedia dan melihat instruksi untuk memperbarui. Untuk informasi lebih lanjut tentang pemeliharaan Amazon Linux, lihat Memperbarui Perangkat Lunak Instans. Untuk contoh Ubuntu, lihat dokumentasi resmi Ubuntu.

Di Windows, periksa Pembaruan Windows secara teratur untuk pembaruan perangkat lunak dan keamanan. Jika Anda mau, sediakan pembaruan secara otomatis.

> **Important**
>
> Untuk informasi tentang kerentanan Meltdown dan Spectre dan cara menambal sistem operasi Anda untuk mengatasinya, lihat Buletin Keamanan -2018-013. AWS

Topik

Upgrade ke Versi Baru DLAMI

DLAMIGambar sistem diperbarui secara teratur untuk memanfaatkan rilis kerangka pembelajaran mendalam yang baru, CUDA dan pembaruan perangkat lunak lainnya, dan penyetelan kinerja. Jika Anda telah menggunakan DLAMI untuk beberapa waktu dan ingin memanfaatkan pembaruan, Anda perlu meluncurkan instance baru. Anda juga harus mentransfer kumpulan data, pos pemeriksaan, atau data berharga lainnya secara manual. Sebagai gantinya, Anda dapat menggunakan Amazon EBS untuk menyimpan data Anda dan melampirkannya ke yang baruDLAMI. Dengan cara ini, Anda dapat sering meningkatkan, sambil meminimalkan waktu yang diperlukan untuk mentransisikan data Anda.

> **Note**
>
> Saat melampirkan dan memindahkan EBS volume AmazonDLAMIs, Anda harus memiliki volume DLAMIs dan volume baru di Availability Zone yang sama.

1. Gunakan Amazon EC2console untuk membuat EBS volume Amazon baru. Untuk petunjuk mendetail, lihat Membuat EBS Volume Amazon.
2. Lampirkan EBS volume Amazon yang baru Anda buat ke yang sudah adaDLAMI. Untuk petunjuk mendetail, lihat Melampirkan EBS Volume Amazon.
3. Transfer data Anda, seperti dataset, pos pemeriksaan, dan file konfigurasi.
4. Luncurkan aDLAMI. Untuk petunjuk terperinci, lihat Menyiapkan sebuah DLAMI instance.
5. Lepaskan EBS volume Amazon dari yang lamaDLAMI. Untuk petunjuk mendetail, lihat Melepaskan EBS Volume Amazon.
6. Lampirkan EBS volume Amazon ke volume baru AndaDLAMI. Ikuti instruksi dari Langkah 2 untuk melampirkan volume.
7. Setelah Anda memverifikasi bahwa data Anda tersedia di data baru AndaDLAMI, hentikan dan hentikan data lama DLAMI Anda. Untuk instruksi pembersihan terperinci, lihat. Membersihkan sebuah DLAMI instance

Kiat untuk Pembaruan Perangkat Lunak

Dari waktu ke waktu, Anda mungkin ingin memperbarui perangkat lunak secara manual pada perangkat lunak AndaDLAMI. Hal ini umumnya dianjurkan bahwa Anda menggunakan `pip` untuk memperbarui paket Python. Anda juga harus menggunakan `pip` untuk memperbarui paket dalam lingkungan Conda di Deep Learning AMI dengan Conda. Lihat situs web kerangka kerja atau perangkat lunak tertentu untuk instruksi peningkatan dan penginstalan.

> **Note**
>
> Kami tidak dapat menjamin bahwa pembaruan paket akan berhasil. Mencoba memperbarui paket di lingkungan dengan dependensi yang tidak kompatibel dapat mengakibatkan kegagalan. Dalam kasus seperti itu, Anda harus menghubungi pengelola perpustakaan untuk melihat apakah mungkin untuk memperbarui dependensi paket. Atau, Anda dapat mencoba memodifikasi lingkungan sedemikian rupa sehingga memungkinkan pembaruan. Namun,

modifikasi ini kemungkinan berarti menghapus atau memperbarui paket yang ada, yang berarti bahwa kami tidak dapat lagi menjamin stabilitas lingkungan ini.

AWS Deep Learning AMIs Muncul dengan banyak lingkungan Conda dan banyak paket yang sudah diinstal sebelumnya. Karena jumlah paket yang sudah diinstal sebelumnya, sulit menemukan satu set paket yang dijamin kompatibel. Anda mungkin melihat peringatan "Lingkungan tidak konsisten, silakan periksa paket paket dengan cermat". DLAMImemastikan bahwa semua lingkungan DLAMI yang disediakan sudah benar, tetapi tidak dapat menjamin bahwa setiap paket yang diinstal pengguna akan berfungsi dengan benar.

Menerima Pemberitahuan tentang Pembaruan Baru

Note

AWS Deep Learning AMIs memiliki irama rilis mingguan untuk patch keamanan. Pemberitahuan rilis akan dikirim untuk tambalan keamanan tambahan ini meskipun mungkin tidak disertakan dalam catatan rilis resmi.

Anda dapat menerima pemberitahuan setiap kali baru DLAMI dirilis. Pemberitahuan diterbitkan dengan Amazon SNS menggunakan topik berikut.

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

Pesan diposting di sini saat DLAMI yang baru diterbitkan. Versi, metadata, dan AMI ID regional AMI akan disertakan dalam pesan.

Pesan-pesan ini dapat diterima menggunakan beberapa metode berbeda. Kami menyarankan Anda menggunakan metode berikut.

1. Buka SNSkonsol Amazon.
2. Di bilah navigasi, ubah AWS Wilayah ke AS Barat (Oregon), jika perlu. Anda harus memilih wilayah tempat SNS pemberitahuan yang Anda berlangganan dibuat.
3. Di panel navigasi, pilih Langganan, Buat langganan.
4. Untuk kotak dialog Buat langganan, lakukan hal berikut:

a. Untuk Topik ARN, salin dan tempel Nama Sumber Daya Amazon berikut (ARN): **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**

b. Untuk Protokol, pilih salah satu dari [AmazonSQS, AWS Lamda, Email, Email- JSON]

c. Untuk Endpoint, masukkan alamat email atau Amazon Resource Name (ARN) sumber daya yang akan Anda gunakan untuk menerima notifikasi.

d. Pilih Buat langganan.

5. Anda menerima email konfirmasi dengan baris subjek AWS Pemberitahuan - Konfirmasi Langganan. Buka email dan pilih Konfirmasi berlangganan untuk menyelesaikan langganan Anda.

Keamanan di AWS Deep Learning AMIs

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program AWS Kepatuhan Program AWS Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku AWS Deep Learning AMIs, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakanDLAMI. Topik berikut menunjukkan cara mengonfigurasi DLAMI untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan DLAMI sumber daya Anda.

Untuk informasi selengkapnya, lihat Keamanan EC2 di Amazon di Panduan EC2 Pengguna Amazon.

Topik

Perlindungan data di AWS Deep Learning AMIs

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS Deep Learning AMIs. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat Privasi Data FAQ. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting GDPR blog di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management ()IAM. Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan otentikasi multi-faktor (MFA) dengan setiap akun.
- GunakanSSL/TLSuntuk berkomunikasi dengan AWS sumber daya. Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan FIPS 140-3 modul kriptografi yang divalidasi saat mengakses AWS melalui antarmuka baris perintah atau, gunakan titik akhir. API FIPS Untuk informasi selengkapnya tentang FIPS titik akhir yang tersedia, lihat Federal Information Processing Standard (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk ketika Anda bekerja dengan DLAMI atau lainnya Layanan AWS menggunakan konsol,API, AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan

atau log diagnostik. Jika Anda memberikan URL ke server eksternal, kami sangat menyarankan agar Anda tidak menyertakan informasi kredensyal dalam URL untuk memvalidasi permintaan Anda ke server tersebut.

Identitas dan manajemen akses untuk AWS Deep Learning AMIs

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. IAMadministrator mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. DLAMI IAMadalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Untuk informasi selengkapnya tentang identitas dan manajemen akses, lihat Identitas dan manajemen akses untuk Amazon EC2.

Topik

- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- IAMdengan Amazon EMR

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai IAM pengguna, atau dengan mengambil peranIAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensi yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (Pusat IAM Identitas), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan IAM peran. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara

kriptografis dengan menggunakan kredensyal Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani AWS API permintaan di Panduan IAM Pengguna.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor di Panduan AWS IAM Identity Center Pengguna dan Menggunakan otentikasi multi-faktor (MFA) AWS di Panduan Pengguna. IAM

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensyal pengguna root di IAMPanduan Pengguna.

Pengguna dan grup IAM

IAMPengguna adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensyal sementara daripada membuat IAM pengguna yang memiliki kredensyal jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensil jangka panjang dengan IAM pengguna, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensyal jangka panjang di IAMPanduan Pengguna.

IAMGrup adalah identitas yang menentukan kumpulan IAM pengguna. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup bernama IAMAdminsdan memberikan izin grup tersebut untuk mengelola sumber dayaIAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari lebih lanjut, lihat Kapan membuat IAM pengguna (bukan peran) di Panduan IAM Pengguna.

IAMperan

IAMPeran adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Ini mirip dengan IAM pengguna, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil IAM peran sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil AWS CLI atau AWS API operasi atau dengan menggunakan kustomURL. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Metode untuk mengambil peran dalam Panduan IAM Pengguna.

IAMperan dengan kredensi sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan IAM Pengguna. Jika Anda menggunakan Pusat IAM Identitas, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, Pusat IAM Identitas menghubungkan izin yang disetel ke peran. IAM Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin IAM pengguna sementara — IAM Pengguna atau peran dapat mengambil IAM peran untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun — Anda dapat menggunakan IAM peran untuk memungkinkan seseorang (prinsipal tepercaya) di akun lain mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin

melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat FAS permintaan, lihat Meneruskan sesi akses.
- Peran layanan — Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam IAMPanduan Pengguna.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan IAM peran untuk mengelola kredenal sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS API meminta. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM peran untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di IAMPanduan Pengguna.

Untuk mempelajari apakah akan menggunakan IAM peran atau IAM pengguna, lihat Kapan membuat IAM peran (bukan pengguna) di Panduan IAM Pengguna.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna

root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai JSON dokumen. Untuk informasi selengkapnya tentang struktur dan isi dokumen JSON kebijakan, lihat Ringkasan JSON kebijakan di Panduan IAM Pengguna.

Administrator dapat menggunakan AWS JSON kebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

IAMkebijakan menentukan izin untuk tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasi. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan itu bisa mendapatkan informasi peran dari AWS Management Console, AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat IAM kebijakan di Panduan Pengguna. IAM

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan sebaris, lihat Memilih antara kebijakan terkelola dan kebijakan sebaris di IAMPanduan Pengguna.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu.

Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLsmirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan. JSON

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnyaACLs, lihat Ikhtisar daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk masing-masing Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan IAM Pengguna.

IAMdengan Amazon EMR

Anda dapat menggunakan IAM Amazon EMR untuk menentukan pengguna, AWS sumber daya, grup, peran, dan kebijakan. Anda juga dapat mengontrol pengguna dan peran mana Layanan AWS yang dapat diakses.

Untuk informasi selengkapnya tentang penggunaan IAM dengan AmazonEMR, lihat AWS Identity and Access Management Amazon EMR.

Validasi kepatuhan untuk AWS Deep Learning AMIs

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS Deep Learning AMIs sebagai bagian dari beberapa program AWS kepatuhan. Untuk informasi tentang program kepatuhan yang didukung, lihat Validasi kepatuhan untuk Amazon EC2.

Untuk daftar cakupan program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup menurut AWS Layanan Program Kepatuhan . Layanan AWS Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di Laporan Artefak di. AWS Artifact

Tanggung jawab kepatuhan Anda saat menggunakan DLAMI ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Quick Start Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah–langkah untuk melakukan deployment terhadap lingkungan dasar di AWS yang menjadi fokus keamanan dan kepatuhan.
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan AWS Config Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi AWS sumber daya Anda dan untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di AWS Deep Learning AMIs

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

Untuk informasi tentang EC2 fitur Amazon untuk membantu mendukung ketahanan data dan kebutuhan pencadangan, lihat Ketahanan di Amazon EC2 di Panduan Pengguna Amazon EC2.

Keamanan infrastruktur di AWS Deep Learning AMIs

Keamanan infrastruktur AWS Deep Learning AMIs didukung oleh AmazonEC2. Untuk informasi selengkapnya, lihat Keamanan infrastruktur EC2 di Amazon di Panduan EC2 Pengguna Amazon.

AWS Deep Learning AMIs Contoh pemantauan

Pemantauan adalah bagian penting untuk menjaga keandalan, ketersediaan, dan kinerja AWS Deep Learning AMIs instans Anda dan AWS solusi Anda yang lain. DLAMIInstans Anda dilengkapi dengan beberapa alat GPU pemantauan, termasuk utilitas yang melaporkan statistik GPU penggunaan ke Amazon CloudWatch. Untuk informasi selengkapnya, lihatGPUPemantauan dan Optimalisasi, dan lihat Memantau EC2 sumber daya Amazon di Panduan EC2 Pengguna Amazon.

Memilih keluar dari pelacakan penggunaan untuk instance DLAMI

Distribusi sistem AWS Deep Learning AMIs operasi berikut mencakup kode yang memungkinkan AWS untuk mengumpulkan jenis instans, ID instance, DLAMI tipe, dan informasi OS.

> **Note**
>
> AWS tidak mengumpulkan atau menyimpan informasi lain tentangDLAMI, seperti perintah yang Anda gunakan dalamDLAMI.

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

Untuk memilih keluar dari pelacakan penggunaan

Jika Anda memilih, Anda dapat memilih keluar dari pelacakan penggunaan untuk DLAMI instance baru. Untuk memilih keluar, Anda harus menambahkan tag ke EC2 instans Amazon Anda selama peluncuran. Tag harus menggunakan kunci `OPT_OUT_TRACKING` dengan nilai terkait yang disetel `ketrue`. Untuk informasi selengkapnya, lihat Menandai EC2 sumber daya Amazon Anda di Panduan EC2 Pengguna Amazon.

DLAMIkebijakan dukungan kerangka kerja

Di sini Anda dapat menemukan rincian kebijakan dukungan untuk AWS Deep Learning AMIs (DLAMI) kerangka kerja.

Untuk daftar DLAMI kerangka kerja yang AWS saat ini mendukung, lihat halaman DLAMIFramework Support Policy. Dalam tabel di halaman itu, ingatlah hal-hal berikut:

- Versi saat ini menentukan versi kerangka kerja dalam format x.y.z. Dalam format ini, x mengacu pada versi utama, y mengacu pada versi minor, dan z mengacu pada versi patch. Misalnya, untuk TensorFlow 2.10.1, versi utama adalah 2, versi minor adalah 10, dan versi tambalan adalah 1.
- Akhir patch menentukan berapa lama AWS mendukung versi kerangka kerja itu.

Untuk informasi rinci tentang spesifikDLAMIs, lihatCatatan rilis untuk DLAMIs.

DLAMIdukungan kerangka kerja FAQs

- Versi kerangka kerja apa yang mendapatkan tambalan keamanan?
- Gambar apa yang dilakukan AWS mempublikasikan ketika versi kerangka kerja baru dirilis?
- Gambar apa yang baru SageMaker/AWS fitur?
- Bagaimana versi saat ini didefinisikan dalam tabel Kerangka Kerja yang Didukung?
- Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?
- Apakah DLAMIs mendukung versi sebelumnya TensorFlow?
- Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?
- Seberapa sering gambar baru dirilis?
- Apakah instance saya akan ditambal di tempat saat beban kerja saya berjalan?
- Apa yang terjadi ketika versi kerangka kerja baru yang ditambal atau diperbarui tersedia?
- Apakah dependensi diperbarui tanpa mengubah versi kerangka kerja?
- Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?
- Akankah gambar dengan versi kerangka kerja yang tidak lagi dipelihara secara aktif ditambal?
- Bagaimana cara menggunakan versi kerangka kerja yang lebih lama?

- Bagaimana cara saya tetap up-to-date dengan perubahan dukungan dalam kerangka kerja dan versinya?
- Apakah saya memerlukan lisensi komersial untuk menggunakan Repositori Anaconda?

Versi kerangka kerja apa yang mendapatkan tambalan keamanan?

Jika versi kerangka kerja diberi label Didukung di AWS Deep Learning AMIs Tabel Framework Support Policy, ia mendapat patch keamanan.

Gambar apa yang dilakukan AWS mempublikasikan ketika versi kerangka kerja baru dirilis?

Kami menerbitkan baru DLAMIs segera setelah versi baru TensorFlow dan PyTorch dirilis. Ini termasuk versi mayor, versi mayor-minor, dan major-minor-patch versi kerangka kerja. Kami juga memperbarui gambar saat versi driver dan pustaka baru tersedia. Untuk informasi lebih lanjut tentang pemeliharaan gambar, lihat Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

Gambar apa yang baru SageMaker/AWS fitur?

Fitur baru biasanya dirilis dalam versi terbaru DLAMIs untuk PyTorch dan TensorFlow. Lihat catatan rilis untuk gambar tertentu untuk detail tentang yang baru SageMaker atau AWS fitur. Untuk daftar yang tersediaDLAMIs, lihat Catatan Rilis untuk DLAMI. Untuk informasi lebih lanjut tentang pemeliharaan gambar, lihat Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

Bagaimana versi saat ini didefinisikan dalam tabel Kerangka Kerja yang Didukung?

Versi saat ini di AWS Deep Learning AMIs Tabel Framework Support Policy mengacu pada versi framework terbaru yang AWS membuat tersedia di GitHub. Setiap rilis terbaru mencakup pembaruan untuk driver, pustaka, dan paket yang relevan di file. DLAMI Untuk informasi tentang pemeliharaan gambar, lihat Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?

Jika Anda menjalankan versi yang tidak ada di AWS Deep Learning AMIs Tabel Kebijakan Dukungan Kerangka Kerja, Anda mungkin tidak memiliki driver, pustaka, dan paket yang relevan yang paling

diperbarui. Untuk up-to-date versi lainnya, kami menyarankan Anda meningkatkan ke salah satu kerangka kerja yang didukung yang tersedia menggunakan yang terbaru DLAMI pilihan Anda. Untuk daftar yang tersediaDLAMIs, lihat Catatan Rilis untuk DLAMI.

Apakah DLAMIs mendukung versi sebelumnya TensorFlow?

Tidak. Kami mendukung versi patch terbaru dari setiap versi utama terbaru kerangka kerja yang dirilis 365 hari dari GitHub rilis awal seperti yang dinyatakan dalam AWS Deep Learning AMIs Tabel Kebijakan Dukungan Kerangka Kerja. Untuk informasi selengkapnya, silakan lihat Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?

Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?

Untuk menggunakan DLAMI dengan versi framework terbaru, ambil DLAMIID dan gunakan untuk meluncurkan DLAMI menggunakan EC2Console. Untuk sampel AWS CLIperintah untuk mengambil AWS Deep Learning AMIs ID, lihat halaman catatan DLAMI rilis catatan DLAMI rilis kerangka tunggal. Versi kerangka kerja yang Anda pilih harus diberi label Didukung di AWS Deep Learning AMIs Tabel Kebijakan Dukungan Kerangka Kerja.

Seberapa sering gambar baru dirilis?

Menyediakan versi tambalan yang diperbarui adalah prioritas tertinggi kami. Kami secara rutin membuat gambar yang ditambal pada kesempatan paling awal. Kami memantau versi kerangka kerja yang baru ditambal (mis. TensorFlow 2.9 hingga TensorFlow 2.9.1) dan versi rilis minor baru (mis. TensorFlow 2.9 hingga TensorFlow 2.10) dan membuatnya tersedia pada kesempatan paling awal. Ketika versi yang TensorFlow ada dirilis dengan versi baruCUDA, kami merilis versi baru DLAMI untuk versi tersebut TensorFlow dengan dukungan untuk CUDA versi baru.

Apakah instance saya akan ditambal di tempat saat beban kerja saya berjalan?

Tidak. Pembaruan tambalan untuk DLAMI bukan pembaruan “di tempat”.

Anda harus mengaktifkan EC2 instance baru, memigrasikan beban kerja dan skrip, lalu mematikan instance sebelumnya.

Apa yang terjadi ketika versi kerangka kerja baru yang ditambal atau diperbarui tersedia?

Periksa halaman catatan rilis secara teratur untuk gambar Anda. Kami mendorong Anda untuk meningkatkan ke kerangka kerja baru yang ditambal atau diperbarui saat tersedia. Untuk daftar yang tersediaDLAMIs, lihat Catatan Rilis untuk DLAMI.

Apakah dependensi diperbarui tanpa mengubah versi kerangka kerja?

Kami memperbarui dependensi tanpa mengubah versi kerangka kerja. Namun, jika pembaruan ketergantungan menyebabkan ketidakcocokan, kami membuat gambar dengan versi yang berbeda. Pastikan untuk memeriksa Catatan Rilis DLAMI untuk informasi ketergantungan yang diperbarui.

Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

DLAMIgambar tidak dapat diubah. Begitu mereka diciptakan, mereka tidak berubah. Ada empat alasan utama mengapa dukungan aktif untuk versi kerangka kerja berakhir:

- Versi kerangka kerja (patch) upgrade
- AWS patch keamanan
- Akhir tanggal tambalan (Aging out)
- Ketergantungan end-of-support

> **Note**
>
> Karena frekuensi upgrade versi patch dan patch keamanan, kami sarankan untuk memeriksa halaman catatan rilis untuk DLAMI sering Anda, dan upgrade ketika perubahan dilakukan.

Versi kerangka kerja (patch) upgrade

Jika Anda memiliki DLAMI beban kerja berdasarkan TensorFlow 2.7.0 dan TensorFlow merilis versi 2.7.1 aktif, maka GitHub AWS merilis yang baru DLAMI dengan TensorFlow 2.7.1. Gambar sebelumnya dengan 2.7.0 tidak lagi dipertahankan secara aktif setelah gambar baru dengan TensorFlow 2.7.1 dirilis. DLAMIDengan TensorFlow 2.7.0 tidak menerima tambalan lebih lanjut. Halaman catatan DLAMI rilis untuk TensorFlow 2.7 kemudian diperbarui dengan informasi terbaru. Tidak ada halaman catatan rilis individual untuk setiap tambalan kecil.

Baru DLAMIs dibuat karena upgrade patch ditunjuk dengan AMIID baru.

AWS patch keamanan

Jika Anda memiliki beban kerja berdasarkan gambar dengan TensorFlow 2.7.0 dan AWS membuat patch keamanan, kemudian versi baru dirilis untuk TensorFlow 2.7.0. DLAMI Versi gambar sebelumnya dengan TensorFlow 2.7.0 tidak lagi dipertahankan secara aktif. Untuk informasi selengkapnya, lihat Apakah instance saya akan ditambal di tempat saat beban kerja saya berjalan? Untuk langkah-langkah menemukan yang terbaruDLAMI, lihat Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?

Baru DLAMIs dibuat karena upgrade patch ditunjuk dengan AMIID baru.

Akhir tanggal tambalan (Aging out)

DLAMIsmencapai akhir tanggal patch mereka 365 hari setelah tanggal GitHub rilis.

Untuk multi-kerangka kerja DLAMIs, ketika salah satu versi kerangka diperbarui, yang baru DLAMI dengan versi yang diperbarui diperlukan. DLAMIDengan versi kerangka lama tidak lagi dipertahankan secara aktif.

> **Important**
>
> Kami membuat pengecualian ketika ada pembaruan kerangka kerja utama. Sebagai contoh. jika TensorFlow 1.15 memperbarui ke TensorFlow 2.0, maka kami terus mendukung versi terbaru TensorFlow 1.15 untuk jangka waktu dua tahun sejak tanggal GitHub rilis atau enam bulan setelah tim pemeliharaan kerangka kerja asal menjatuhkan dukungan, tanggal mana pun yang lebih awal.

Ketergantungan end-of-support

Jika Anda menjalankan beban kerja pada gambar TensorFlow 2.7.0 DLAMI dengan Python 3.6 dan versi Python ditandai, end-of-support maka semua gambar DLAMI berdasarkan Python 3.6 tidak akan lagi dipertahankan secara aktif. Demikian pula, jika versi OS seperti Ubuntu 16.04 ditandai untuk end-of-support, maka semua DLAMI gambar yang bergantung pada Ubuntu 16.04 tidak akan lagi dipertahankan secara aktif.

Akankah gambar dengan versi kerangka kerja yang tidak lagi dipelihara secara aktif ditambal?

Tidak. Gambar yang tidak lagi dipelihara secara aktif tidak akan memiliki rilis baru.

Bagaimana cara menggunakan versi kerangka kerja yang lebih lama?

Untuk menggunakan versi kerangka kerja yang lebih lama, ambil DLAMIID dan gunakan untuk meluncurkan DLAMI menggunakan EC2Konsol. DLAMI Untuk AWS CLIperintah untuk mengambil AMI ID, lihat halaman catatan rilis dalam catatan DLAMIrilis kerangka tunggal.

Bagaimana cara saya tetap up-to-date dengan perubahan dukungan dalam kerangka kerja dan versinya?

Tetap up-to-date dengan DLAMI kerangka kerja dan versi menggunakan AWS Deep Learning AMIs Tabel Framework Support Policy, catatan DLAMI rilis.

Apakah saya memerlukan lisensi komersial untuk menggunakan Repositori Anaconda?

Anaconda beralih ke model lisensi komersial untuk pengguna tertentu. Dipelihara secara aktif DLAMIs telah dimigrasikan ke versi open-source Conda (conda-forge) yang tersedia untuk umum dari saluran Anaconda.

NVIDIADriver penting berubah menjadi DLAMIs

Pada 15 November 2023, AWS membuat perubahan penting untuk AWS Deep Learning AMIs (DLAMI) terkait dengan NIVIDA driver yang DLAMIs menggunakan. Untuk informasi tentang apa yang berubah dan apakah hal itu memengaruhi penggunaan AndaDLAMIs, lihatDLAMINVIDIAperubahan pengemudi FAQs.

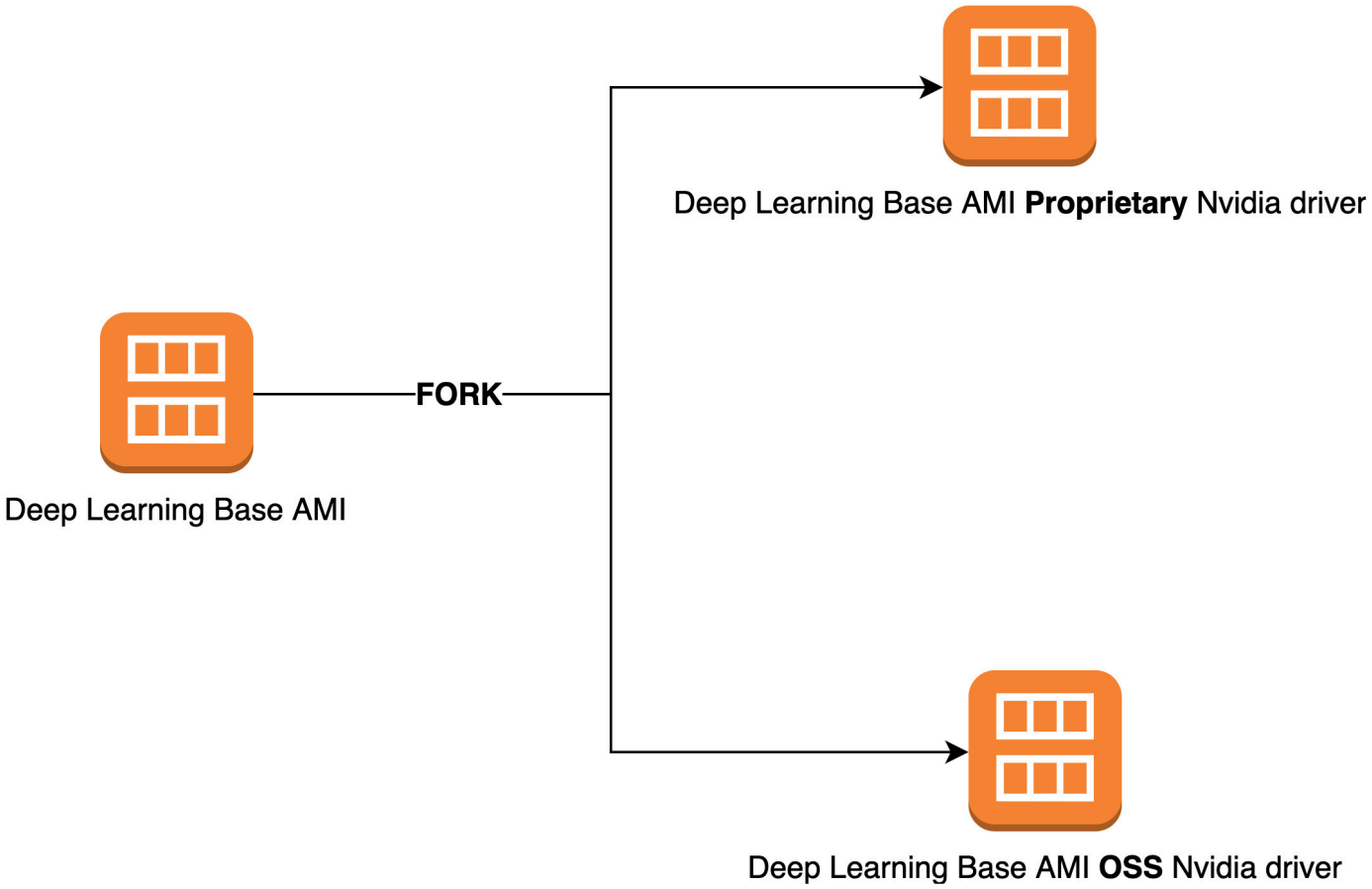
DLAMINVIDIAperubahan pengemudi FAQs

Apa yang berubah?

Kami membagi DLAMIs menjadi dua kelompok terpisah:

- DLAMIsyang menggunakan driver NVIDIA berpemilik (untuk mendukung P3, P3dn, G3)
- DLAMIsyang menggunakan NVIDIA OSS driver (untuk mendukung G4dn, G5, P4, P5)

Hasilnya, kami membuat baru DLAMIs untuk masing-masing dari dua kategori dengan nama baru dan baru AMIIDs. Ini DLAMIs tidak bisa dipertukarkan. Artinya, DLAMIs dari satu grup tidak mendukung contoh yang didukung grup lain. Misalnya, DLAMI yang mendukung P5 tidak mendukung G3, dan DLAMI yang mendukung G3 tidak mendukung P5.



Mengapa perubahan ini diperlukan?

Sebelumnya, DLAMIs untuk NVIDIA GPUs disertakan driver kernel berpemilik dariNVIDIA. Namun, komunitas kernel Linux hulu menerima perubahan yang mengisolasi driver kernel berpemilik, seperti driver, dari berkomunikasi dengan NVIDIA GPU driver kernel lainnya. Perubahan ini dinonaktifkan GPUDirect RDMA pada instans seri P4 dan P5, yang merupakan mekanisme yang memungkinkan GPUs untuk digunakan EFA secara efisien untuk pelatihan terdistribusi. Akibatnya, DLAMIs sekarang gunakan driver OpenRM (driver NVIDIA open source), yang ditautkan dengan driver open source EFA untuk mendukung G4dn, G5, P4, dan P5. Namun, driver OpenRM ini tidak mendukung instance lama (seperti P3 dan G3). Oleh karena itu, untuk memastikan bahwa kami terus menyediakan arus, berkinerja, dan aman DLAMIs yang mendukung kedua jenis instans, kami membagi DLAMIs menjadi dua kelompok: satu dengan driver OpenRM (yang mendukung G4dn, G5, P4, dan P5), dan satu dengan driver berpemilik yang lebih lama (yang mendukung P3, P3dn, dan G3).

DLAMIsApa yang mempengaruhi perubahan ini?

Perubahan ini mempengaruhi semuaDLAMIs.

Apa artinya ini bagi Anda?

Semua DLAMIs akan terus menyediakan fungsionalitas, kinerja, dan keamanan selama Anda menjalankannya pada jenis instans Amazon Elastic Compute Cloud (AmazonEC2) yang didukung. Untuk menentukan jenis EC2 instance yang DLAMI didukung, periksa catatan rilis untuk ituDLAMI, lalu cari EC2Instans yang Didukung. Untuk daftar DLAMI opsi yang saat ini didukung dan tautan ke catatan rilis mereka, lihatCatatan rilis untuk DLAMIs.

Selain itu, Anda harus menggunakan yang benar AWS Command Line Interface (AWS CLI) perintah untuk memanggil arusDLAMIs.

Untuk basis DLAMIs yang mendukung P3, P3dn, dan G3, gunakan perintah ini:

```
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
```

Untuk basis DLAMIs yang mendukung G4dn, G5, P4, dan P5, gunakan perintah ini:

```
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
```

Apakah ada kehilangan fungsionalitas dengan yang lebih baru? DLAMIs

Tidak, tidak ada kehilangan fungsionalitas. Saat ini DLAMIs menyediakan semua fungsionalitas, kinerja, dan keamanan yang sebelumnyaDLAMIs, asalkan Anda menjalankannya pada jenis EC2 instans yang didukung.

Apakah perubahan ini memengaruhi Deep Learning Containers?

Tidak, perubahan ini tidak mempengaruhi AWS Deep Learning Containers, karena tidak termasuk NVIDIA driver. Namun, pastikan untuk menjalankan Deep Learning Containers AMIs yang kompatibel dengan instance yang mendasarinya.

Informasi terkait tentang DLAMI

Anda dapat menemukan sumber daya lain dengan informasi terkait DLAMI di luar AWS Deep Learning AMIs Panduan Pengembang. Pada AWS re:Post, periksa pertanyaan tentang DLAMI dari pelanggan lain, atau ajukan pertanyaan Anda sendiri. Pada AWS Machine Learning Blog dan lainnya AWS blog, baca posting resmi tentangDLAMI.

AWS re:Post

Tag: AWS Deep Learning AMIs

AWS Blog

- AWS Machine Learning Blog | Kategori: AWS Deep Learning AMIs
- AWS Machine Learning Blog | Pelatihan lebih cepat dengan TensorFlow 1.6 yang dioptimalkan di instans Amazon EC2 C5 dan P3
- AWS Machine Learning Blog | Baru AWS Deep Learning AMIs untuk Praktisi Machine Learning
- AWS Partner Network (APN) Blog | Kursus Pelatihan Baru Tersedia: Pengantar Machine Learning & Deep Learning on AWS
- AWS Blog Berita | Perjalanan ke Pembelajaran Mendalam dengan AWS

Catatan rilis untuk DLAMIs

Di sini Anda dapat menemukan catatan rilis terperinci untuk semua opsi AWS Deep Learning AMIs (DLAMI) yang saat ini didukung.

Untuk catatan rilis untuk DLAMI framework yang tidak lagi kami dukung, lihat bagian Unsupported Framework Release Notes Archive pada halaman Framework DLAMISupport Policy.

> **Note**
>
> AWS Deep Learning AMIs Memiliki irama rilis malam untuk patch keamanan. Kami tidak menyertakan patch keamanan tambahan ini dalam catatan rilis resmi.

Basis DLAMIs

GPU

- X86
  - AWS Basis Pembelajaran Mendalam AMI (Amazon Linux 2)
  - AWS Basis Pembelajaran Mendalam AMI (Ubuntu 22.04)
  - AWS Basis Pembelajaran Mendalam AMI (Ubuntu 20.04)
- ARM64
  - AWS Basis Pembelajaran Mendalam ARM64 AMI (Ubuntu 22.04)
  - AWS Basis Pembelajaran Mendalam ARM64 AMI (Amazon Linux 2)

AWS Neuron

- X86
  - AWS AMINeuron Dasar Pembelajaran Mendalam (Amazon Linux 2)
  - AWS AMINeuron Dasar Pembelajaran Mendalam (Ubuntu 20.04)

Qualcomm

- X86

- AWS Basis Pembelajaran Mendalam Qualcomm AMI (Amazon Linux 2)

Kerangka tunggal DLAMIs

PyTorch-spesifik AMIs

GPU

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

AWS Neuron

- X86
  - AWS AMINeuron Pembelajaran Mendalam PyTorch 1.13 (Amazon Linux 2)
  - AWS AMINeuron Pembelajaran Mendalam PyTorch 1.13 (Ubuntu 20.04)

TensorFlow-spesifik AMIs

GPU

- X86
  - AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.17 (Ubuntu 22.04)
  - AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.16 (Amazon Linux 2)

- AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.16 (Ubuntu 20.04)
- AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.15 (Amazon Linux 2)
- AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.15 (Ubuntu 20.04)

AWS Neuron

- X86
  - AWS AMINeuron Pembelajaran Mendalam TensorFlow 2.10 (Amazon Linux 2)
  - AWS AMINeuron Pembelajaran Mendalam TensorFlow 2.10 (Ubuntu 20.04)

Multi-kerangka DLAMIs

> **Tip**
>
> Jika Anda hanya menggunakan satu kerangka pembelajaran mesin, maka kami merekomendasikan kerangka kerja tunggal DLAMI.

GPU

- X86
  - AWS Pembelajaran Mendalam AMI (Amazon Linux 2)

AWS Neuron

- X86
  - AWS AMINeuron Pembelajaran Mendalam (Amazon Linux 2023)
  - AWS AMINeuron Pembelajaran Mendalam (Ubuntu 22.04)

Fitur usang dari DLAMI

Tabel berikut mencantumkan fitur usang AWS Deep Learning AMIs (DLAMI), tanggal bahwa kami tidak menggunakannya lagi, dan rincian tentang mengapa kami tidak menggunakannya lagi.

Fitur	Tanggal	Detail
Ubuntu 16.04	10/07/2021	Ubuntu Linux 16.04 LTS mencapai akhir LTS jendela lima tahun pada 30 April 2021 dan tidak lagi didukung oleh vendornya. Tidak ada lagi pembaruan ke Deep Learning Base AMI (Ubuntu 16.04) dalam rilis baru per Oktober 2021. Rilis sebelumnya akan terus tersedia.
Amazon Linux	10/07/2021	Amazon Linux adalah end-of-lifepada Desember 2020. Tidak ada lagi pembaruan untuk Deep Learning AMI (Amazon Linux) dalam rilis baru per Oktober 2021. Rilis Deep Learning sebelumny a AMI (Amazon Linux) akan terus tersedia.
Chainer	07/01/2020	Chainer telah mengumumk an akhir rilis utama pada Desember 2019. Akibatnya , kami tidak akan lagi menyertakan lingkunga n Chainer Conda pada DLAMI awal Juli 2020. Rilis sebelumnya dari DLAMI

Fitur	Tanggal	Detail
		yang berisi lingkungan ini akan terus tersedia. Kami akan memberikan pembaruan untuk lingkungan ini hanya jika ada perbaikan keamanan yang diterbitkan oleh komunitas open source untuk kerangka kerja ini.
Python 3.6	06/15/2020	Karena permintaan pelanggan, kami pindah ke Python 3.7 untuk rilis TF/MX/PT baru.
Python 2	01/01/2020	Komunitas open source Python telah secara resmi mengakhiri dukungan untuk Python 2. The TensorFlow, PyTorch, and MXNet community juga telah mengumumkan bahwa rilis TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4, dan MXNet 1.6.0 akan menjadi yang terakhir mendukung Python 2.

Riwayat dokumen untuk DLAMI

Tabel berikut menyediakan riwayat DLAMI rilis terbaru dan perubahan terkait dengan Panduan AWS Deep Learning AMIs Pengembang.

Perubahan terbaru

Perubahan	Deskripsi	Tanggal
ARM64 DLAMI	AWS Deep Learning AMIs Sekarang mendukung gambar berbasis prosesor GPUs Arm64.	29 November 2021
TensorFlow 2	The Deep Learning AMI with Conda sekarang hadir dengan TensorFlow 2 dengan CUDA 10.	3 Desember 2019
AWS Inferensia	Deep Learning AMI sekarang mendukung perangkat keras AWS Inferentia dan AWS NeuronSDK.	3 Desember 2019
Menggunakan TensorFlow Melayani dengan Model Inception	Contoh untuk menggunakan inferensi dengan model Inception ditambahkan untuk TensorFlow Serving, baik dengan maupun tanpa Elastic Inference.	28 November 2018
Menginstal PyTorch dari Nightly Build	Sebuah tutorial telah ditambahkan yang mencakup bagaimana Anda dapat menghapus instalasi PyTorch, lalu menginstal build malam	25 September 2018

	PyTorch pada Deep Learning AMI Anda dengan Conda.	
Conda Tutorial	Contoh MOTD telah diperbarui untuk mencerminkan rilis yang lebih baru.	Juli 23, 2018

Perubahan sebelumnya

Tabel berikut memberikan riwayat DLAMI rilis sebelumnya dan perubahan terkait sebelum Juli 2018.

Perubahan	Deskripsi	Tanggal
TensorFlow dengan Horovod	Ditambahkan tutorial untuk pelatihan ImageNet dengan TensorFlow dan Horovod.	Selasa, 06 Juni 2018
Panduan peningkatan	Menambahkan panduan peningkatan.	15 Mei 2018
Daerah baru dan tutorial 10 menit baru	Wilayah baru ditambahk an: AS Barat (California N.), Amerika Selatan, Kanada (Tengah), UE (London), dan UE (Paris). Juga, rilis pertama dari tutorial 10 menit berjudul: “Memulai dengan Pembelaja ran AMI Mendalam”.	26 April 2018
Tutorial rantai	Tutorial untuk menggunak an Chainer dalam multi-GPU, tunggalGPU, dan CPU mode telah ditambahkan. CUDAintegrasi ditingkatkan dari CUDA 8 menjadi CUDA 9 untuk beberapa kerangka kerja.	28 Februari 2018

Perubahan	Deskripsi	Tanggal
Linux AMIs v3.0, ditambah pengenalan MXNet Model Server, TensorFlow Serving, dan TensorBoard	Menambahkan tutorial untuk Conda AMIs dengan model baru dan kemampuan penyajian visualisasi menggunakan MXNet Model Server v0.1.5, TensorFlow Melayani v1.4.0, dan v0.4.0. TensorBoard AMIdan CUDA kemampuan kerangka kerja yang dijelaskan dalam Conda dan CUDA ikhtisar. Catatan rilis terbaru dipindahkan ke https://aws.amazon.com/releasenotes/	25 Januari 2018
Linux AMIs v2.0	Base, Source, dan Conda AMIs diperbarui dengan NCCL 2.1. Sumber dan Conda AMIs diperbarui dengan MXNet v1.0, PyTorch 0.3.0, dan Keras 2.0.9.	11 Desember 2017
Dua AMI opsi Windows ditambahkan	Windows 2012 R2 dan 2016 AMIs dirilis: ditambahkan ke panduan AMI pemilihan dan ditambahkan ke catatan rilis.	30 November 2017
Rilis dokumentasi awal	Penjelasan rinci tentang perubahan dengan tautan ke topik/bagian yang diubah.	15 November 2017

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.