aws

Panduan Developer

Pembelajaran Mendalam AMI

Pembelajaran Mendalam AMI: Panduan Developer

Table of Contents

Apa itu AWS Deep Learning AMI?

Selamat datang di Panduan Pengguna untuk AWS Deep Learning AMI.

AWS Deep Learning AMI (DLAMI) adalah toko serba ada Anda untuk pembelajaran mendalam di cloud. Gambar mesin yang disesuaikan ini tersedia di sebagian besar wilayah Amazon EC2 untuk berbagai jenis instans, dari instans khusus CPU kecil hingga instans multi-GPU berdaya tinggi terbaru. Muncul dikonfigurasikan sebelumnya dengan NVIDIA CUDA dan NVIDIA cuDNN, serta rilis terbaru dari kerangka pembelajaran mendalam yang paling populer.

Tentang Panduan Ini

Panduan ini akan membantu Anda meluncurkan dan menggunakan DLAMI. Ini mencakup beberapa kasus penggunaan yang umum untuk pembelajaran mendalam, baik untuk pelatihan maupun inferensi. Memilih AMI yang tepat untuk tujuan Anda dan jenis contoh yang mungkin Anda inginkan juga tercakup. DLAMI dilengkapi dengan beberapa tutorial untuk masing-masing kerangka kerja. Ini juga memiliki tutorial tentang pelatihan terdistribusi, debugging, menggunakan AWS Inferentia dan AWS Trainium, dan konsep kunci lainnya. Anda akan menemukan petunjuk tentang cara mengkonfigurasi Jupyter untuk menjalankan tutorial di browser Anda.

Prasyarat

Anda harus terbiasa dengan alat baris perintah dan Python dasar untuk berhasil menjalankan DLAMI. Tutorial tentang cara menggunakan setiap kerangka kerja disediakan oleh kerangka kerja itu sendiri, namun, panduan ini dapat menunjukkan kepada Anda cara mengaktifkan masing-masing kerangka kerja dan menemukan tutorial yang sesuai untuk memulai.

Contoh Penggunaan DLAMI

Belajar tentang pembelajaran mendalam: DLAMI adalah pilihan tepat untuk belajar atau mengajar pembelajaran mesin dan kerangka pembelajaran mendalam. Ini menghilangkan sakit kepala dari pemecahan masalah instalasi setiap kerangka kerja dan membuat mereka bermain bersama di komputer yang sama. DLAMI dilengkapi dengan notebook Jupyter dan memudahkan untuk menjalankan tutorial yang disediakan oleh kerangka kerja untuk orang-orang yang baru mengenal pembelajaran mesin dan pembelajaran mendalam.

Pengembangan aplikasi: Jika Anda seorang pengembang aplikasi dan tertarik untuk menggunakan pembelajaran mendalam untuk membuat aplikasi Anda memanfaatkan kemajuan terbaru dalam AI, 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 tertarik untuk memproses data Anda dengan pembelajaran mendalam, 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 dan ingin mencoba kerangka kerja baru, menguji model baru, atau melatih model baru, DLAMI AWS dan kemampuan untuk skala dapat mengurangi rasa sakit instalasi yang membosankan dan pengelolaan beberapa node pelatihan.

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

Fitur DLAMI

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 NVIDIA CUDA dan dependensi lainnya

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

Ini adalah daftar lengkap kerangka kerja yang didukung oleh Deep Learning AMI dengan Conda:

- PyTorch
- TensorFlow 2

> **Note**
>
> Kami tidak lagi mendukung MXNet, CNTK, Caffe, Caffe2, Theano, Chainer, atau Keras di. AWS Deep Learning AMI

Perangkat Lunak GPU yang sudah diinstal sebelumnya

Bahkan jika Anda menggunakan instance khusus CPU, DLAMI akan memiliki NVIDIA CUDA dan NVIDIA cuDNN. Perangkat lunak yang diinstal sama terlepas dari jenis instancenya. Perlu diingat bahwa alat khusus GPU hanya berfungsi pada instance yang memiliki setidaknya satu GPU. Informasi lebih lanjut tentang ini tercakup dalamMemilih Jenis Instance untuk DLAMI.

Untuk informasi lebih lanjut tentang Instalasi CUDA, lihat. Instalasi 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.

- TensorFlow Melayani

Memulai

Cara Memulai DLAMI

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

Jika Anda baru menggunakan AWS atau menggunakan Amazon EC2, mulailah dengan. Pembelajaran Mendalam AMI dengan Conda Jika Anda terbiasa dengan Amazon EC2 dan AWS layanan lain seperti Amazon EMR, Amazon EFS, atau Amazon S3, dan tertarik untuk mengintegrasikan layanan tersebut untuk proyek yang memerlukan pelatihan atau inferensi terdistribusi, Informasi Terkait periksa untuk melihat apakah sesuai dengan kasus penggunaan Anda.

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

Pilihan lain adalah tutorial cepat ini: Luncurkan AWS Deep Learning AMI (dalam 10 menit).

Langkah Selanjutnya

Memilih DLAMI Anda

Memilih DLAMI Anda

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

- Jenis DLAMI: CUDA versus Basis versus Kerangka Tunggal versus Multi-Kerangka (Conda DLAMI)
- Arsitektur Komputasi: Graviton berbasis x86 versus ARM64 AWS
- Jenis Prosesor: GPU versus CPU versus Inferensia
- SDK: CUDA versus Neuron AWS
- 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 AMI

Untuk nomor versi kerangka kerja tertentu, lihat Catatan Rilis untuk DLAMI

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 `conda` menggunakan lingkungan virtual, mereka hadir baik multi-kerangka kerja atau kerangka kerja tunggal DLAMI. 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 ditawarkan DLAMI. Sebagian besar pengguna menemukan bahwa AMI Pembelajaran Mendalam baru dengan Conda sangat cocok untuk mereka.

Mereka sering diperbarui dengan versi terbaru dari kerangka kerja, dan memiliki driver dan perangkat lunak GPU terbaru. Mereka umumnya disebut sebagai AWS Deep Learning AMI dalam sebagian besar dokumen. DLAMI ini mendukung Ubuntu20.04, sistem Operasi Amazon Linux 2. Dukungan sistem operasi tergantung pada dukungan dari OS upstream.

Kandidat Stabil Versus Rilis

AMI Conda 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 MKL DNN, yang mempercepat pelatihan dan inferensi pada jenis instans CPU C5 dan C4. Binari juga dikompilasi untuk mendukung set instruksi Intel tingkat lanjut termasuk namun tidak terbatas pada AVX, AVX-2, SSE4.1, dan SSE4.2. Ini mempercepat operasi vektor dan floating point pada arsitektur CPU Intel. Selain itu, untuk jenis instans GPU, CUDA dan cuDNN diperbarui dengan versi mana pun yang didukung rilis resmi terbaru.

AMI Pembelajaran Mendalam dengan Conda secara otomatis menginstal versi framework yang paling dioptimalkan untuk instans Amazon EC2 Anda pada aktivasi pertama framework. Untuk informasi lebih lanjut, lihatMenggunakan AMI Pembelajaran Mendalam 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 DLAMI sebelumnya (v26, v25, dll) yang berisi lingkungan Python 2 Conda terus tersedia. Namun, kami menyediakan pembaruan untuk lingkungan Python 2 Conda pada versi DLAMI yang diterbitkan sebelumnya hanya jika ada perbaikan keamanan yang diterbitkan oleh komunitas sumber terbuka untuk versi tersebut. Rilis

DLAMI dengan versi terbaru PyTorch dan kerangka kerja tidak mengandung lingkungan Python 2 Conda. TensorFlow

Dukungan CUDA

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

Selanjutnya

Pilihan Arsitektur DLAMI

Topik Terkait

- Untuk tutorial tentang menggunakan AMI Pembelajaran Mendalam dengan Conda, lihat Menggunakan AMI Pembelajaran Mendalam dengan Conda tutorialnya.

Pilihan Arsitektur DLAMI

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

Untuk informasi tentang memulai dengan DLAMI GPU ARM64, lihat. DLAMI ARM64 Untuk detail selengkapnya tentang jenis instans yang tersedia, lihatMemilih Jenis Instance untuk DLAMI.

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 Instance untuk DLAMI

Memilih Jenis Instance untuk DLAMI

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

- Jika Anda baru mengenal pembelajaran mendalam, maka instance dengan satu GPU mungkin sesuai dengan kebutuhan Anda.
- Jika Anda sadar anggaran, maka Anda dapat menggunakan instance khusus CPU.
- 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 instans GPU berkinerja tinggi dengan arsitektur CPU berbasis ARM64, maka Anda dapat menggunakan jenis instans G5G.
- Jika Anda tertarik untuk menjalankan model terlatih untuk inferensi dan prediksi, Anda dapat melampirkan Amazon Elastic Inference ke instans Amazon EC2 Anda. Amazon Elastic Inference memberi Anda akses ke akselerator dengan sebagian kecil dari GPU.
- Untuk layanan inferensi volume tinggi, satu instance CPU 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 sekelompok GPU. 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 instans, lihat Jenis EC2.

Topik berikut memberikan informasi tentang pertimbangan jenis instance.

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

Topik

Harga untuk DLAMI

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

Beberapa jenis instans Amazon EC2 diberi label gratis. Dimungkinkan untuk menjalankan DLAMI pada salah 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 core CPU, lebih banyak ruang disk, lebih banyak RAM, atau satu atau lebih GPU, maka Anda memerlukan instance yang tidak ada di kelas instans free-tier.

Untuk informasi selengkapnya tentang pemilihan dan harga instans, lihat harga Amazon EC2.

Ketersediaan Wilayah DLAMI

Setiap Wilayah mendukung berbagai jenis instans yang berbeda dan seringkali jenis instans memiliki biaya yang sedikit berbeda di Wilayah yang berbeda. DLAMI tidak tersedia di setiap Wilayah, tetapi dimungkinkan untuk menyalin DLAMI ke Wilayah pilihan Anda. Lihat Menyalin AMI untuk informasi

selengkapnya. 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 .

Selanjutnya

Instans GPU yang Direkomendasikan

Instans GPU yang Direkomendasikan

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

Jenis contoh berikut mendukung DLAMI. Untuk informasi tentang opsi tipe instans GPU dan kegunaannya, lihat Jenis Instans dan pilih Komputasi Akselerasi.

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

- Instans Amazon EC2 P3 memiliki hingga 8 GPU NVIDIA Tesla V100.
- Instans Amazon EC2 P4 memiliki hingga 8 GPU NVIDIA Tesla A100.
- Instans Amazon EC2 P5 memiliki hingga 8 GPU NVIDIA Tesla H100.
- Instans Amazon EC2 G3 memiliki hingga 4 GPU NVIDIA Tesla M60.
- Instans Amazon EC2 G4 memiliki hingga 4 GPU NVIDIA T4.
- Instans Amazon EC2 G5 memiliki hingga 8 GPU NVIDIA A10G.
- Instans Amazon EC2 G6 memiliki hingga 8 GPU NVIDIA L4.
- Instans Amazon EC2 G5G memiliki prosesor Graviton2 berbasis ARM64.AWS

Instans DLAMI menyediakan perkakas untuk memantau dan mengoptimalkan proses GPU Anda. Untuk informasi selengkapnya tentang memantau proses GPU Anda, lihatPemantauan dan Optimasi GPU.

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

Selanjutnya

Instans CPU yang Direkomendasikan

Instans CPU yang Direkomendasikan

Baik Anda memiliki anggaran terbatas, belajar tentang pembelajaran mendalam, atau hanya ingin menjalankan layanan prediksi, Anda memiliki banyak opsi terjangkau dalam kategori CPU. Beberapa kerangka kerja memanfaatkan Intel MKL DNN, yang mempercepat pelatihan dan inferensi pada jenis instans CPU C5 (tidak tersedia di semua Wilayah). Untuk informasi tentang jenis instans CPU, lihat Jenis Instans EC2 dan pilih Compute Optimized.

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

- Instans Amazon EC2 C5 memiliki hingga 72 vCPU Intel. 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 RAM instans yang tersedia, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

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

Untuk informasi lebih lanjut tentang memulai dengan AWS Inferentia DLAMIS, lihat. Chip AWS Inferentia 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 RAM instans yang tersedia, 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.

Kebijakan Support Framework

AWS Deep Learning AMIs (DLAMI) menyederhanakan konfigurasi gambar untuk beban kerja pembelajaran mendalam dan dioptimalkan dengan kerangka kerja, perangkat keras, driver, pustaka, dan sistem operasi terbaru. Halaman ini merinci kebijakan dukungan kerangka kerja untuk DLAMI. Untuk daftar DLAMI yang tersedia, lihat Catatan Rilis untuk DLAMI.

Kerangka Kerja yang Didukung

Referensikan tabel AWS Deep Learning AMI Framework Support Policy berikut untuk memeriksa kerangka kerja dan versi mana yang didukung secara aktif.

Lihat Akhir tambalan untuk memeriksa berapa lama AWS mendukung versi saat ini yang secara aktif didukung oleh tim pemeliharaan kerangka kerja asal. Kerangka kerja dan versi tersedia dalam DLAMI kerangka tunggal, atau DLAMI multi-kerangka kerja.

> **Note**
>
> Dalam versi kerangka x.y.z, x mengacu pada versi utama, y mengacu pada versi minor, dan z mengacu pada versi patch. Misalnya, untuk TensorFlow 2.6.5, versi utama adalah 2, versi minor adalah 6, dan versi patch adalah 5.

Lihat catatan rilis untuk detail lebih lanjut tentang gambar tertentu:

- Catatan rilis DLAMI kerangka tunggal
- Catatan rilis DLAMI multi-kerangka

Pertanyaan yang Sering Diajukan

- Versi kerangka kerja apa yang mendapatkan tambalan keamanan?
- Gambar apa yang AWS dipublikasikan saat versi kerangka kerja baru dirilis?
- Gambar apa yang mendapatkan AWS fitur SageMaker baru/?
- 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?

Versi kerangka kerja apa yang mendapatkan tambalan keamanan?

Jika versi framework diberi label Supported dalam tabel AWS Deep Learning AMI Framework Support Policy, maka akan mendapat patch keamanan.

Gambar apa yang AWS dipublikasikan saat versi kerangka kerja baru dirilis?

Kami menerbitkan DLAMI baru segera setelah versi baru TensorFlow dan dirilis PyTorch. 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 selengkapnya tentang pemeliharaan gambar, lihat Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

Gambar apa yang mendapatkan AWS fitur SageMaker baru/?

Fitur baru biasanya dirilis dalam versi terbaru DLAMI untuk PyTorch dan. TensorFlow Lihat catatan rilis untuk gambar tertentu untuk detail tentang fitur baru SageMaker atau AWS fitur. Untuk daftar DLAMI yang tersedia, lihat Catatan Rilis untuk DLAMI. Untuk informasi selengkapnya 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 dalam tabel AWS Deep Learning AMI Framework Support Policy mengacu pada versi framework terbaru yang AWS tersedia di GitHub. Setiap rilis terbaru mencakup pembaruan untuk driver, perpustakaan, dan paket yang relevan di 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 dalam tabel AWS Deep Learning AMI Framework Support Policy, Anda mungkin tidak memiliki driver, pustaka, dan paket yang relevan yang paling diperbarui. Untuk up-to-date versi lainnya, kami sarankan Anda meningkatkan ke salah satu kerangka kerja yang didukung yang tersedia menggunakan DLAMI terbaru pilihan Anda. Untuk daftar DLAMI yang tersedia, lihat Catatan Rilis untuk DLAMI.

Apakah DLAMI mendukung versi sebelumnya? TensorFlow

Tidak. Kami mendukung versi patch terbaru dari setiap versi utama terbaru framework yang dirilis 365 hari dari GitHub rilis awal seperti yang dinyatakan dalam tabel AWS Deep Learning AMI Framework Support Policy. 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 kerangka kerja terbaru, ambil ID DLAMI dan gunakan untuk meluncurkan DLAMI menggunakan Konsol EC2. Untuk contoh perintah AWS CLI untuk mengambil AWS Deep Learning AMI ID, lihat bagian Deep Learning Frameworks di Katalog. AWS Deep Learning AMI AWS Kueri ID CLI AMI juga disertakan dalam catatan rilis DLAMI kerangka tunggal. Versi framework yang Anda pilih harus diberi label Supported dalam tabel AWS Deep Learning AMI Framework Support Policy.

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 baru CUDA, kami merilis DLAMI baru untuk versi tersebut dengan dukungan untuk versi TensorFlow CUDA 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 instans EC2 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 DLAMI yang tersedia, 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 untuk DLAMI untuk informasi ketergantungan yang diperbarui.

Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

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

- Upgrade versi kerangka kerja (tambalan)
- AWSpatch keamanan
- Akhir tanggal tambalan (Aging out)
- Ketergantungan end-of-support

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

Upgrade versi kerangka kerja (tambalan)

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

DLAMI baru yang dibuat karena peningkatan tambalan ditunjuk dengan ID AMI baru.

AWSpatch keamanan

Jika Anda memiliki beban kerja berdasarkan gambar dengan TensorFlow 2.7.0 dan AWS membuat patch keamanan, maka versi baru DLAMI dirilis untuk 2.7.0. TensorFlow 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 DLAMI terbaru, lihat Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?

DLAMI baru yang dibuat karena peningkatan tambalan ditunjuk dengan ID AMI baru.

Akhir tanggal tambalan (Aging out)

DLAMI mencapai akhir tanggal patch mereka 365 hari setelah tanggal GitHub rilis.

Untuk DLAMI multi-kerangka kerja, ketika salah satu versi kerangka diperbarui, DLAMI baru dengan versi yang diperbarui diperlukan. DLAMI dengan versi kerangka kerja 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 DLAMI TensorFlow 2.7.0 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 gambar DLAMI 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 DLAMI dengan versi kerangka kerja yang lebih lama, ambil ID DLAMI dan gunakan untuk meluncurkan DLAMI menggunakan Konsol EC2. Untuk perintah AWS CLI untuk mengambil ID AMI, lihat bagian Deep Learning Frameworks di Katalog AMI AWSPembelajaran Mendalam. AWS Kueri ID CLI AMI juga disertakan dalam catatan rilis DLAMI kerangka tunggal.

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

Tetap up-to-date dengan kerangka kerja dan versi DLAMI menggunakan tabel Kebijakan Dukungan AWS Deep Learning AMI Kerangka Kerja, catatan rilis DLAMI.

Apakah saya memerlukan lisensi komersial untuk menggunakan Repositori Anaconda?

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

Meluncurkan dan Mengkonfigurasi DLAMI

Jika Anda di sini, Anda seharusnya sudah memiliki ide bagus tentang AMI mana yang ingin Anda luncurkan. Jika tidak, temukan DLAMI dan perangkat keras terkait, kerangka kerja, dan pengambilan ID di file. Catatan Rilis untuk DLAMI

Anda juga harus tahu jenis dan wilayah instance mana yang akan Anda pilih. Jika tidak, jelajahiMemilih Jenis Instance untuk DLAMI.

> **Note**
>
> Kita akan menggunakan p3.16xlarge sebagai tipe instance default dalam contoh. Ganti saja ini dengan jenis instance apa pun yang ada dalam pikiran Anda.

> **Important**
>
> Jika Anda berencana untuk menggunakan Elastic Inference, Anda memiliki Elastic Inference Setup yang harus diselesaikan sebelum meluncurkan DLAMI Anda.

Topik

Langkah 1: Luncurkan DLAMI

> **Note**
>
> Untuk panduan ini, kami mungkin membuat referensi khusus untuk Deep Learning AMI (Ubuntu 18.04). Bahkan jika Anda memilih DLAMI yang berbeda, Anda harus dapat mengikuti panduan ini.

1. Temukan ID DLAMI Anda
2. Luncurkan instans Amazon EC2 dari DLAMI

   Anda akan menggunakan Konsol Amazon EC2. Ikuti instruksi yang dirinci di Peluncuran dari Konsol Amazon EC2

   > **Tip**
   >
   > Opsi CLI: Jika Anda memilih untuk memutar DLAMI menggunakan AWS CLI, Anda akan memerlukan ID AMI, wilayah dan jenis instance, dan informasi token keamanan Anda. Pastikan AMI dan ID instans Anda sudah siap. Jika Anda belum mengatur AWS CLI, lakukan itu terlebih dahulu menggunakan panduan untuk Menginstal Antarmuka Baris AWS Perintah.

3. Setelah Anda menyelesaikan langkah-langkah dari salah satu opsi tersebut, tunggu hingga instans siap. Ini biasanya hanya memakan waktu beberapa menit. Anda dapat memverifikasi status instans di Konsol EC2.

Ambil ID DLAMI

Setiap AMI memiliki pengenal unik (ID). Anda dapat menanyakan ID untuk DLAMI pilihan Anda dengan AWS Command Line Interface (AWS CLI). Jika Anda belum AWS CLI menginstal, lihat Memulai dengan AWS CLI.

> **Note**
>
> Pengingat: Anda dapat menemukan semua DLAMI dan prosesor/akselorator terkait, sistem operasi, arsitektur komputasi, keluarga instans Amazon EC2 yang direkomendasikan, status

dukungan, dan kueri pengambilan ID, di Katalog.AWS Deep Learning AMI Lihat juga catatan Catatan Rilis untuk DLAMI rilis DLAMI untuk informasi tambahan (driver, versi python, jenis Amazon EBS).

1. Pastikan AWS kredensil Anda dikonfigurasi.

```
aws configure
```

2. Gunakan perintah berikut untuk mengambil ID DLAMI Anda atau menemukan kueri yang disediakan dalam Katalog. AWS Deep Learning AMI

```
aws ec2 describe-images --region us-east-1 --owners amazon \
--filters 'Name=name,Values=Deep Learning AMI (Ubuntu 18.04) Version ??.?'
 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' --output text
```

> **Note**
>
> Anda dapat menentukan versi rilis untuk kerangka kerja tertentu atau mendapatkan rilis terbaru dengan mengganti nomor versi dengan tanda tanya.

3. Outputnya akan serupa dengan yang berikut ini:

```
ami-094c089c38ed069f2
```

Salin ID DLAMI ini dan q tekan untuk keluar dari prompt.

Langkah Selanjutnya

Peluncuran dari Konsol Amazon EC2

Peluncuran dari Konsol Amazon EC2

> **Note**
>
> Untuk meluncurkan instance dengan Elastic Fabric Adapter (EFA), lihat langkah-langkah berikut.

1. Buka Konsol EC2.
2. Perhatikan wilayah Anda saat ini di navigasi paling atas. Jika ini bukan yang Anda inginkan Wilayah AWS, ubah opsi ini sebelum melanjutkan. Untuk informasi lebih lanjut, lihat Wilayah EC2.
3. Pilih Luncurkan Instans.
4. Masukkan nama untuk instans Anda dan pilih DLAMI yang tepat untuk Anda.
   a. Temukan DLAMI yang ada di AMI Saya atau pilih Mulai Cepat.
   b. Cari berdasarkan ID DLAMI. Jelajahi opsi lalu pilih pilihan Anda.
5. Pilih jenis instance. Anda dapat menemukan keluarga instans yang direkomendasikan untuk DLAMI Anda di Katalog. AWS Deep Learning AMI Untuk rekomendasi umum tentang jenis instans DLAMI, lihat Pemilihan Instance.

   > **Note**
   >
   > Jika Anda ingin menggunakan Elastic Inference (EI), klik Configure Instance Details, pilih Add an Amazon EI accelerator, lalu pilih ukuran akselerator Amazon EI.

6. Pilih Luncurkan Instans.

> **Tip**
>
> Lihat Memulai Pembelajaran Mendalam Menggunakan AMI AWS Pembelajaran Mendalam untuk panduan dengan tangkapan layar.

Langkah Selanjutnya

Langkah 2: Connect ke DLAMI

Langkah 2: Connect ke DLAMI

Connect ke DLAMI yang Anda luncurkan dari klien (Windows, macOS, atau Linux). Untuk informasi selengkapnya, lihat Connect to Your Linux Instance di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Simpan salinan perintah login SSH jika Anda ingin melakukan pengaturan Jupyter setelah masuk. Anda akan menggunakan variasinya untuk terhubung ke halaman web Jupyter.

Langkah Selanjutnya

Langkah 3: Uji DLAMI Anda

Langkah 3: Uji DLAMI Anda

Tergantung pada versi DLAMI Anda, Anda memiliki opsi pengujian yang berbeda:

- Pembelajaran Mendalam AMI dengan Conda- pergi keMenggunakan AMI Pembelajaran Mendalam dengan Conda.
- Deep Learning Base AMI— lihat dokumentasi instalasi kerangka kerja yang Anda inginkan.

Anda juga dapat membuat notebook Jupyter, mencoba tutorial, atau mulai coding dengan Python. Untuk informasi selengkapnya, lihat Mengatur Server Notebook Jupyter.

Langkah 4: Kelola Instans DLAMI Anda

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 DLAMI Anda, Anda akan 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

Pembersihan

Ketika Anda tidak lagi membutuhkan DLAMI, Anda dapat menghentikannya atau menghentikannya untuk menghindari biaya berkelanjutan. Menghentikan sebuah instance akan menyimpannya sehingga Anda dapat melanjutkannya nanti. Konfigurasi, file, dan informasi non-volatile lainnya disimpan dalam volume di Amazon S3. Anda akan dikenakan biaya S3 kecil untuk mempertahankan volume saat instans dihentikan, tetapi Anda tidak akan lagi dikenakan biaya untuk sumber daya komputasi saat berada dalam keadaan berhenti. Ketika Anda memulai instance lagi, itu akan memasang volume itu dan data Anda akan ada di sana. Jika Anda menghentikan sebuah instance, itu hilang, dan Anda tidak dapat memulainya lagi. Data Anda sebenarnya masih berada di S3, jadi untuk mencegah biaya lebih lanjut Anda perlu menghapus volume juga. Untuk petunjuk selengkapnya, lihat Menghentikan Instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Mengatur Server Notebook Jupyter

Sebuah server notebook Jupyter memungkinkan Anda untuk membuat dan menjalankan notebook Jupyter dari contoh DLAMI Anda. Dengan notebook Jupyter, Anda dapat melakukan eksperimen machine learning (ML) untuk pelatihan dan inferensi saat menggunakanAWS infrastruktur dan mengakses paket yang dibangun ke dalam DLAMI. Untuk informasi selengkapnya tentang notebook Jupyter, lihat dokumentasi Notebook Jupyter.

Untuk mengatur server notebook Jupyter, Anda harus:

- Konfigurasikan server notebook Jupyter pada instans Amazon EC2 DLAMI Anda.
- Konfigurasikan klien Anda sehingga Anda dapat terhubung ke server notebook Jupyter. Kami menyediakan petunjuk konfigurasi untuk klien Windows, macOS, dan Linux.
- Uji pengaturan dengan masuk ke server notebook Jupyter.

Untuk menyelesaikan langkah-langkah menyiapkan Jupyter, ikuti petunjuk dalam topik berikut. Setelah Anda menyiapkan server notebook Jupyter, lihatMenjalankan Tutorial Notebook Jupyter informasi tentang menjalankan contoh notebook yang dikirimkan di DLAMI.

Topik

- Amankan Server Jupyter Anda
- Mulai server notebook Jupyter

- Mengkonfigurasi klien untuk Connect ke Server Jupyter
- Uji dengan Masuk ke server notebook Jupyter

Amankan Server Jupyter Anda

Di sini kita mengatur Jupyter dengan SSL dan kata sandi khusus.

Connect ke instans Amazon EC2, dan kemudian selesaikan prosedur berikut.

Konfigurasikan server Jupyter

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

   ```
   $ jupyter notebook password
   ```

   Output akan terlihat seperti ini:

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

2. Membuat sertifikat SSL yang ditandatangani sendiri. Ikuti petunjuk untuk mengisi lokalitas Anda sesuai kebutuhan Anda. Anda harus memasukkan . jika Anda ingin meninggalkan 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 sertifikat SSL reguler yang ditandatangani pihak ketiga dan tidak menyebabkan browser memberi Anda peringatan keamanan. Proses ini jauh lebih terlibat. Kunjungi dokumentasi Jupyter untuk informasi lebih lanjut.

Langkah Selanjutnya

Mulai server notebook Jupyter

Mulai server notebook Jupyter

Sekarang Anda dapat menjalankan server Jupyter dengan masuk ke instance dan menjalankan perintah berikut yang menggunakan sertifikat SSL yang Anda buat di langkah sebelumnya.

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

Dengan server dimulai, Anda sekarang dapat terhubung ke melalui terowongan SSH dari komputer klien Anda. Ketika server berjalan, Anda akan melihat beberapa output dari Jupyter mengkonfirmasikan bahwa server sedang berjalan. Pada titik ini, abaikan keterangan bahwa Anda dapat mengakses server melalui URL localhost, karena itu tidak akan berfungsi sampai Anda membuat tunnel.

> **Note**
>
> Jupyter akan menangani lingkungan switching untuk Anda ketika Anda beralih kerangka kerja menggunakan antarmuka web Jupyter. Info lebih lanjut tentang ini dapat ditemukan diBeralih Lingkungan dengan Jupyter.

Langkah Selanjutnya

Mengkonfigurasi klien untuk Connect ke Server Jupyter

Mengkonfigurasi klien untuk Connect ke Server Jupyter

Setelah mengkonfigurasi klien Anda untuk terhubung ke server notebook Jupyter, Anda dapat membuat dan mengakses notebook di server di ruang kerja Anda dan menjalankan kode pembelajaran mendalam Anda di server.

Pilih informasi konfigurasi, memilih salah satu tautan berikut.

Topik

- Konfigurasi Klien Windows
- Konfigurasi Linux atau macOS Client

Konfigurasi Klien Windows

Mempersiapkan

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

- Nama DNS publik instans Amazon EC2 Anda. Anda dapat menemukan nama DNS publik di konsol EC2.
- key pair untuk file kunci privat. Untuk informasi selengkapnya tentang mengakses key pair Anda, lihat Pasangan kunci Amazon EC2 dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

Menggunakan Notebook Jupyter dari Klien Windows

Lihat panduan ini tentang menghubungkan ke instans Amazon EC2 dari klien Windows.

1. Pemecahan Masalah Menghubungkan ke Instans Anda
2. Menghubungkan ke Instans Linux Anda dari Windows Menggunakan PuTTY

Untuk membuat tunnel ke server Jupyter yang berjalan, pendekatan yang disarankan adalah menginstal Git Bash pada klien Windows Anda, lalu ikuti instruksi klien Linux/macOS. Namun, Anda dapat menggunakan pendekatan apa pun yang Anda inginkan untuk membuka terowongan SSH dengan pemetaan port. Lihat dokumentasi Jupyter untuk informasi lebih lanjut.

Langkah Selanjutnya

Konfigurasi Linux atau macOS Client

Konfigurasi Linux atau macOS Client

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 instans Amazon EC2 dan nama DNS publik instans Amazon EC2 Anda. Catatan, untuk Amazon Linux AMI, nama penggunanya `ec2-user` bukan `ubuntu`.

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

Perintah ini membuka sebuah tunnel antara klien Anda dan instans Amazon EC2 yang menjalankan server notebook Jupyter.

Langkah Selanjutnya

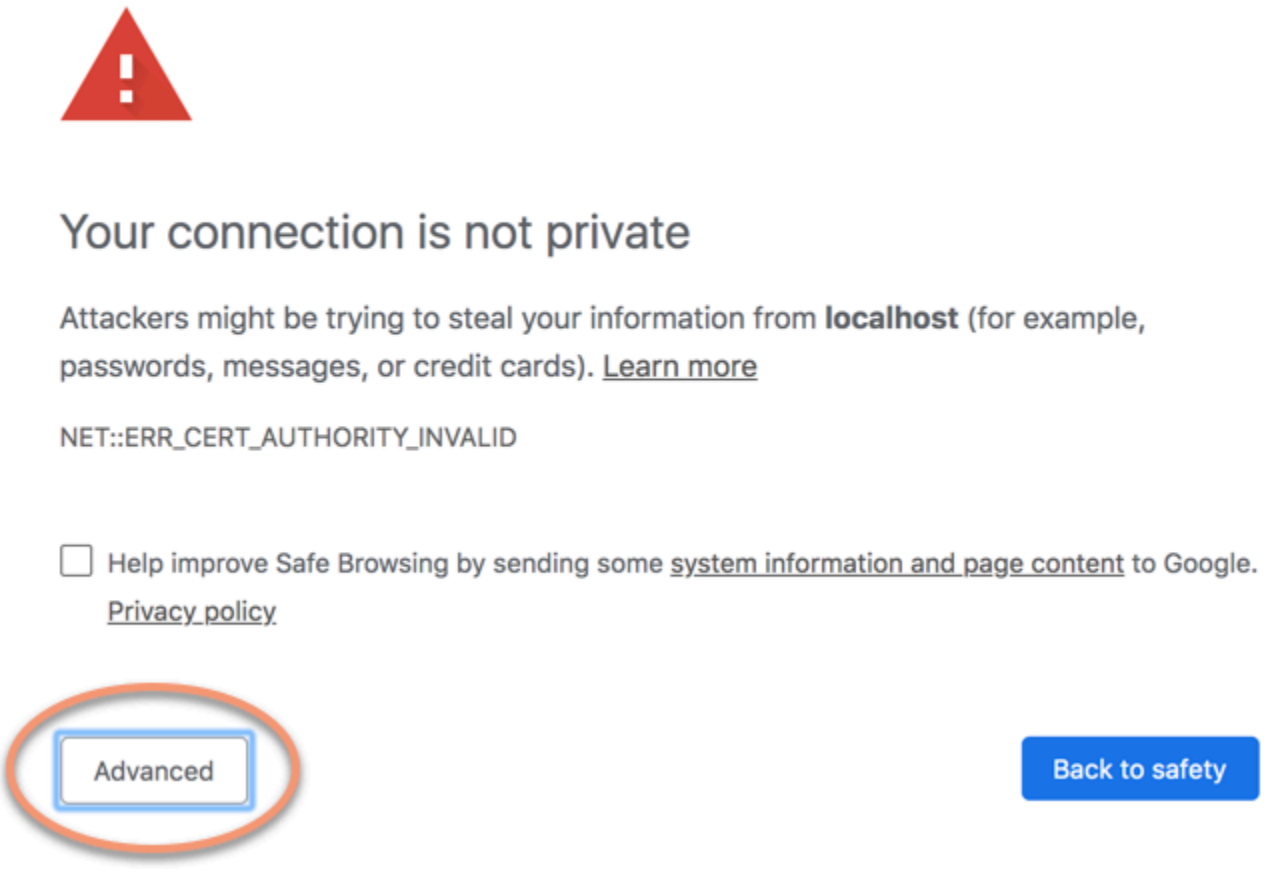
[Uji dengan Masuk ke server notebook Jupyter]

Uji dengan Masuk ke server notebook Jupyter

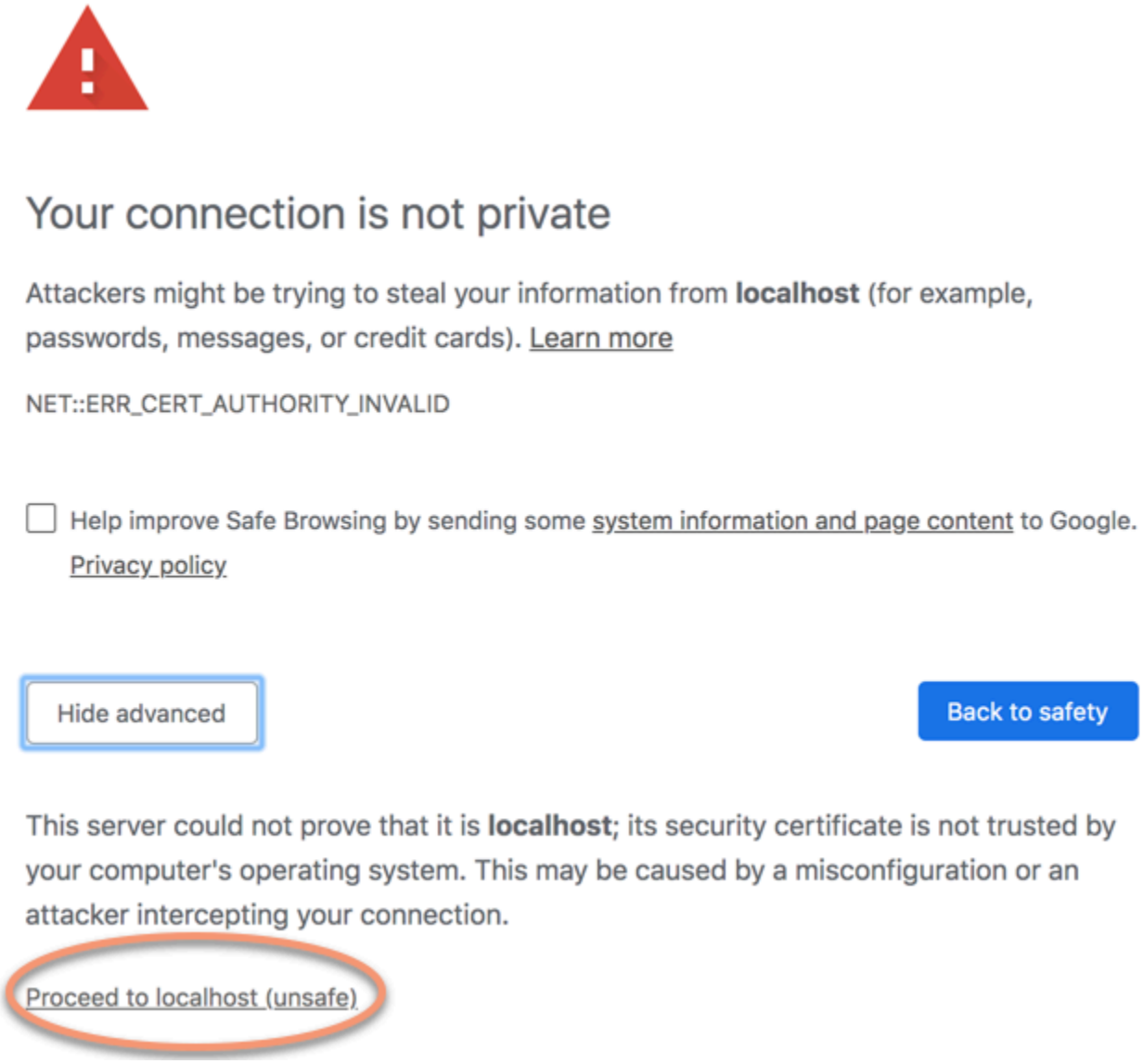
Sekarang Anda siap untuk masuk ke server notebook Jupyter.

Langkah selanjutnya adalah menguji koneksi ke server melalui browser Anda.

1. Di bilah alamat browser Anda, ketik URL berikut, atau klik tautan ini: https://localhost:8888
2. Dengan sertifikat SSL 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. Tergantung browser Anda, Anda akan mendapatkan “lanjutan”, “tampilkan detail”, atau tombol serupa.



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

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

Menggunakan DLAMI

Topik

Bagian berikut menjelaskan bagaimana AMI Pembelajaran Mendalam dengan Conda dapat digunakan untuk beralih lingkungan, menjalankan kode sampel dari masing-masing kerangka kerja, dan menjalankan Jupyter sehingga Anda dapat mencoba berbagai tutorial notebook.

Menggunakan AMI Pembelajaran Mendalam dengan Conda

Topik

Pengantar AMI Pembelajaran Mendalam 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.

AMI Pembelajaran Mendalam 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 DLAMI Anda

Setelah Anda masuk ke server Anda, Anda akan melihat server “message of the day” (MOTD) yang menjelaskan berbagai perintah Conda yang dapat Anda gunakan untuk beralih di antara kerangka kerja pembelajaran mendalam yang berbeda. Di bawah ini adalah contoh MOTD. MOTD spesifik Anda dapat bervariasi saat versi baru DLAMI 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. AMI Pembelajaran Mendalam dengan Conda secara otomatis menginstal versi kerangka kerja

yang paling dioptimalkan untuk instans EC2 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 terminal IPython.

```
(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 berada di konsol IPython, `quit()` gunakan, 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 terminal IPython.

```
(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 pada DLAMI, 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 hadir dengan platform dasar driver GPU dan pustaka akselerasi untuk menerapkan lingkungan pembelajaran mendalam Anda sendiri yang disesuaikan. Secara default AMI dikonfigurasi

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

Mengkonfigurasi Versi CUDA

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

```
nvcc --version
```

Anda dapat memilih dan memverifikasi versi CUDA 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 rilis DLAMI Dasar.

Menjalankan Tutorial Notebook Jupyter

Tutorial dan contoh dikirimkan dengan masing-masing sumber proyek pembelajaran mendalam dan dalam banyak kasus mereka akan berjalan pada DLAMI apa pun. 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 pada DLAMI, Anda harus melakukannya. Mengatur Server Notebook Jupyter

Setelah server Jupyter berjalan, Anda dapat menjalankan tutorial melalui browser web Anda. Jika Anda menjalankan AMI Pembelajaran Mendalam 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 AMI Pembelajaran Mendalam dengan Conda.

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

> **Tip**
>
> Tutorial dan contoh pembelajaran mendalam sering bergantung pada satu atau lebih GPU. Jika tipe instans Anda tidak memiliki GPU, 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 dengan 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 DLAMI Anda saat ini. Klik pada nama kerangka kerja untuk melihat tutorial yang terdaftar, lalu klik tutorial untuk meluncurkannya.

Pertama kali Anda menjalankan notebook pada AMI Pembelajaran Mendalam dengan Conda, ia 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 CUDA mana yang aktif, salah satu cara untuk melihatnya adalah di MOTD saat Anda pertama kali masuk ke DLAMI.

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 AMI Pembelajaran Mendalam dengan perangkat lunak Conda.

Topik

10 Menit Tutorial

- Luncurkan AWS Deep Learning AMI (dalam 10 menit)
- Latih model Deep Learning dengan DLC di Amazon EC2 (dalam 10 menit)

Mengaktifkan Kerangka Kerja

Berikut ini adalah kerangka kerja pembelajaran mendalam yang diinstal pada AMI Pembelajaran Mendalam dengan Conda. Klik pada kerangka kerja untuk mempelajari cara mengaktifkannya.

Topik

- PyTorch
- TensorFlow 2

PyTorch

Mengaktifkan PyTorch

Ketika paket Conda yang stabil dari kerangka kerja dirilis, itu diuji dan diinstal sebelumnya pada 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 pada AMI Pembelajaran Mendalam Anda dengan Conda.

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

```
$ source activate pytorch_p310
```

Mulai terminal IPython.

```
(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 AMI Pembelajaran Mendalam Anda 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 instance GPU) - Instal build malam terbaru dengan CUDA.0: PyTorch

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

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

   ```
   (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 terminal IPython dan periksa versinya. 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 contoh MNIST, Anda dapat menjalankan skrip pengujian dari PyTorch repositori contoh:

```
(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 AMI Pembelajaran Mendalam dengan Conda (DLAMI di Conda) dan menjalankan program 2. TensorFlow

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

Mengaktifkan 2 TensorFlow

Untuk menjalankan TensorFlow DLAMI dengan Conda

1. Untuk mengaktifkan TensorFlow 2, buka instance Amazon Elastic Compute Cloud (Amazon EC2) dari 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 terminal IPython:

```
(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 API TensorFlow Python atau lihat situs webnya. TensorFlow

Pelatihan terdistribusi menggunakan Adaptor Kain Elastis

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

Topik berikut menunjukkan cara memulai menggunakan EFA dengan DLAMI.

> **Note**
>
> Pilih DLAMI Anda dari daftar DLAMI GPU Dasar ini

Topik

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

Meluncurkan AWS Deep Learning AMI Instance Dengan EFA

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

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

Catatan:

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

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

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

Topik

- Mempersiapkan Grup Keamanan Berkemampuan EFA
- Luncurkan Instance Anda
- Verifikasi Lampiran EFA

Mempersiapkan Grup Keamanan Berkemampuan EFA

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

1. Buka konsol Amazon EC2 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 untuk tujuan peluncuran instans Anda yang didukung EFA.
   - 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 instans DLAMI Anda.

Luncurkan Instance Anda

EFA pada saat AWS Deep Learning AMI 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 instance DLAMI yang diaktifkan EFA. Untuk informasi selengkapnya tentang meluncurkan instans berkemampuan EFA, lihat Meluncurkan Instans Berkemampuan EFA ke dalam Grup Penempatan Cluster.

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Luncurkan Instans.
3. Pada halaman Pilih AMI, pilih DLAMI yang didukung yang ditemukan di Halaman Catatan Rilis DLAMI
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 instans yang didukung: Memulai EFA dan MPI
5. Pada halaman Konfigurasi Detail Instans, lakukan langkah berikut:
   - Untuk Jumlah instans, masukkan jumlah instans yang diaktifkan EFA yang ingin Anda luncurkan.
   - Untuk Jaringan dan Subnet, pilih VPC dan subnet sebagai tujuan peluncuran instans.
   - [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. Secara opsional, Anda dapat menentukan alamat IPv4 primer dan satu atau lebih alamat IPv4 sekunder. Jika Anda meluncurkan instans ke subnet yang memiliki blok CIDR IPv6 terkait, Anda dapat secara opsional menentukan alamat IPv6 primer dan satu atau lebih alamat IPv6 sekunder.
- Pilih Berikutnya: Tambahkan Penyimpanan.

6. Di halaman Tambahkan Penyimpanan, tentukan volume yang akan dilampirkan ke instans selain volume yang ditentukan oleh AMI (seperti volume perangkat root), lalu pilih Selanjutnya: Tambahkan Tanda.
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 Lampiran EFA

Dari Konsol

Setelah meluncurkan instance, periksa detail instance di AWS Console. Untuk melakukan ini, pilih instance di konsol EC2 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 EFA tidak diaktifkan, Anda dapat memperbaikinya dengan:

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

1. Di Konsol EC2, buka Antarmuka Jaringan.
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 Instans EC2 dan pilih instans Anda.
6. Buka Actions: Instance State dan hentikan instance sebelum Anda melampirkan EFA.
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 DLAMI. 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 DLAMI. 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 AMI

Menjalankan Aplikasi Multi-Node dengan EFA

Untuk menjalankan aplikasi di seluruh cluster node konfigurasi berikut diperlukan

Topik

- Aktifkan SSH Tanpa Kata Sandi
- Buat File Host
- Tes NCCL

Aktifkan SSH Tanpa Kata Sandi

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

1. Pada node pemimpin, hasilkan keypair RSA.

```
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
```

Tes NCCL

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

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

Instans yang Didukung: P3dn, P4, P5

Tes Fungsionalitas

Tes Multi Node Transfer Pesan NCCL

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

Tes nccl_message_transfer berikut adalah untuk versi CUDA xx.x generik. Anda dapat menjalankan perintah untuk versi CUDA yang tersedia di instans Amazon EC2 Anda dengan mengganti versi CUDA 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 bahwa EFA sedang digunakan sebagai penyedia OFI.

```
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 Kinerja NCCL Multi-node pada P4D.24xLarge

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

Saat membuat skrip Anda sendiri, lihat panduan berikut:

- Gunakan jalur lengkap ke mpirun seperti yang ditunjukkan pada contoh saat menjalankan aplikasi NCCL dengan EFA.
- Ubah params np dan N berdasarkan jumlah instance dan GPU di cluster Anda.
- Tambahkan flag NCCL_DEBUG=INFO dan pastikan bahwa log menunjukkan penggunaan EFA sebagai “Penyedia Terpilih adalah EFA”.
- 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 penggunaan GPU. `watch nvidia-smi`Perintah berikut adalah untuk versi CUDA xx.x generik dan

bergantung pada Sistem Operasi instance Anda. Anda dapat menjalankan perintah untuk versi CUDA yang tersedia di instans Amazon EC2 Anda dengan mengganti versi CUDA dalam skrip.

- Amazon Linux 2:

```
 $ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO -x --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 -x NCCL_PROTO=simple -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 -x --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 -x NCCL_PROTO=simple-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 tes EFA mengembalikan hasil yang valid, silakan gunakan tes berikut untuk mengonfirmasi:

- Dapatkan jenis instans menggunakan Metadata Instans EC2:

```
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}"
```

Pemantauan dan Optimasi GPU

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

- Pemantauan

Pemantauan

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

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

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

Monitor GPU dengan CloudWatch

Saat Anda menggunakan DLAMI dengan 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 metrik GPU 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 DLAMI Anda dengan agen CloudWatch terpadu untuk mengonfigurasi metrik GPU dan memantau pemanfaatan proses bersama GPU di instans akselerasi Amazon EC2.

Ada empat cara untuk mengonfigurasi metrik GPU dengan DLAMI Anda:

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

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

Prasyarat

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

Konfigurasikan metrik GPU minimal

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

- `utilization_gpu`
- `utilization_memory`

Anda dapat menemukan `systemd` layanan untuk metrik GPU 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 metrik GPU sebagian

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

- `utilization_gpu`
- `utilization_memory`
- `memory_total`
- `memory_used`
- `memory_free`

Anda dapat menemukan `systemd` layanan untuk metrik GPU 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 metrik GPU yang tersedia

Konfigurasikan semua metrik GPU yang tersedia menggunakan layanan. `dlami-cloudwatch-agent@all` `systemd` 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 metrik GPU yang telah dikonfigurasi sebelumnya 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 metrik GPU 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

DLAMI yang baru dirilis dikonfigurasi dengan patch keamanan AWS CloudWatch agen terbaru yang tersedia. Lihat bagian berikut untuk memperbarui DLAMI Anda saat ini 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 2 DLAMI.

```
sudo yum update
```

Ubuntu

Untuk mendapatkan patch AWS CloudWatch keamanan terbaru untuk DLAMI dengan Ubuntu, Anda perlu menginstal ulang agen AWS CloudWatch 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 bernama gpumon.py sudah diinstal pada DLAMI Anda. Ini terintegrasi dengan CloudWatch dan mendukung pemantauan penggunaan per GPU: memori GPU, suhu GPU, dan

Daya GPU. Script secara berkala mengirimkan data yang dipantau ke CloudWatch. Anda dapat mengonfigurasi tingkat perincian 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 pemantauan GPU dengan CloudWatch

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

   Kebijakan IAM 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 pengguna IAM dan menambahkan kebijakan untuk CloudWatch, lihat CloudWatch dokumentasi.

2. Pada DLAMI Anda, AWS jalankan configure dan tentukan kredenal pengguna IAM.

   ```
   $ 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 TIDAK 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 umum Python 3 DLAMI.

```
$ source activate python3
```

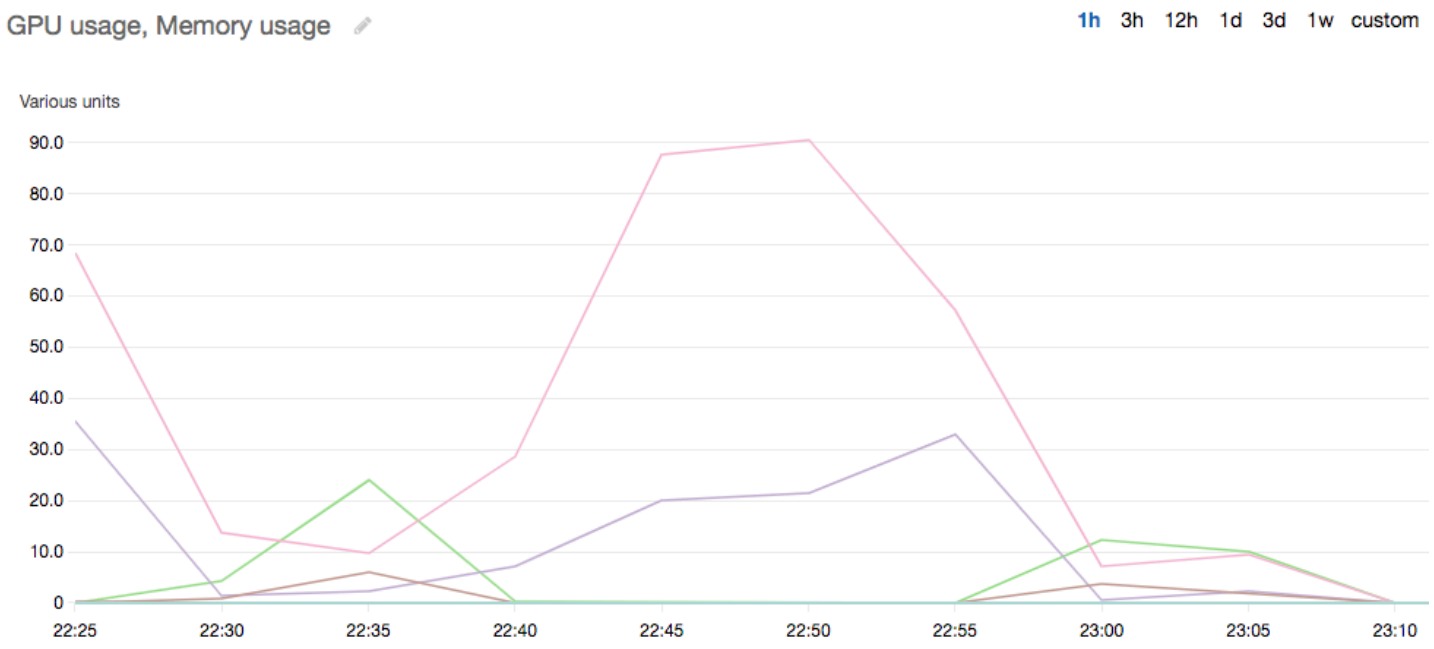
5. Jalankan utilitas gpumon di latar belakang.

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

6. Buka browser Anda ke https://console.aws.amazon.com/cloudwatch/ lalu pilih metrik. 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.



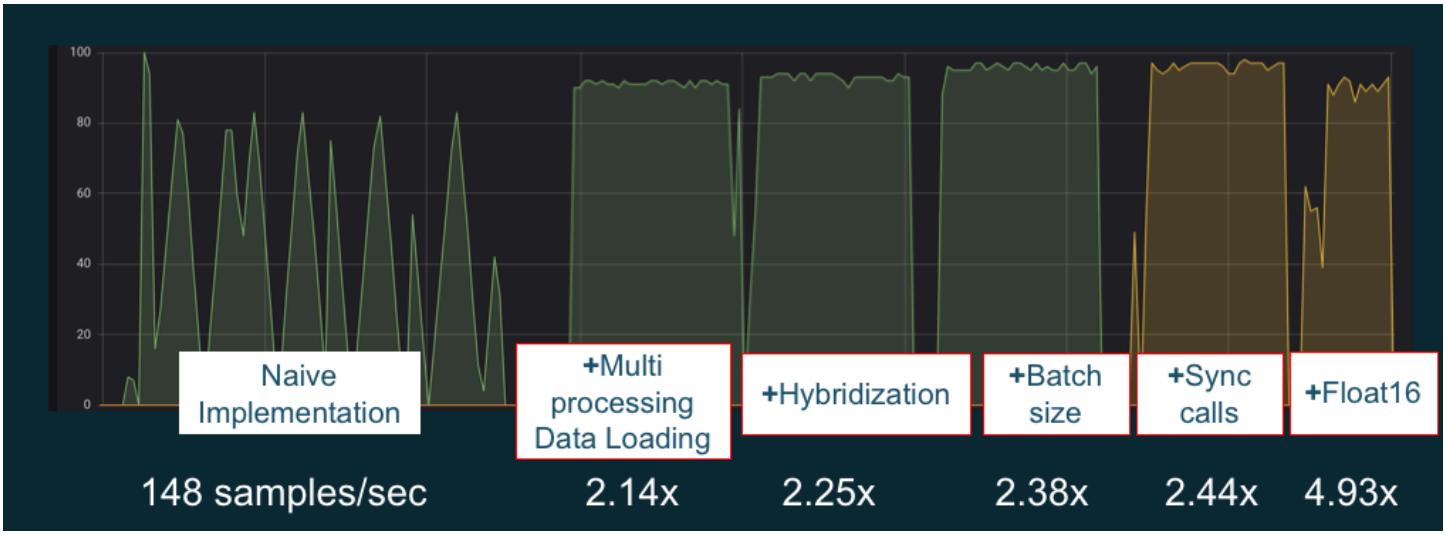
Anda mungkin tertarik dengan topik lain tentang pemantauan dan pengoptimalan GPU ini:

- Pemantauan
  - Monitor GPU dengan CloudWatch
- Pengoptimalan
  - Pemrosesan awal
  - Pelatihan

Pengoptimalan

Untuk memaksimalkan GPU Anda, 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 GPU secara tidak konsisten dan tidak secara maksimal. Saat Anda mengoptimalkan preprocessing dan pemuatan data, Anda dapat mengurangi hambatan dari CPU ke GPU Anda. 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 optimasi kinerja GPU. Sumber bagan: Trik Kinerja dengan MxNet Gluon

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

Topik

- Pemrosesan awal
- Pelatihan

Pemrosesan awal

Preprocessing data melalui transformasi atau augmentasi seringkali dapat menjadi proses yang terikat CPU, 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.

- Perpustakaan Augmentasi Data NVIDIA (DALI): DALI membongkar augmentasi data ke GPU. Ini tidak diinstal sebelumnya pada DLAMI, tetapi Anda dapat mengaksesnya dengan menginstalnya atau memuat wadah kerangka kerja yang didukung pada DLAMI Anda atau instans Amazon Elastic Compute Cloud lainnya. Lihat halaman proyek DALI di situs web NVIDIA untuk detailnya. Untuk contoh kasus penggunaan dan untuk mengunduh contoh kode, lihat contoh Kinerja Pelatihan SageMaker Preprocessing.
- NVJPEG: perpustakaan dekoder JPEG yang dipercepat GPU untuk pemrogram C. Ini mendukung decoding gambar tunggal atau batch serta operasi transformasi berikutnya yang umum dalam

pembelajaran mendalam. nvJPEG dilengkapi built-in dengan DALI, atau Anda dapat mengunduh dari halaman nvjpeg situs web NVIDIA dan menggunakannya secara terpisah.

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

- Pemantauan
  - Monitor GPU dengan CloudWatch
- Pengoptimalan
  - Pemrosesan awal
  - 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.

- NVIDIA Deep Learning SDK - dokumen di situs web NVIDIA yang menjelaskan implementasi presisi campuran untuk MXNet,, 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 situs blog NVIDIA.
> - Pelatihan presisi campuran menggunakan float16 dengan MXNet - artikel FAQ di situs web MXNet.
> - NVIDIA Apex: alat untuk pelatihan presisi campuran yang mudah dengan PyTorch - artikel blog di situs web NVIDIA.

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

- Pemantauan
  - Monitor GPU dengan CloudWatch
- Pengoptimalan
  - Pemrosesan awal
  - Pelatihan

Chip AWS Inferentia 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 AMI (DLAMI).

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

Daftar Isi

Meluncurkan Instance DLAMI dengan Neuron AWS

DLAMI terbaru siap digunakan AWS dengan Inferentia dan dilengkapi dengan AWS paket Neuron API. Untuk meluncurkan instans DLAMI, lihat Meluncurkan dan Mengonfigurasi DLAMI. Setelah Anda memiliki DLAMI, gunakan langkah-langkah di sini untuk memastikan bahwa chip Inferentia AWS dan sumber daya Neuron AWS 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:

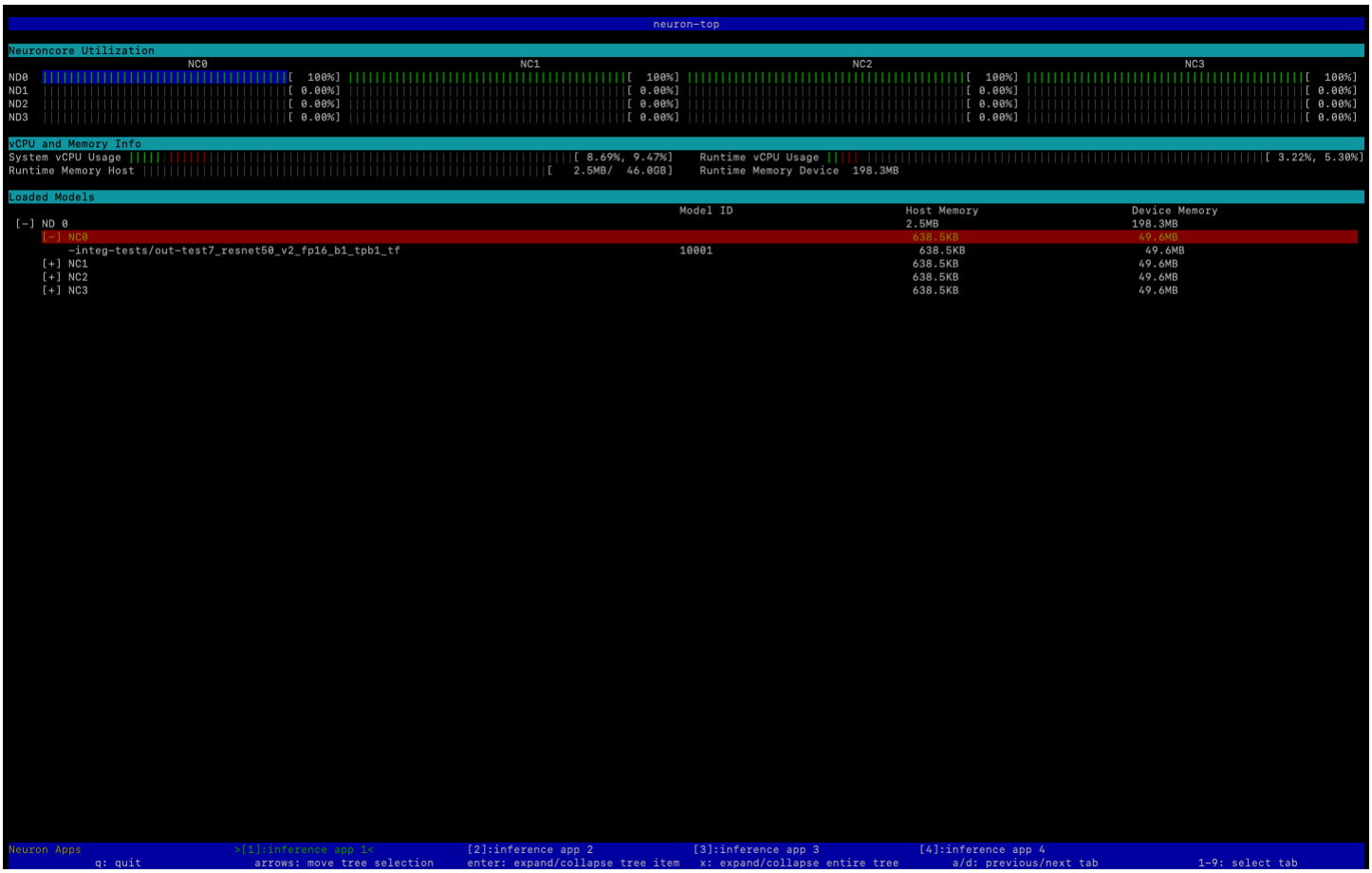
- PERANGKAT NEURON: ID logis yang ditetapkan ke NeuronDevice. ID ini digunakan saat mengonfigurasi beberapa runtime untuk menggunakan yang berbeda. NeuronDevices
- NEURON CORES: Jumlah NeuronCores hadir di NeuronDevice.
- MEMORI NEURON: Jumlah memori DRAM di. NeuronDevice
- PERANGKAT YANG TERHUBUNG: Lainnya NeuronDevices terhubung ke NeuronDevice.
- PCI BDF: ID Fungsi Perangkat Bus PCI (BDF) dari file. NeuronDevice

Lihat Penggunaan Sumber Daya

Lihat informasi yang berguna tentang NeuronCore dan pemanfaatan vCPU, penggunaan memori, model yang dimuat, dan aplikasi Neuron dengan perintah. `neuron-top` `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 perangkat lunak Neuron SDK dalam DLAMI, lihat AWS Panduan Pengaturan Neuron.

Langkah Selanjutnya

Menggunakan DLAMI dengan Neuron AWS

Menggunakan DLAMI dengan Neuron AWS

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 AMI (DLAMI) sudah diinstal sebelumnya dengan semua yang Anda butuhkan untuk mengkompilasi dan menjalankan inferensi dalam instance Inf1 yang menggunakan Inferentia.

Bagian berikut menjelaskan cara menggunakan DLAMI dengan 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 Neuron SDK, lihat dokumentasi AWS Neuron SDK.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan Instance DLAMI dengan Neuron AWS. 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 API yang sama dengan 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 Neuron SDK, lihat dokumentasi AWS Neuron SDK.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan Instance DLAMI dengan Neuron AWS. 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 DRAM perangkat Inferentia 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 gRPC 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 Neuron Compiler AWS

API kompilasi MXNET-neuron menyediakan metode untuk mengkompilasi grafik model yang dapat Anda jalankan pada perangkat Inferentia. AWS

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

Untuk informasi lebih lanjut tentang Neuron SDK, lihat dokumentasi AWS Neuron SDK.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan Instance DLAMI dengan Neuron AWS. 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 API Python kompilasi MXNET-neuron untuk mengkompilasi model -50. ResNet

```
import mxnet as mx
import numpy as np

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

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

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

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

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

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 model MXNet pra-terlatih untuk melakukan klasifikasi gambar real-time dengan Multi Model Server (MMS). MMS adalah 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 menggunakan MXNet.

Untuk informasi lebih lanjut tentang Neuron SDK, lihat dokumentasi AWS Neuron SDK.

Daftar Isi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan Instance DLAMI dengan Neuron AWS. 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 API NDArray dan 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 delete 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

API kompilasi PyTorch -Neuron 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 Dokumentasi AWS Neuron PyTorch SDK.

Daftar Isi

- ResNet50 Inferensi

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan diMeluncurkan Instance DLAMI dengan Neuron AWS. 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 Python API 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']
```

DLAMI ARM64

AWS ARM64 GPU DLAMI dirancang 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 ARM64 GPU DLAMI telah dikonfigurasi sebelumnya dengan Docker, NVIDIA Docker, NVIDIA Driver, CUDA, cuDNN, NCCL, 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 dipercepat GPU dengan biaya yang jauh lebih rendah jika dibandingkan dengan instans berbasis x86 dengan akselerasi GPU.

Pilih DLAMI ARM64

Luncurkan instans G5G dengan ARM64 DLAMI pilihan Anda.

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

Untuk daftar DLAMI ARM64 terbaru, lihat Catatan Rilis untuk DLAMI.

Mulai

Topik berikut menunjukkan cara memulai menggunakan DLAMI ARM64.

Daftar Isi

Menggunakan DLAMI PyTorch GPU ARM64

AWS Deep Learning AMI Ini siap digunakan dengan GPU berbasis prosesor Arm64, dan dioptimalkan untuk. PyTorch PyTorch DLAMI GPU ARM64 mencakup lingkungan Python yang

telah dikonfigurasi sebelumnya PyTorchdengan, TorchVision, dan untuk pelatihan pembelajaran mendalam TorchServedan 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 pelatihan MNIST:

```
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 menyediakan 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 AMI Pembelajaran Mendalam dengan Conda. Klik salah satu opsi untuk mempelajari cara menggunakannya.

Topik

- TensorFlow Melayani
- TorchServe

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 model MNIST. `~/examples/tensorflow-serving/`

Untuk menjalankan salah satu contoh ini, pertama-tama hubungkan ke AMI Pembelajaran Mendalam Anda 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 DLAMI Anda.

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. Nama 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 Model MNIST

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 model MNIST. 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 AMI Pembelajaran Mendalam 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 (Amazon EC2) 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 di JSON yang 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 instance DLAMI Anda. Anda dapat melihatnya di halaman contoh repositori TorchServe proyek.

Info Lebih Lanjut

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

Meningkatkan DLAMI Anda

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

Topik

Upgrade ke Versi DLAMI Baru

Gambar sistem DLAMI diperbarui secara teratur untuk memanfaatkan rilis kerangka pembelajaran mendalam 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 DLAMI baru. Dengan cara ini, Anda dapat sering meningkatkan, sambil meminimalkan waktu yang diperlukan untuk mentransisikan data Anda.

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

1. Gunakan Amazon EC2Console untuk membuat volume Amazon EBS baru. Untuk petunjuk mendetail, lihat Membuat Volume Amazon EBS.
2. Lampirkan volume Amazon EBS yang baru dibuat ke DLAMI yang sudah ada. Untuk petunjuk mendetail, lihat Melampirkan Volume Amazon EBS.
3. Transfer data Anda, seperti dataset, pos pemeriksaan, dan file konfigurasi.
4. Luncurkan DLAMI. Untuk petunjuk terperinci, lihat Meluncurkan dan Mengkonfigurasi DLAMI.
5. Lepaskan volume Amazon EBS dari DLAMI lama Anda. Untuk petunjuk mendetail, lihat Melepaskan Volume Amazon EBS.

6. Lampirkan volume Amazon EBS ke DLAMI baru Anda. Ikuti instruksi dari Langkah 2 untuk melampirkan volume.
7. Setelah Anda memverifikasi bahwa data Anda tersedia di DLAMI baru Anda, hentikan dan hentikan DLAMI lama Anda. Untuk instruksi pembersihan terperinci, lihat. Pembersihan

Kiat untuk Pembaruan Perangkat Lunak

Dari waktu ke waktu, Anda mungkin ingin memperbarui perangkat lunak secara manual pada DLAMI Anda. Hal ini umumnya dianjurkan bahwa Anda menggunakan `pip` untuk memperbarui paket Python. Anda juga harus menggunakan `pip` untuk memperbarui paket dalam lingkungan Conda pada AMI Pembelajaran Mendalam 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 AMI 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”. DLAMI memastikan bahwa semua lingkungan yang disediakan DLAMI sudah benar, tetapi tidak dapat menjamin bahwa setiap paket yang diinstal pengguna akan berfungsi dengan benar.

Menerima Pemberitahuan tentang Pembaruan Baru

> **Note**
>
> AWS AMI Deep Learning 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 DLAMI baru dirilis. Pemberitahuan diterbitkan dengan Amazon SNS menggunakan topik berikut.

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

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

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

1. Buka konsol Amazon SNS.
2. Di bilah navigasi, ubah AWS Wilayah ke AS Barat (Oregon), jika perlu. Anda harus memilih wilayah tempat notifikasi SNS 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 (ARN) berikut: **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**
   b. Untuk Protokol, pilih salah satu dari [Amazon SQS, AWS Lamda, Email, Email-JSON]
   c. Untuk Endpoint, masukkan alamat email atau Nama Sumber Daya Amazon (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 AMI

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 menjalankan AWS layanan di 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 untuk DLAMI, AWS lihat Layanan dalam Lingkup oleh Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup 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 menggunakan DLAMI. Topik berikut menunjukkan cara mengonfigurasi DLAMI untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya DLAMI Anda.

Untuk informasi selengkapnya, lihat Keamanan di Amazon EC2.

Topik

Perlindungan Data di AWS Deep Learning AMI

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AMI AWS Pembelajaran Mendalam. 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. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal 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 autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- 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 modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

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 saat Anda bekerja dengan DLAMI atau Layanan AWS lainnya menggunakan konsol, API AWS CLI, atau SDK. AWS 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. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan

supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Identity and Access Management di AWS Deep Learning AMI

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

Untuk informasi selengkapnya tentang Identity and Access Management, lihat Identity and Access Management untuk Amazon EC2.

Topik

- Mengautentikasi Menggunakan Identitas
- Mengelola Akses Menggunakan Kebijakan
- IAM dengan Amazon EMR

Mengautentikasi Menggunakan Identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. 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 penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

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 selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) dalam AWS dalam 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 kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan Grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. 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 yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

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 selengkapnya, lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial 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 dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk 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 Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil 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 pengguna atau peran IAM 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. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS 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 ketika mengajukan permintaan FAS, lihat Sesi akses maju.
- Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. 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. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensyal sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

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 dokumen JSON. Untuk

informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

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

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan Berbasis Identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. 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 kebijakan IAM dalam 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 dilampirkan 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 yang dikelola atau kebijakan inline, lihat Memilih antara kebijakan yang dikelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan Berbasis Sumber Daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM 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 dari IAM dalam kebijakan berbasis sumber daya.

Daftar Kontrol Akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

Tipe Kebijakan Lainnya

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.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran 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 batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON 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 di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .

- 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 eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai Tipe 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 Pengguna IAM.

IAM dengan Amazon EMR

Anda dapat menggunakan AWS Identity and Access Management Amazon EMR untuk menentukan pengguna, AWS sumber daya, grup, peran, dan kebijakan. Anda juga dapat mengontrol AWS layanan mana yang dapat diakses oleh pengguna dan peran ini.

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

Logging dan Monitoring di AWS Deep Learning AMI

AWS Deep Learning AMI Instans Anda dilengkapi dengan beberapa alat pemantauan GPU termasuk utilitas yang melaporkan statistik penggunaan GPU ke Amazon. CloudWatch Untuk informasi selengkapnya, lihat Pemantauan dan Pengoptimalan dan Pemantauan GPU Amazon EC2.

Pelacakan Penggunaan

Distribusi sistem AWS Deep Learning AMI operasi berikut mencakup kode yang memungkinkan AWS untuk mengumpulkan jenis instance, ID instance, tipe DLAMI, dan informasi OS. Tidak ada informasi tentang perintah yang digunakan dalam DLAMI yang dikumpulkan atau disimpan. Tidak ada informasi lain tentang DLAMI yang dikumpulkan atau disimpan.

- Ubuntu 16.04
- Ubuntu 18.04

- Ubuntu 20.04
- Amazon Linux 2

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

Validasi Kepatuhan untuk AWS Deep Learning AMI

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

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan . 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 Pengunduhan AWS Artifak .

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 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— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di AWS Deep Learning AMI

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah 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 AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

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

Keamanan Infrastruktur di AWS Deep Learning AMI

Keamanan infrastruktur didukung AWS Deep Learning AMI oleh Amazon EC2. Untuk informasi selengkapnya, lihat Keamanan Infrastruktur di Amazon EC2.

Perubahan Penting pada DLAMI

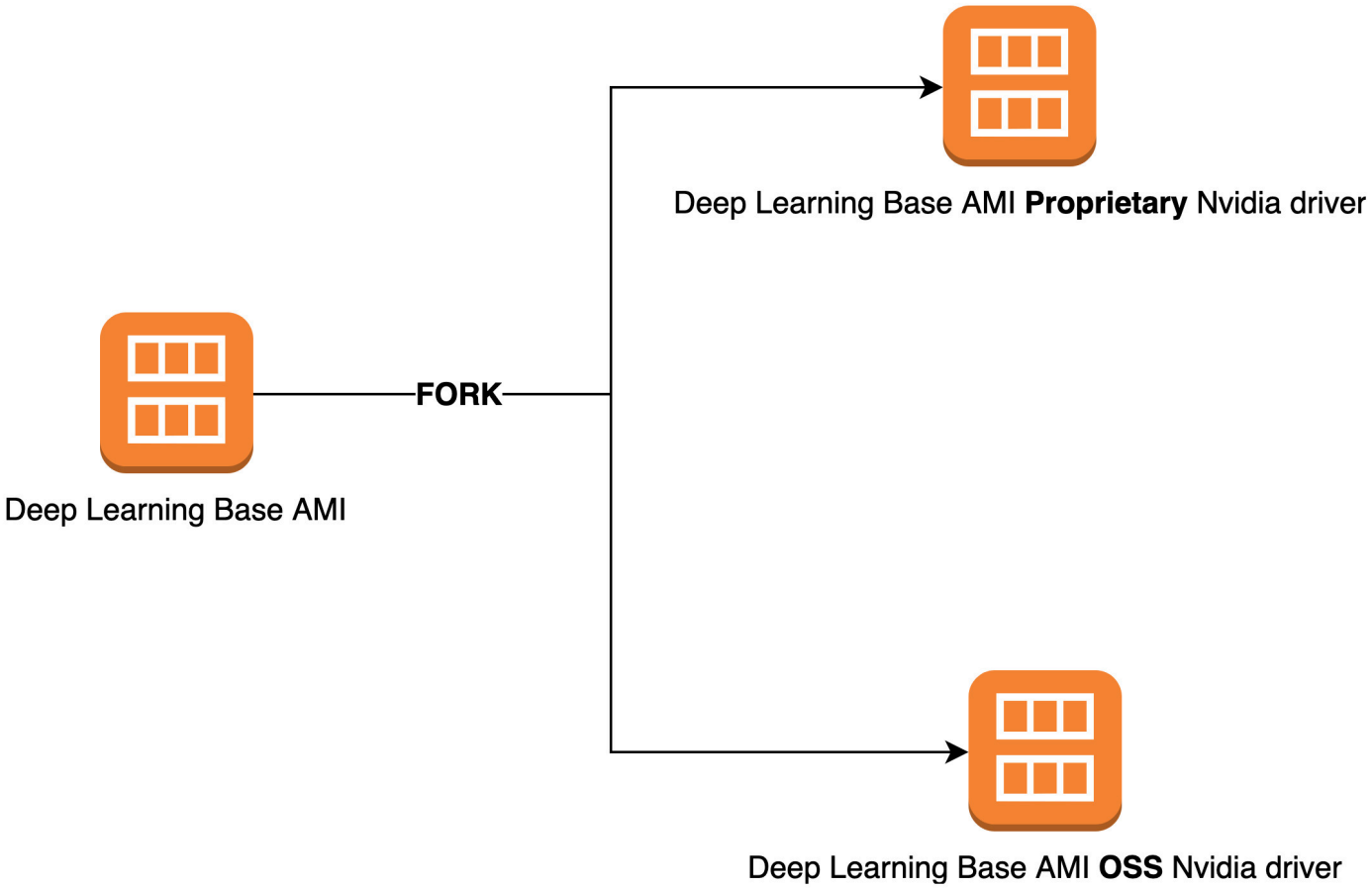
Pertanyaan yang Sering Diajukan

Apa yang berubah?

Pada 15/11/2023 AWS Deep Learning AMI (DLAMI) akan dibagi menjadi dua kelompok terpisah:

- DLAMI yang menggunakan driver berpemilik Nvidia (untuk mendukung P3, P3dn, G3).
- DLAMI yang menggunakan driver Nvidia OSS (untuk mendukung G4dn, G5, P4, P5).

Akibatnya, DLAMI baru akan dibuat untuk masing-masing dari dua kategori dengan nama baru dan ID AMI baru. DLAMI ini tidak akan dapat dipertukarkan - yaitu DLAMI dari satu grup tidak akan mendukung instance yang didukung oleh grup lain misalnya DLAMI yang mendukung p5 tidak akan mendukung g3 dan sebaliknya.



Mengapa perubahan ini diperlukan?

Saat ini DLAMI untuk GPU NVIDIA menyertakan driver kernel berpemilik dari NVIDIA. Namun, baru-baru ini komunitas kernel Linux hulu menerima perubahan yang mengisolasi driver kernel berpemilik, seperti driver GPU NVIDIA, dari berkomunikasi dengan driver kernel lainnya. Perubahan ini menonaktifkan GPUDirect RDMA pada instance seri P4/P5, yang merupakan mekanisme yang memungkinkan GPU menggunakan EFA secara efisien untuk pelatihan terdistribusi. Akibatnya DLAMI akan menggunakan driver OpenRM (driver open source NVIDIA), ditautkan dengan driver EFA open source untuk mendukung G4dn, G5, P4 dan P5. Namun, driver OpenRM ini tidak akan mendukung instance lama (P3, G3, dll.) Oleh karena itu, untuk memastikan bahwa kami terus menyediakan DLAMI saat ini, berkinerja, dan aman yang mendukung kedua jenis instance, kami akan membagi DLAMI menjadi dua grup - satu dengan driver OpenRM (mendukung G4dn, G5, P4 dan P5) dan satu dengan driver berpemilik yang lebih lama (mendukung instance lama P3, P3dn, G3).

DLAMI mana yang terpengaruh oleh perubahan ini?

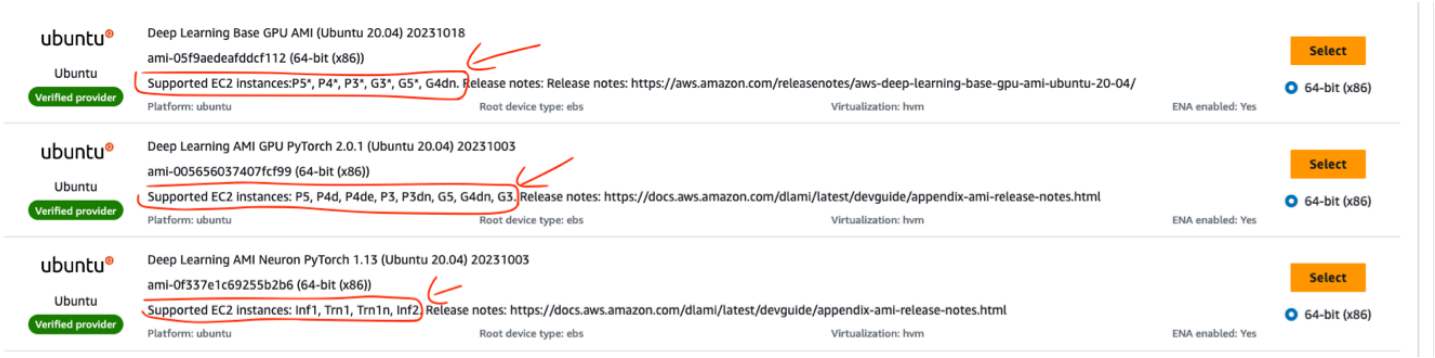
Semua DLAMI dipengaruhi oleh perubahan ini.

Apa artinya ini bagi Anda?

DLAMI baru akan terus menyediakan fungsionalitas, kinerja, dan keamanan DLAMI saat ini selama mereka dijalankan pada jenis instance yang kompatibel. Jika Anda menggunakan DLAMI maka Anda perlu memastikan bahwa DLAMI diluncurkan pada salah satu contoh kompatibel yang disebutkan dalam catatan rilis setiap DLAMI (lihat di sini). Misalnya: Anda perlu mengakomodasi perubahan ini ke:

- Panggil DLAMI dengan kueri CLI yang tepat (lihat di bawah)
- Luncurkan DLAMIS dari konsol dan CLI pada jenis instans yang kompatibel

Jika Anda meluncurkan DLAMIS dari konsol EC2 Quickstart: Setiap deskripsi DLAMI mencantumkan jenis instans yang didukung di konsol EC2. Anda harus meluncurkan DLAMI pada instance yang kompatibel.



Jika Anda meluncurkan DLAMIS menggunakan CLI maka Anda harus memodifikasi kueri Anda. Sebagai contoh:

Saat ini kueri CLI berikut digunakan untuk DLAMI dasar yang mendukung semua instance [P3, P3dn, G3, G4dn, G5, P4, P5]:

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

Kueri CLI baru adalah:

Untuk DLAMI dasar yang mendukung P3, P3dn, dan G3:

```
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 DLAMI dasar yang mendukung G4dn, G5, P4, dan P5:

```
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
```

Silakan lihat catatan rilis yang diperbarui untuk AMI baru di sini. Untuk cara meluncurkan AMI pada instans EC2, silakan lihat instruksi di sini.

Kapan Anda harus mulai menggunakan DLAMI baru?

Anda harus mulai menggunakan DLAMI baru sesegera mungkin untuk kerangka kerja, dependensi, tambalan, dan fungsionalitas terbaru. Secara opsional, jika Anda menggunakan Amazon Linux 2 DLAMI yang dirilis sebelum 11/8/2023, maka Anda dapat memilih untuk melanjutkan penambalan langsung DLAMI mereka (lihat instruksi di sini) hingga 11/30/2023.

Apakah akan ada kehilangan fungsionalitas dengan DLAMI baru?

Tidak, tidak ada kehilangan fungsionalitas dengan DLAMI baru. DLAMI baru setelah pemisahan akan terus menyediakan semua fungsionalitas, kinerja, dan keamanan DLAMI lama sebelum dibagi, selama dijalankan pada instance yang kompatibel. Kami membagi DLAMI menjadi dua grup sehingga kami terus menawarkan DLAMI yang terkini, berkinerja, dan aman untuk Anda gunakan pada berbagai instance.

Bagaimana dengan DLC?

DLC tidak menyertakan driver NVIDIA sehingga tidak terpengaruh oleh perubahan ini. Tetapi Anda harus memastikan bahwa DLC dijalankan pada AMI yang kompatibel dengan instance yang mendasarinya.

Informasi Terkait

Topik

- Forum
- Posting blog terkait
- Pertanyaan yang Sering Diajukan

Forum

- Forum: AMIAWS Pembelajaran Mendalam

Posting blog terkait

- Daftar Artikel Terbaru Terkait AMI Pembelajaran Mendalam
- LuncurkanAWS Deep Learning AMI (dalam 10 menit)
- Pelatihan Lebih Cepat dengan Optimized TensorFlow 1.6 pada Instans Amazon EC2 C5 dan P3
- AMIAWS Pembelajaran Mendalam Baru untuk Praktisi Machine Learning
- Kursus Pelatihan Baru Tersedia: Pengantar Machine Learning & Pembelajaran MendalamAWS
- Perjalanan ke Deep Learning denganAWS

Pertanyaan yang Sering Diajukan

- T: Bagaimana cara melacak pengumuman produk terkait DLAMI?

  Berikut adalah dua saran untuk ini:

  - Bookmark kategori blog ini, "AMI PembelajaranAWS Mendalam" ditemukan di sini: Daftar Artikel Terbaru Terkait AMI Pembelajaran Mendalam.
  - "Tonton" Forum: AMIAWS Pembelajaran Mendalam
- T: Apakah driver NVIDIA dan CUDA diinstal?

  Ya. Beberapa DLAM memiliki versi yang berbeda. IniPembelajaran Mendalam AMI dengan Conda memiliki versi terbaru dari setiap DLAMI. Ini dibahas secara lebih rinci dalamInstalasi CUDA dan

Framework Bindings. Anda juga dapat merujuk ke catatan rilis AMI tertentu untuk mengonfirmasi apa yang diinstal.

- T: Apakah CuDNN diinstal?

  Ya.

- T: Bagaimana saya melihat bahwa GPU terdeteksi dan statusnya saat ini?

  Jalankan `nvidia-smi`. Ini akan menampilkan satu atau lebih GPU, tergantung pada jenis instans, bersama dengan konsumsi memori mereka saat ini.

- T: Apakah lingkungan virtual disiapkan untuk saya?

  Ya, tapi hanya padaPembelajaran Mendalam AMI dengan Conda.

- T: Versi Python apa yang diinstal?

  Setiap DLAMI memiliki Python 2 dan 3. Pembelajaran Mendalam AMI dengan CondaMemiliki lingkungan untuk kedua versi untuk setiap kerangka kerja.

- T: Apakah Keras diinstal?

  Ini tergantung pada AMI. Pembelajaran Mendalam AMI dengan CondaMemiliki Keras tersedia sebagai front end untuk setiap kerangka kerja. Versi Keras bergantung pada dukungan kerangka kerja untuk itu.

- Q. apakah ini gratis?

  Semua DLAM gratis. Namun, bergantung pada jenis instans yang Anda pilih, instans mungkin tidak gratis. LihatHarga untuk DLAMI untuk info lebih lanjut.

- T: Saya mendapatkan kesalahan CUDA atau pesan terkait GPU dari kerangka kerja saya. Apa yang salah?

  Periksa jenis instans apa yang Anda gunakan. Perlu memiliki GPU untuk banyak contoh dan tutorial untuk bekerja. Jika berjalan tidak`nvidia-smi` menunjukkan GPU, maka Anda perlu memutar DLAMI lain menggunakan instance dengan satu atau lebih GPU. LihatMemilih Jenis Instance untuk DLAMI untuk info lebih lanjut.

- T: Dapatkah saya menggunakan Docker?

  Docker telah diinstal sebelumnya sejak versi 14 dari Deep Learning AMI dengan Conda. Perhatikan bahwa Anda akan ingin menggunakan nvidia-docker pada instans GPU untuk menggunakan GPU.

- T: Wilayah apa saja yang Linux DLAMis tersedia?

Wilayah	Code
US East (Ohio)	us-east-2
US East (N. Virginia)	as-east-1
GovCloud	us-gov-west-1
US West (Northern California)	us-west-1
US West (Oregon)	us-west-2
Beijing (Tiongkok)	cn-north-1
Ningxia (Tiongkok)	cn-northwest-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
EU (Frankfurt)	eu-central-1
EU (Ireland)	eu-west-1
EU (London)	eu-west-2
EU (Paris)	eu-west-3
SA (Sao Paulo)	sa-east-1

- T: Wilayah apa saja Windows DLAM tersedia di?

Wilayah	Code
US East (Ohio)	us-east-2
US East (N. Virginia)	as-east-1
GovCloud	us-gov-west-1
US West (Northern California)	us-west-1
US West (Oregon)	us-west-2
Beijing (Tiongkok)	cn-north-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
EU (Frankfurt)	eu-central-1
EU (Ireland)	eu-west-1
EU (London)	eu-west-2
EU (Paris)	eu-west-3
SA (Sao Paulo)	sa-east-1

Catatan Rilis untuk DLAMI

> **Note**
>
> AWS Deep Learning AMI s memiliki irama rilis malam untuk patch keamanan. Patch keamanan tambahan ini tidak termasuk dalam catatan rilis resmi.

Silakan rujuk halaman Kebijakan Dukungan DLAMI untuk catatan rilis kerangka kerja yang tidak didukung.

DLAMI Dasar

GPU

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

Qualcomm

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

DLAMI Kerangka Tunggal

PyTorch-AMI Spesifik

GPU

- X86
  - 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 2.2 (Ubuntu 20.04)

AWS Neuron

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

TensorFlow-AMI Spesifik

GPU

- X86
  - 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 Pembelajaran Mendalam AMI GPU TensorFlow 2.13 (Amazon Linux 2)
  - AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.13 (Ubuntu 20.04)

AWS Neuron

- X86

- AWS Pembelajaran Mendalam AMI Neuron TensorFlow 2.10 (Amazon Linux 2)
- AWS Pembelajaran Mendalam AMI Neuron TensorFlow 2.10 (Ubuntu 20.04)

DLAMI Multi-Kerangka

> **Note**
>
> Jika Anda hanya menggunakan satu kerangka pembelajaran mesin, kami merekomendasikan DLAMI Kerangka Tunggal

GPU

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

AWS Neuron

- X86
  - AWS Pembelajaran Mendalam AMI Neuron (Ubuntu 22.04)

Depresiasi Pemberitahuan Depresiasi Depresiasi Depresiasi Depresiasi

Tabel berikut mencantumkan informasi tentang fitur usang diAWS Deep Learning AMI.

Fitur Fitur yang tidak digunakan Fitur Depresiasi	Tanggal Tanggal Tanggal Tanggal Tanggal Tanggal Tanggal Depresiasi Tanggal	Pemberitahuan Depresiasi Pemberitahuan Depresiasi
Ubuntu 16.04	10/07/2021	Ubuntu Linux 16.04 LTS mencapai akhir jendela LTS lima tahun pada 30 April 2021 dan tidak lagi didukung oleh vendornya. Tidak ada lagi pembaruan untuk 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 pada Oktober 2021. Deep Learning AMI (Amazon Linux) yang lebih baru akan terus tersedia.
Chainer	07/01/2020	Chainer telah mengumumkan akhir rilis besar pada Desember 2019. Akibatnya, kami tidak akan lagi menyertakan lingkungan

Fitur Fitur yang tidak digunakan Fitur Depresiasi	Tanggal Tanggal Tanggal Tanggal Tanggal Tanggal Tanggal Depresiasi Tanggal	Pemberitahuan Depresiasi Pemberitahuan Depresiasi
		Chainer Conda di DLAMI mulai Juli 2020. Rilis sebelumnya dari DLAMI yang berisi lingkungan ini akan terus tersedia. Kami akan memberika n 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. Komunitas TensorFlo w, PyTorch, dan MXNet juga telah mengumumk an bahwa rilis TensorFlo w 1.15, TensorFlow 2.1, PyTorch 1.4, dan MXNet 1.6.0 akan menjadi yang terakhir yang mendukung Python 2.

Riwayat Dokumen untuk Panduan AWS Deep Learning AMI Pengembang

Perubahan	Deskripsi	Tanggal
ARM64 DLAMI	AWS Deep Learning AMI Sekarang mendukung gambar pada GPU berbasis prosesor Arm64.	29 November 2021
TensorFlow 2	AMI Pembelajaran Mendalam dengan 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 Neuron SDK.	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
Elastic Inference	Prasyarat inferensi elastis dan info terkait telah ditambahkan ke panduan pengaturan.	28 November 2018
Menginstal PyTorch dari Nightly Build	Sebuah tutorial telah ditambahkan yang mencakup bagaimana Anda dapat menghapus instalasi PyTorch, lalu menginstal build malam PyTorch pada AMI Pembelaja	25 September 2018

	ran Mendalam Anda dengan Conda.	
Docker sekarang sudah diinstal sebelumnya di DLAMI Anda	Sejak v14 dari Deep Learning AMI dengan Conda, Docker dan versi NVIDIA dari Docker untuk GPU telah diinstal sebelumnya.	25 September 2018
Conda Tutorial	Contoh MOTD diperbarui untuk mencerminkan rilis yang lebih baru.	Juli 23, 2018

Pembaruan Sebelumnya:

Tabel berikut menjelaskan perubahan penting dalam setiap rilis AWS Deep Learning AMI 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 Deep Learning AMI”.	26 April 2018
Tutorial rantai	Tutorial untuk menggunakan Chainer dalam mode multi-	28 Februari 2018

Perubahan	Deskripsi	Tanggal
	GPU, GPU tunggal, dan CPU ditambahkan. Integrasi CUDA ditingkatkan dari CUDA 8 ke CUDA 9 untuk beberapa kerangka kerja.	
Linux AMI v3.0, ditambah pengenalan MXNet Model Server, Serving, dan TensorFlow TensorBoard	Ditambahkan tutorial untuk Conda AMI dengan model baru dan kemampuan melayani visualisasi menggunakan MXNet Model Server v0.1.5, Melayani v1.4.0, dan v0.4.0. TensorFlo w TensorBoard AMI dan kerangka kerja kemampuan CUDA dijelaskan dalam ikhtisar Conda dan CUDA. Catatan rilis terbaru dipindahk an ke https://aws.amazo n.com/releasenotes/	25 Januari 2018
Linux AMI v2.0	Base, Source, dan Conda AMI diperbarui dengan NCCL 2.1. Sumber dan Conda AMI diperbarui dengan MxNet v1.0, 0.3.0 PyTorch , dan Keras 2.0.9.	11 Desember 2017
Dua opsi AMI Windows ditambahkan	Windows 2012 R2 dan 2016 AMI dirilis: ditambahkan ke panduan pemilihan AMI 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

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

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