
AWS
re:Invent
NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

CMP324-R2

Scaling machine learning workloads on Amazon EC2

Arun Subramaniyan (he/him)
Head of HPC, Quantum, and ML Solutions
Amazon Web Services

Mark Saroufim (he/him)
PyTorch Partner Engineer
Meta

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Outline

- AWS ML stack
- ML on EC2
- Distributed Training
- Accelerated Inference
- Bin packing models
- Large-scale Customer Examples
- Scaling PyTorch

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

THE AWS ML STACK

Broadest and most complete set of Machine Learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALIZATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Contact Lens *For Amazon Connect*

ML SERVICES

Amazon SageMaker

Ground Truth

AWS Marketplace for ML

SageMaker Studio IDE

Built-in algorithms | Notebooks | Experiments | Processing | Model training & tuning | Debugger | Autopilot | Model hosting | Model Monitor

Neo

Augmented AI

SELF-MANAGED ML

TensorFlow mxnet PYTORCH

GLUON Keras HOROVOD DeepGraphLibrary

Deep Learning AMIs & Containers | GPUs & CPUs | Elastic Inference | Inferentia | FPGA

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

PyTorch, the PyTorch logo and any related marks are trademarks of Facebook Inc.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML on EC2

Broad choices to meet your evolving development needs

*OSS – Open source software

TOOLS

DATA PROCESSING & LABELING


Amazon Kinesis


AWS Glue


Amazon EMR


Amazon SageMaker Ground Truth

OSS Kafka Apache Spark hadoop

DL FRAMEWORKS


AWS Deep Learning AMIs (DLAMI)


AWS Deep Learning Containers

OSS TensorFlow PYTORCH mxnet

DEV, TRAINING, AND TUNING (OSS)


Notebooks

HOROVOD Katib Kubeflow Operators

AutoGluon DASK

DEPLOY

KFServing*


AWS IoT Greengrass

TF Serving*, NVIDIA Triton*

SERVERLESS


AWS Lambda

Nuclio*

ML OPS

AWS CodeBuild
AWS CodeCommit
AWS CDK
Amazon ECR
Jenkins*

ORCHESTRATION

AWS SERVICES


Amazon EKS


Amazon ECS


AWS Fargate


AWS Batch


AWS ParallelCluster

OSS

Kubernetes

WORKFLOW

AWS/PARTNER SERVICES


AWS Step Functions


AWS Cloud Formation


Terraform

OSS

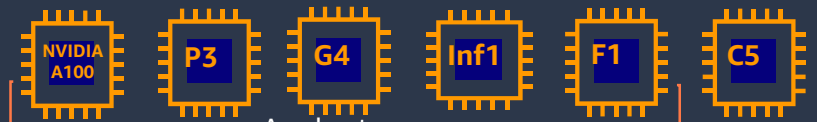
Kubeflow Pipelines

Airflow

INFRASTRUCTURE

COMPUTE

(Preview)



Accelerators

STORAGE


Amazon S3


Amazon EFS



Amazon FSx for Lustre

DATA TRANSFER


AWS Direct connect


AWS Snowcone


AWS Snowball


AWS Snowball Edge

NETWORKING


EFA

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

PyTorch, the PyTorch logo and any related marks are trademarks of Facebook Inc.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Distributed Training

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Distributed Training

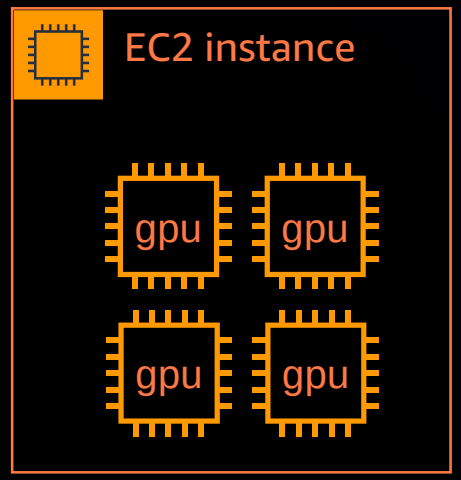
- Real-life deep learning model training requires massive compute
- Training commonly done on large number of multi-GPU nodes
- Nodes need to work in parallel to complete training job
- Shared access to data from all nodes is needed
- Training should be fault tolerant and resumable

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Compute Challenges

NLU Deep Learning models sizes:

- BERT Base – Neural Net with 12 layers, 768 hidden layer dimension, total of 110M parameters
- GPT-3: 175B parameters (2020)
- T5-XXL: 1.6T parameters (2021)



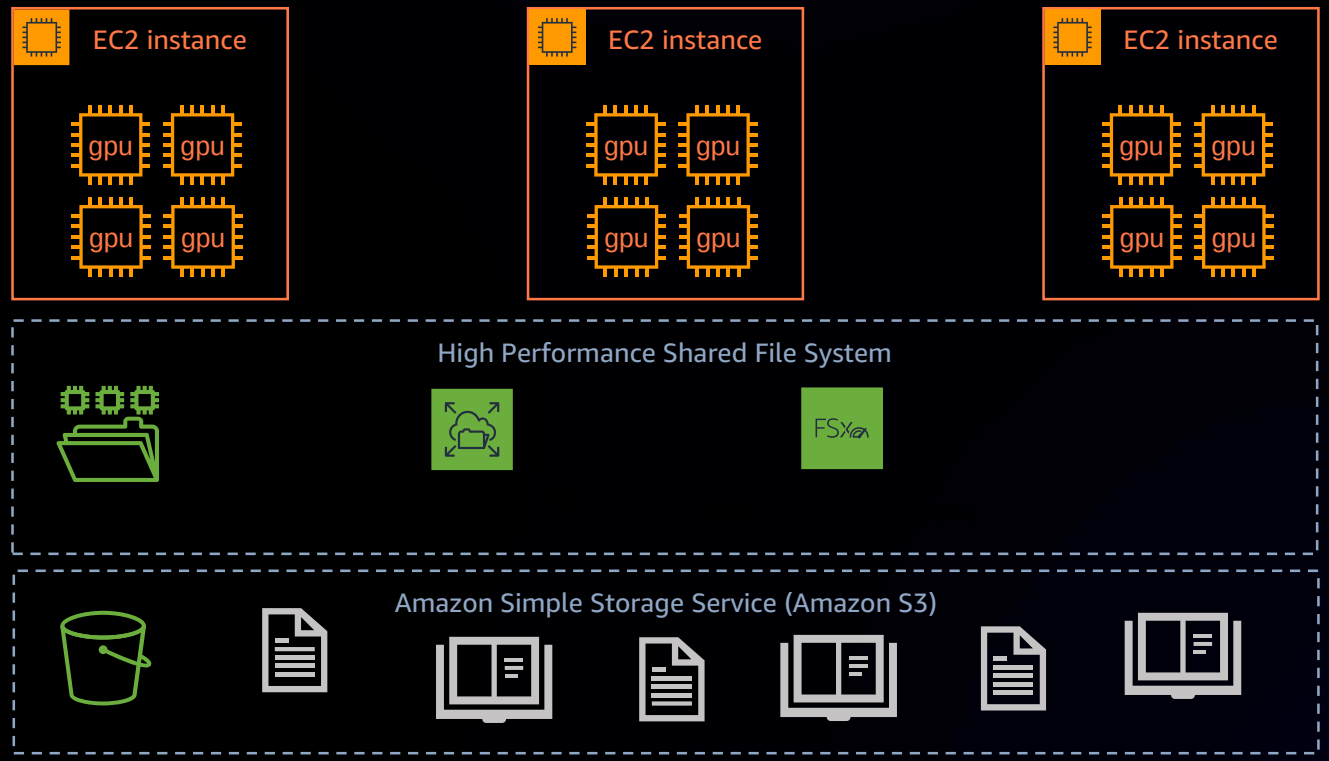
P4d.24xlarge: up to 3.1x speed up as compared to p3dn for BERT models

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Storage & Memory Challenges

NLU models trained on internet scale datasets:

- Original BERT pre-trained on **16 GB of Wikipedia** text (2500M tokens) & **11k books** (800M tokens)
- T5-XXL: Colossal Clean Crawled Corpus **750 GB**



Read intensive job:

GPU Memory bound:

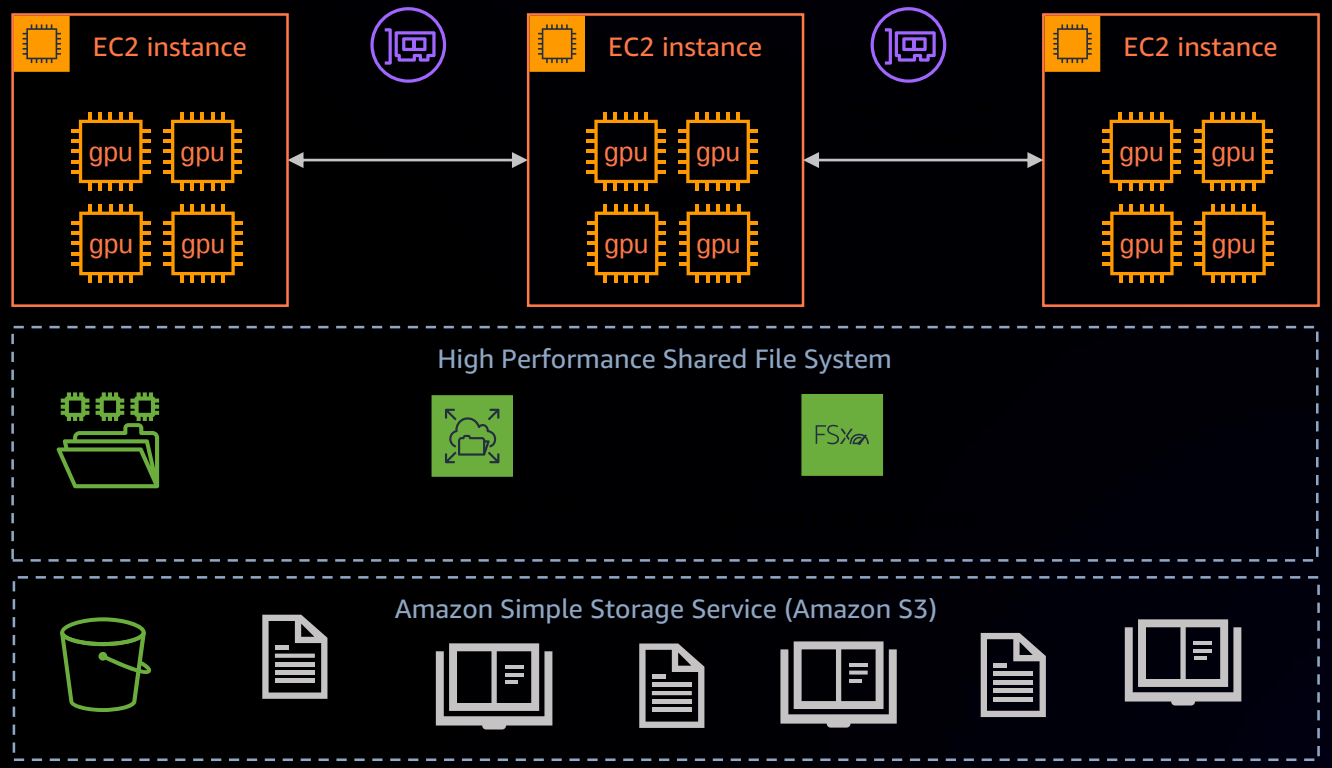
Rule of thumb: Let's

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Networking Challenges

Gradient descent over multi-node multi-gpu architecture:

- Model + mini-batch data fits on single GPU: Data Parallel Training
- Model + mini-batch needs multiple GPUs: Model Parallel Training

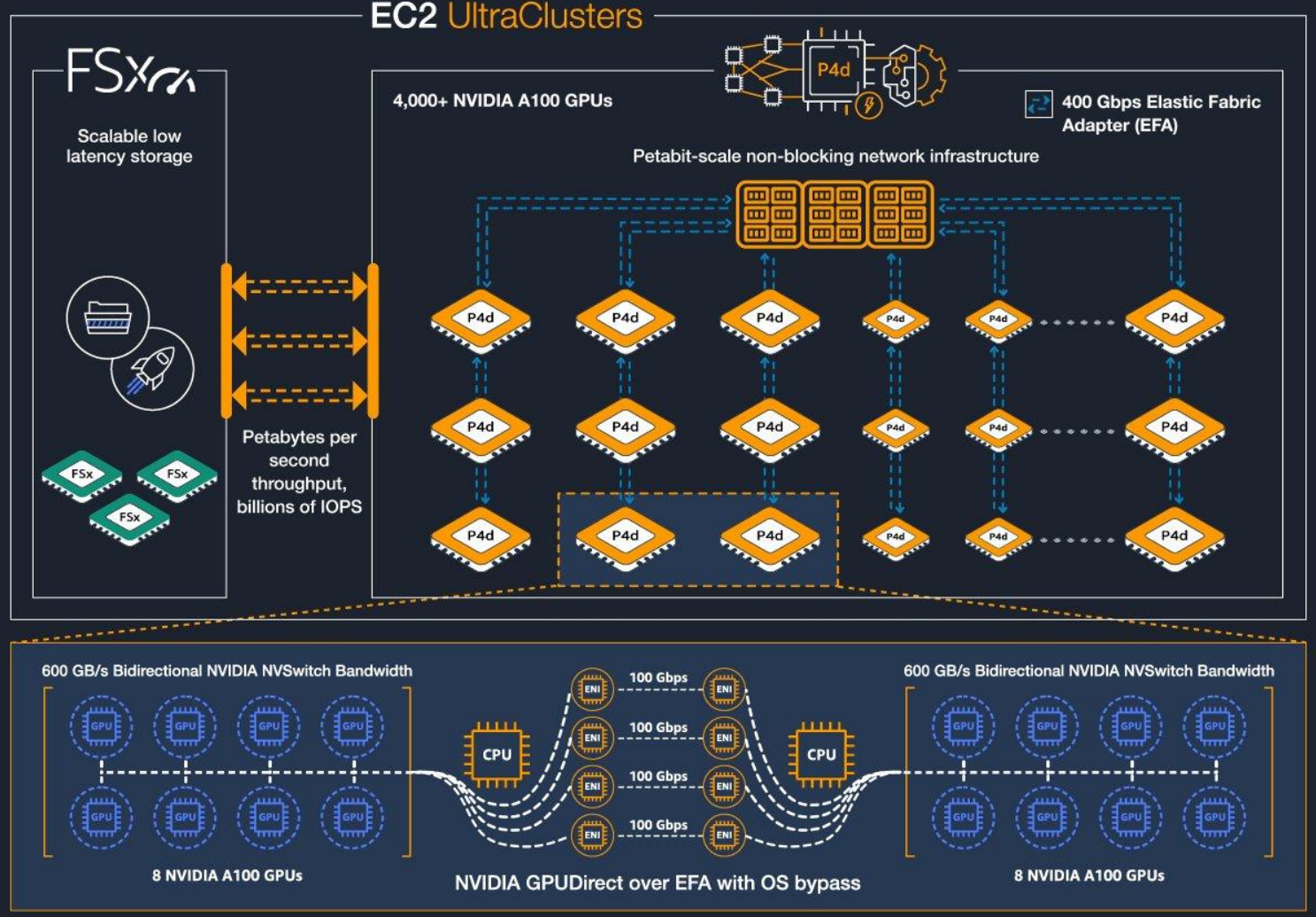


Fast GPU to GPU communication:

Gradient sync strategy can impact compute cluster design!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

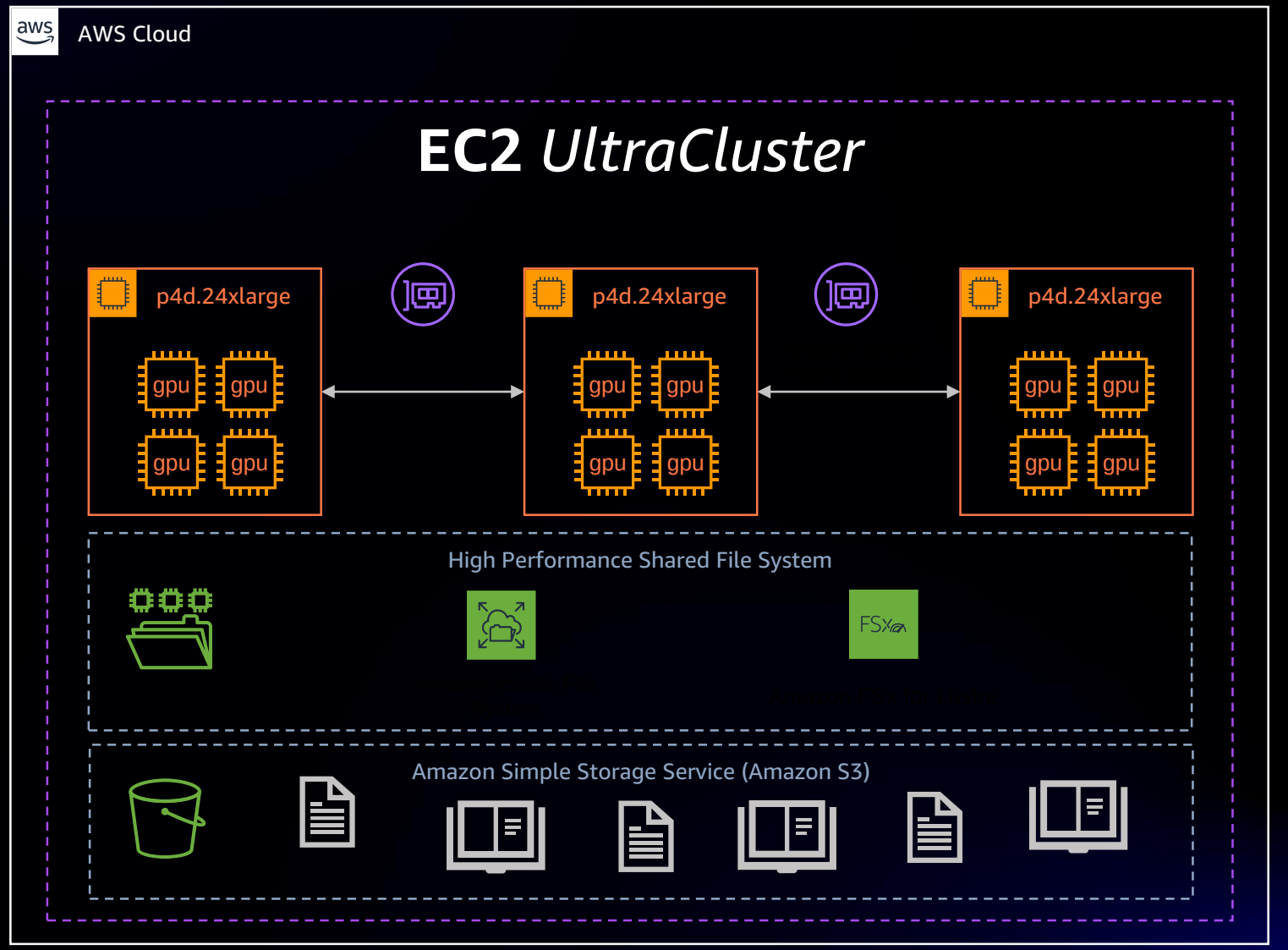
EC2 ULTRA CLUSTERS OF P4d INSTANCES



- **EC2 UltraClusters of P4d** – democratize access to supercomputing-class performance
- Available on demand via a pay-as-you-go model on AWS
- Over 4,000 A100 Tensor Core GPUs
- Petabit-scale non-blocking networking
- High-throughput, low-latency storage with FSx for Lustre
- Support for unlimited storage on S3

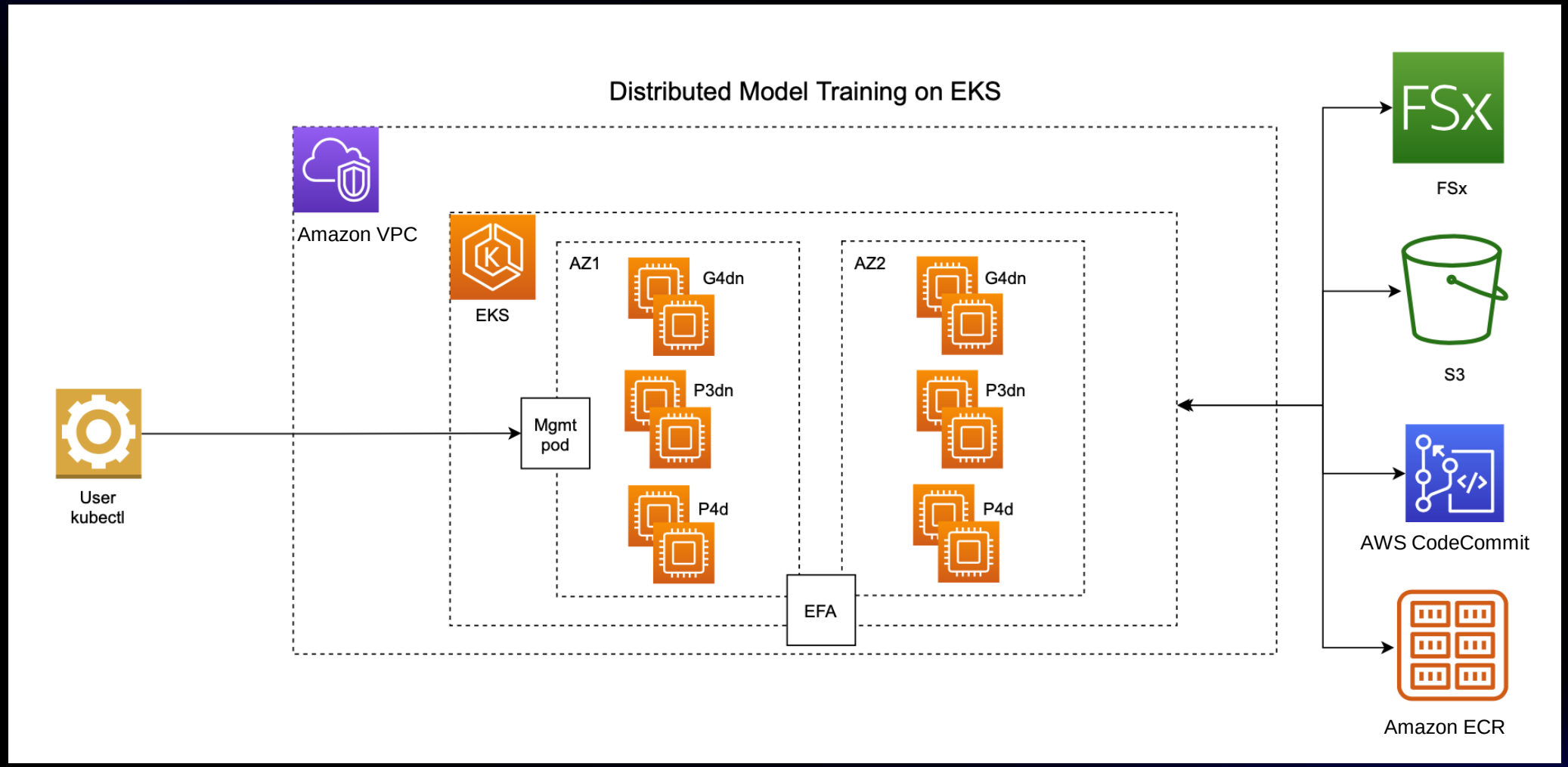
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Distributed ML on p4d UltraClusters



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Distributed Training Infrastructure



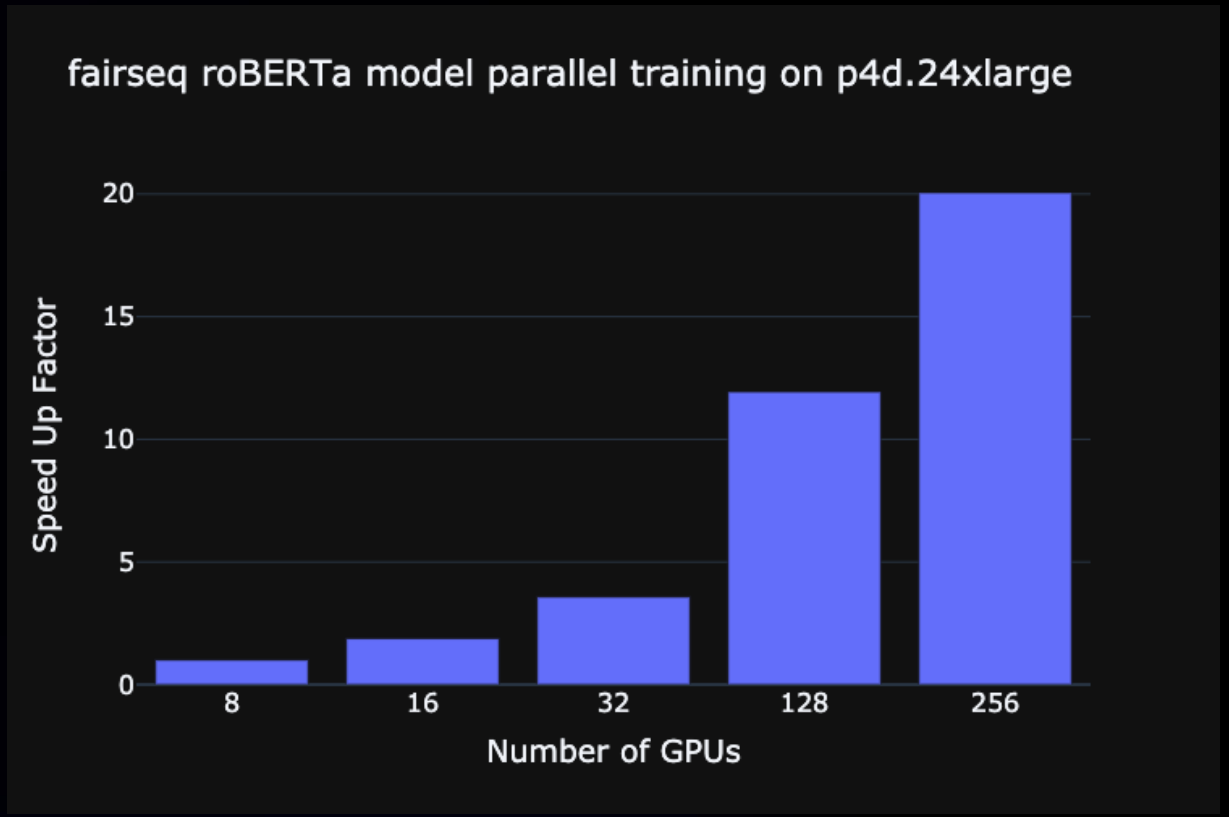
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

roBERTa 3B parameters scaling performance

Using UltraClusters of P4d instances:

- 8x A100 40GB GPUs (reference 1x speed)
- 400 Gbps Networking with EFA
- 300 GB/s NVLink

2.7x speed up from previous generation instance (p3dn.24xlarge)



Drastically reduce model training times by distributing workloads across hundreds or thousands of p4d instances

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Accelerated Inference

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Neuron

- High-performance Software Development Kit (SDK)



Neuron Compiler





Easy to get started

Integrated with major frameworks

PyTorch TensorFlow

mxnet



Minimal Code Change

Deploy existing models with minimal code changes; maintain hardware portability without dependency on AWS software



Documentation, Examples, & Support

github.com/aws/aws-neuron-sdk

aws

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.
PyTorch, the PyTorch logo and any related marks are trademarks of Facebook Inc.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Native integration with minimum code change

- *Run HuggingFace Transformers models on Inferentia with single line of code added*

```
from transformers import BertTokenizer, BertModel
import torch
import torch_neuron

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased',return_dict=False)

inputs = tokenizer("Hello, my dog is cute", return_tensors="pt")

neuron_model = torch.neuron.trace(model,
                                  example_inputs = (inputs['input_ids'],inputs['attention_mask']),
                                  verbose=1)

outputs = neuron_model(*(inputs['input_ids'],inputs['attention_mask']))
```

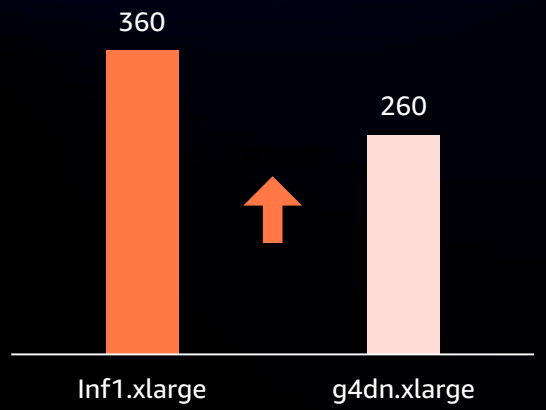


Sample code taken from docstring for BERT Base model: https://huggingface.co/transformers/model_doc/bert.html#bertmodel

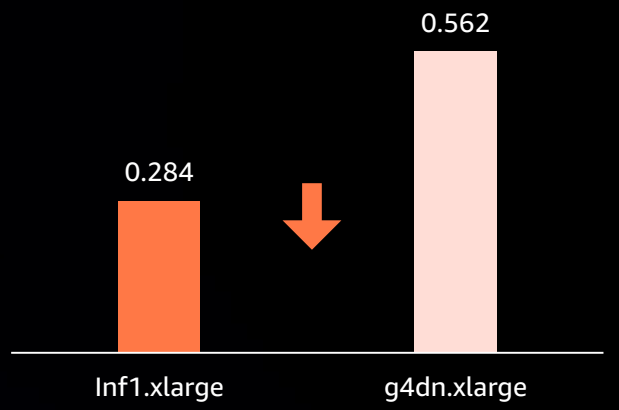
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

INF1 DELIVERS BEST PRICE/PERFORMANCE

Natural language:



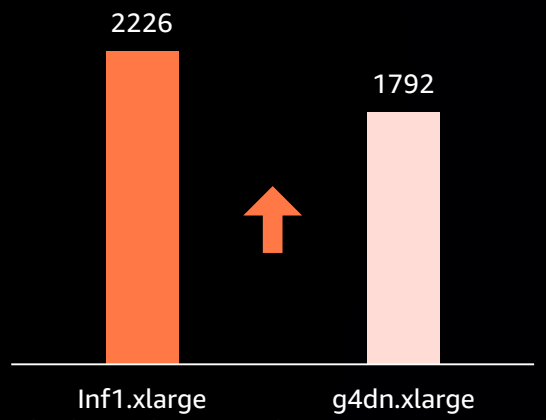
38%
Higher throughput than G4



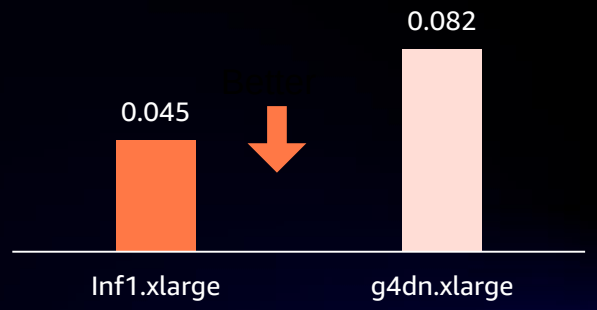
49%
Lower cost-per-inference than G4

Results based on running BERT-Large model end-to-end with TensorFlow, seqlen=128

Image classification:



24%
Higher throughput than G4



45%
Lower cost-per-inference than G4

Results based on running ResNet-50 model end-to-end with TensorFlow

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Bin-packing Models

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Problem Statement

- Given N_{models} with resource utilization
  - Memory
  - Compute

- Pack N_{models} into Minimum number of Bins (or) Instances
  - Each Instance has a Max Memory and Compute Limit

- Calculate Model & Instance Assignment Matrix

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Mathematically

- m_k: $Memory\ required\ for\ the\ model\ k, k = 1\ ..\ n_{models}$
- c_k: $Compute\ required\ for\ the\ model\ k, k = 1\ ..\ n_{models}$

- max GPU memory (M_{max}) and GPU utilization of (C_{max})

- x_{ij}: $\{0, 1\}\ to\ denote\ Model\ i\ in\ Bin\ j, i$: $[1\ ..\ n_{models}]$,j: $[1\ ..\ n_{models}]$
  y_j: $\{0, 1\}\ to\ denote\ Bin\ j\ is\ being\ used, j$: $[1\ ..\ n_{models}]$

Bin Packing Problem:
Implement and perform Combinatorial Optimization

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Formulation

Objective: $Minimize \sum_{j=0}^{n_{models}} y_j$

Ensure allocation of each item to exactly one bin: $\sum_{i=0}^{n_{models}} x_{ij} = 1$

Memory Capacity Constraint:

$\sum_{i=0}^{n_{models}} x_{ij} * m_i < y_j * M_{max}, for\ j = [1 .. n_{models}]$

Processor Capacity Constraint:

$\sum_{i=0}^{n_{models}} x_{ij} * c_i < y_j * C_{max}, for\ j = [1 .. n_{models}]$

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Demonstration (Memory & Compute Constraint)

Generate Memory and Compute

Num Models

68

30 100

Memory Capacity (MB)

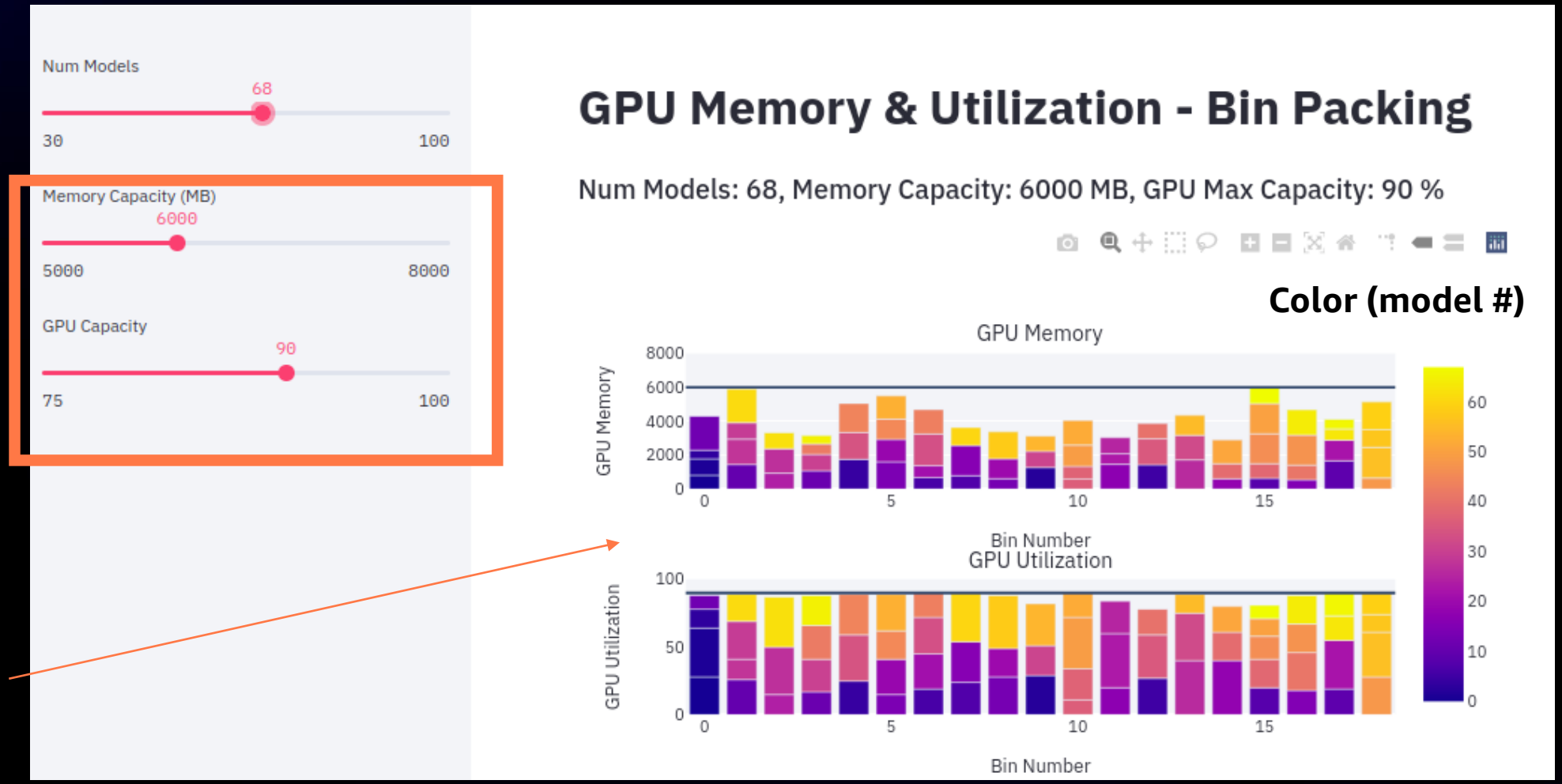
6000

5000 8000

GPU Capacity

90

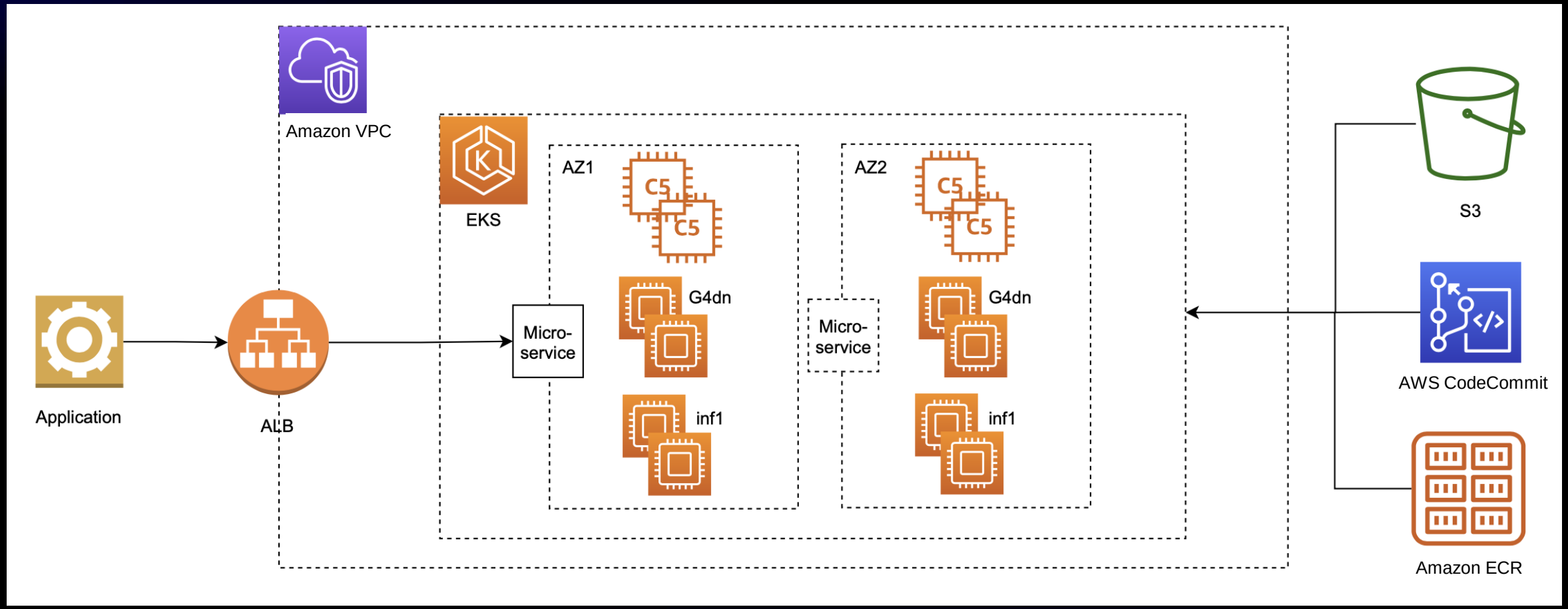
75 100



Compute Constrains is clearly observed

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Infrastructure



Micro service pod and Model pods running on the cluster nodes.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Customer Examples

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Inference @ Scale for Salesforce

- Serve 3,000+ models for inference on AWS
- Each model needs to be accessible through an API endpoint
- < 100ms Latency

- Packed multiple models in each GPU/Inferentia instance
- Reduced cost over **15x**
  - $1578/hr in a traditional deployment to under $100/hr

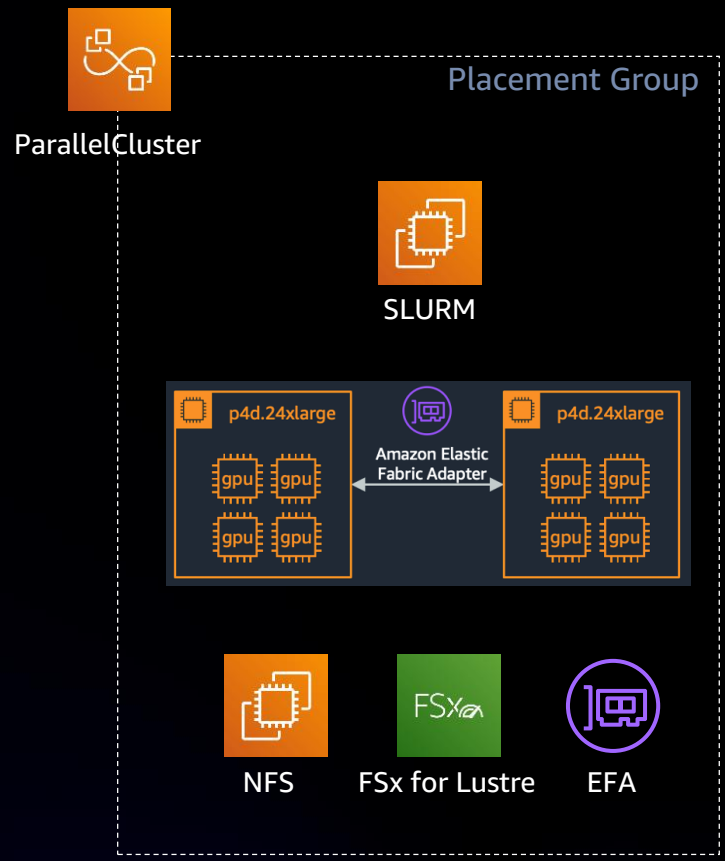
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

LARGE-SCALE NLU RESEARCH ON AWS

Strategic customer actively researching on NLP/NLU at scale
Large scale ML system in production since 2019
Started with a 100+ p3dn cluster in 2019, added 260+ P4ds in 2020
Expanded to 850+ P4ds, with plans to scale to 1100+ in 2021
Single spine in a placement group for the lowest latency
Periodically adds additional instances on-demand for special workloads

Key infrastructural elements of the system:

- ParallelCluster for cluster management
- SLURM to schedule jobs
- FSx for Lustre
- EFA to support fast inter-node communications at scale
- 850+ nodes of P4d and growing
- Custom DLAMI built off of AWS base DLAMI
- PyTorch + DDP + NCCL



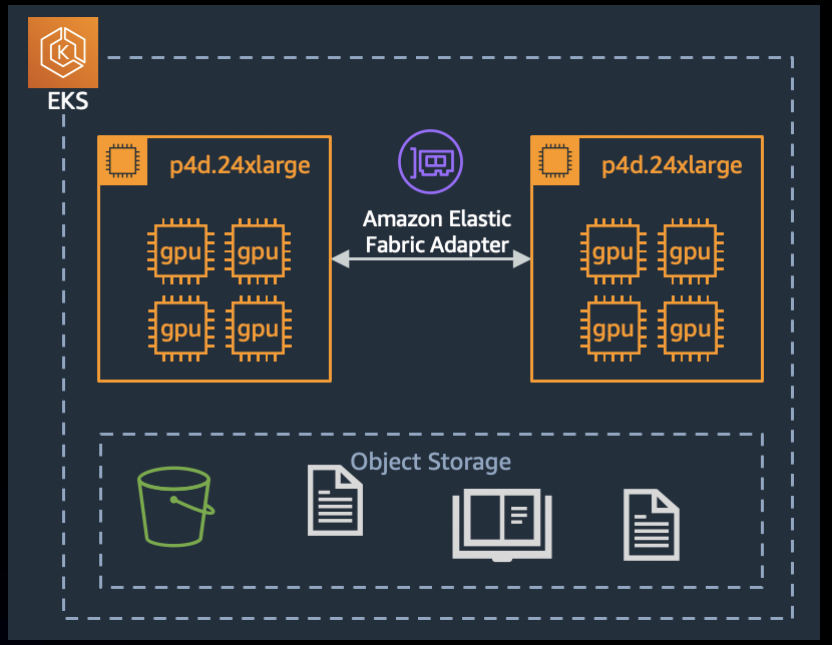
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

LARGE-SCALE NLP STARTUP ON AWS

CA startup aims to build the world's largest NLP model (2T+ parameters)
Decided to run on AWS after a POC to train a 50B parameter model across 200 P4ds (800 A100 GPUs)
Training workload is now in production across 875+ P4ds (7000 GPUs)
Runs on EC2 with EKS orchestration and EFA for networking speedup
Deployed in a single AZ, looking to extend architecture to span AZs.

Key infrastructural elements of the system:

- EKS for orchestration
- EFA to support fast inter-node communications at scale
- 875+ nodes of P4d and growing
- S3 storage layer
- Custom DLAMI built off of AWS base DLAMI
- PyTorch + DDP + NCCL



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

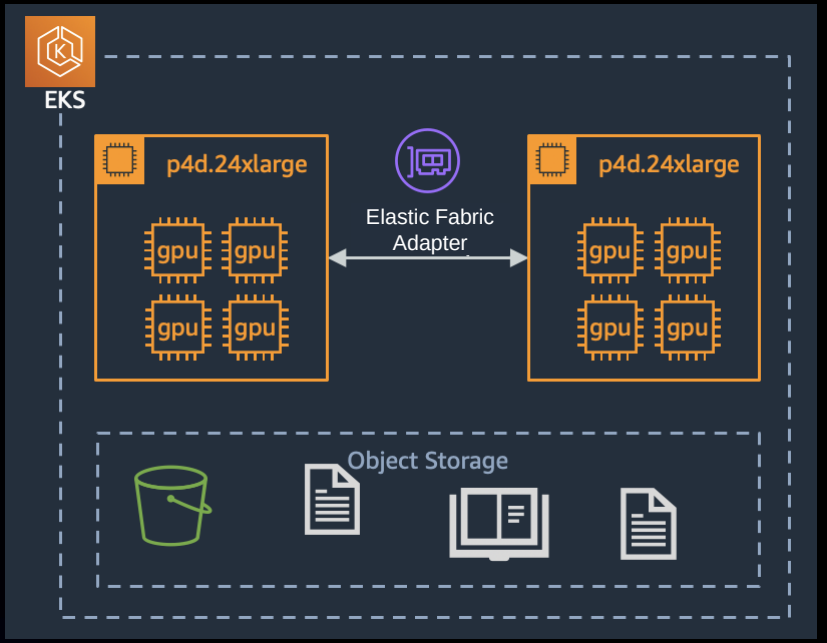
LARGE-SCALE NLP STARTUP ON AWS

Israel startup trains large NLP models and builds apps based these models
Trained a 178B parameter model (larger than GPT-3) across 104 P4ds (832 GPUs)
Launched in production after a POC to train a smaller model across 36 P4ds
Being one of the early P4d customers, they used Spot instances
Trained with dataset containing 300 billion tokens
Applications powered: chatbots, writing (rephrase, sentence completion)

Key infrastructural elements of the system:

for orchestration

EFA to support fast inter-node communications at scale
104 nodes of P4d, used Spot
S3 storage layer
Custom DLAMI built off of AWS base DLAMI
PyTorch

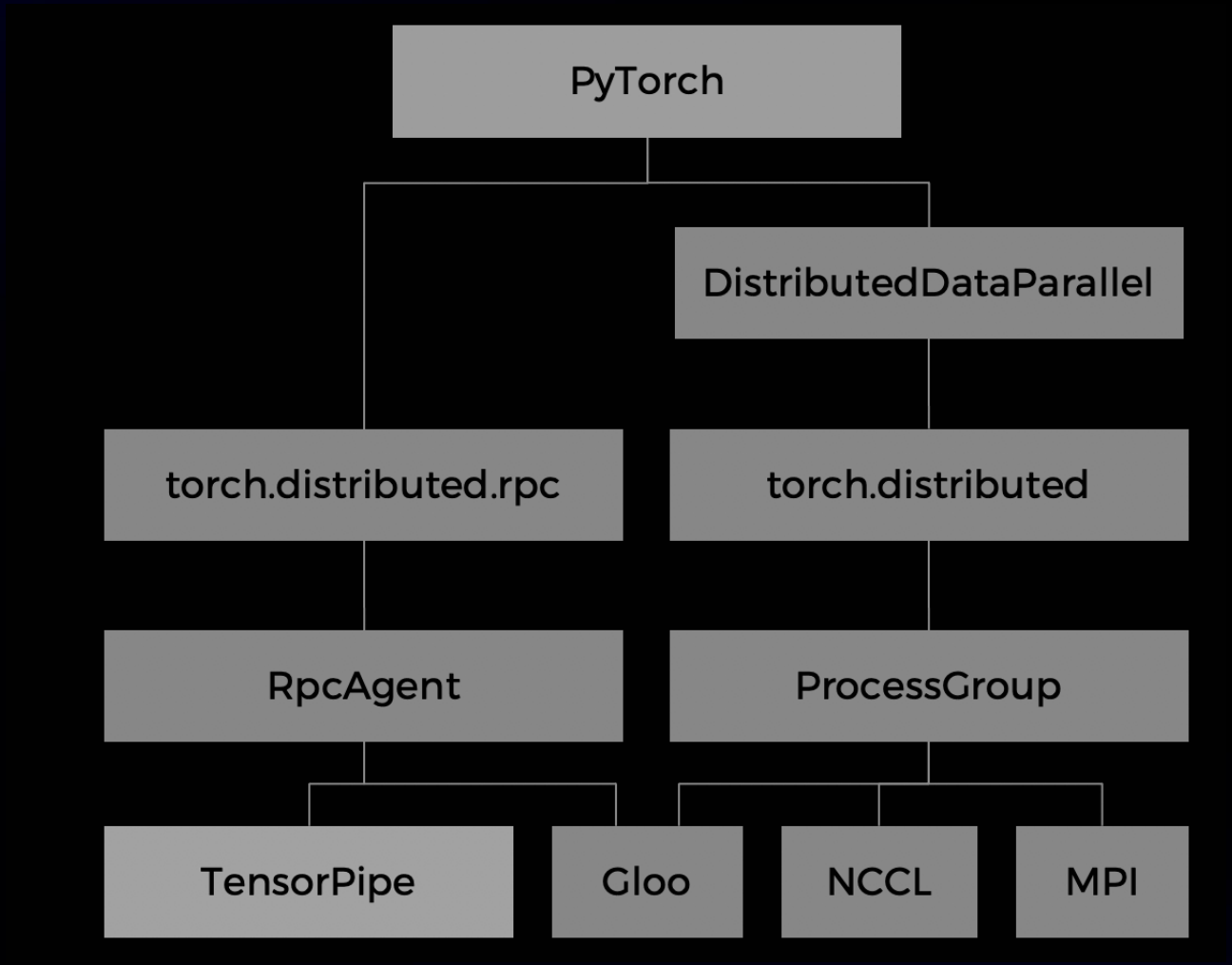


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling PyTorch

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Torch distributed stack



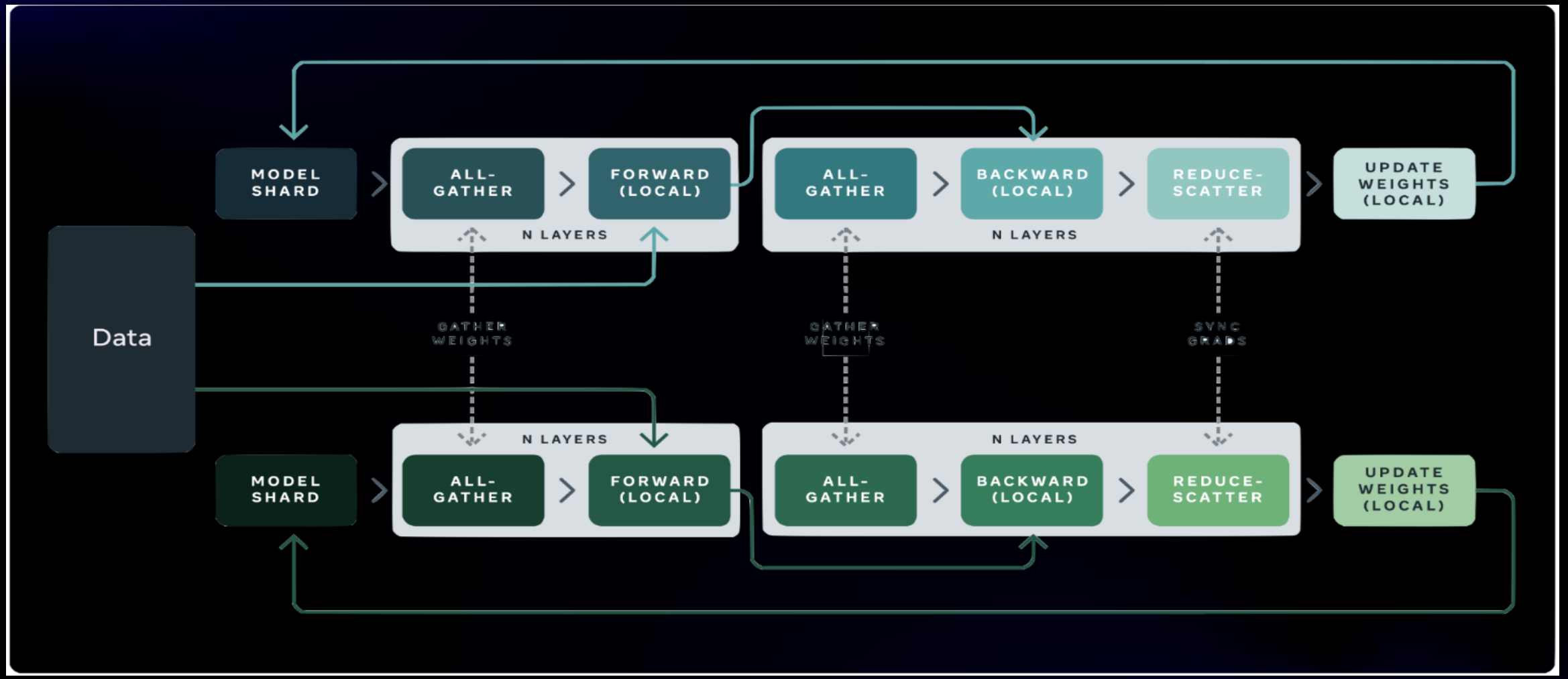
https://pytorch.org/docs/stable/distributed.html

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fully sharded data parallel



https://engineering.fb.com/2021/07/15/open-source/fsdp/

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ZeroRedundancyOptimizer

Shard optimizer states and model weights

```
os.environ['MASTER_ADDR'] = 'localhost'
os.environ['MASTER_PORT'] = '29500'

# create default process group
dist.init_process_group("gloo", rank=rank, world_size=world_size)

# create local model
model = nn.Sequential(*[nn.Linear(2000, 2000).to(rank) for _ in range(20)])

# construct DDP model
ddp_model = DDP(model, device_ids=[rank])
print_peak_memory("Max memory allocated after creating DDP", rank)

# define loss function and optimizer
loss_fn = nn.MSELoss()

optimizer = ZeroRedundancyOptimizer(
    ddp_model.parameters(),
    optimizer_class=torch.optim.Adam,
    lr=0.01
    )
```

https://pytorch.org/docs/master/distributed.optim.html

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Torch RPC

- Optimized Tensor communications (TensorPipe)
- Remote memory management
- Distributed autograd

```
>>> export MASTER_ADDR=localhost
>>> export MASTER_PORT=5678

>>> # On worker 0:
>>> import torch
>>> import torch.distributed.rpc as rpc
>>> rpc.init_rpc("worker0", rank=0, world_size=2)
>>> ret = rpc.rpc_sync("worker1", torch.add, args=(torch.ones(2), 3))
>>> rpc.shutdown()

>>> # On worker 1:
>>> import torch.distributed.rpc as rpc
>>> rpc.init_rpc("worker1", rank=1, world_size=2)
>>> rpc.shutdown()
```

https://pytorch.org/docs/stable/rpc.html

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Power SGD

- DDP communication hook
- Up to 10x speedup in DDP
- NCCL, Gloo, and MPI support

```
import torch.distributed.algorithms.ddp_comm_hooks.powerSGD_hook as powerSGD

ddp_model = nn.parallel.DistributedDataParallel(
    module=model,
    device_ids=[rank],
)

state = PowerSGD.PowerSGDState(
  process_group=process_group,
  matrix_approximation_rank=1,
  start_powerSGD_iter=1_000,
)
ddp_model.register_comm_hook(state, PowerSGD.powerSGD_hook)
```

https://medium.com/pytorch/accelerating-pytorch-ddp-by-10x-with-powersgd-585aef12881d

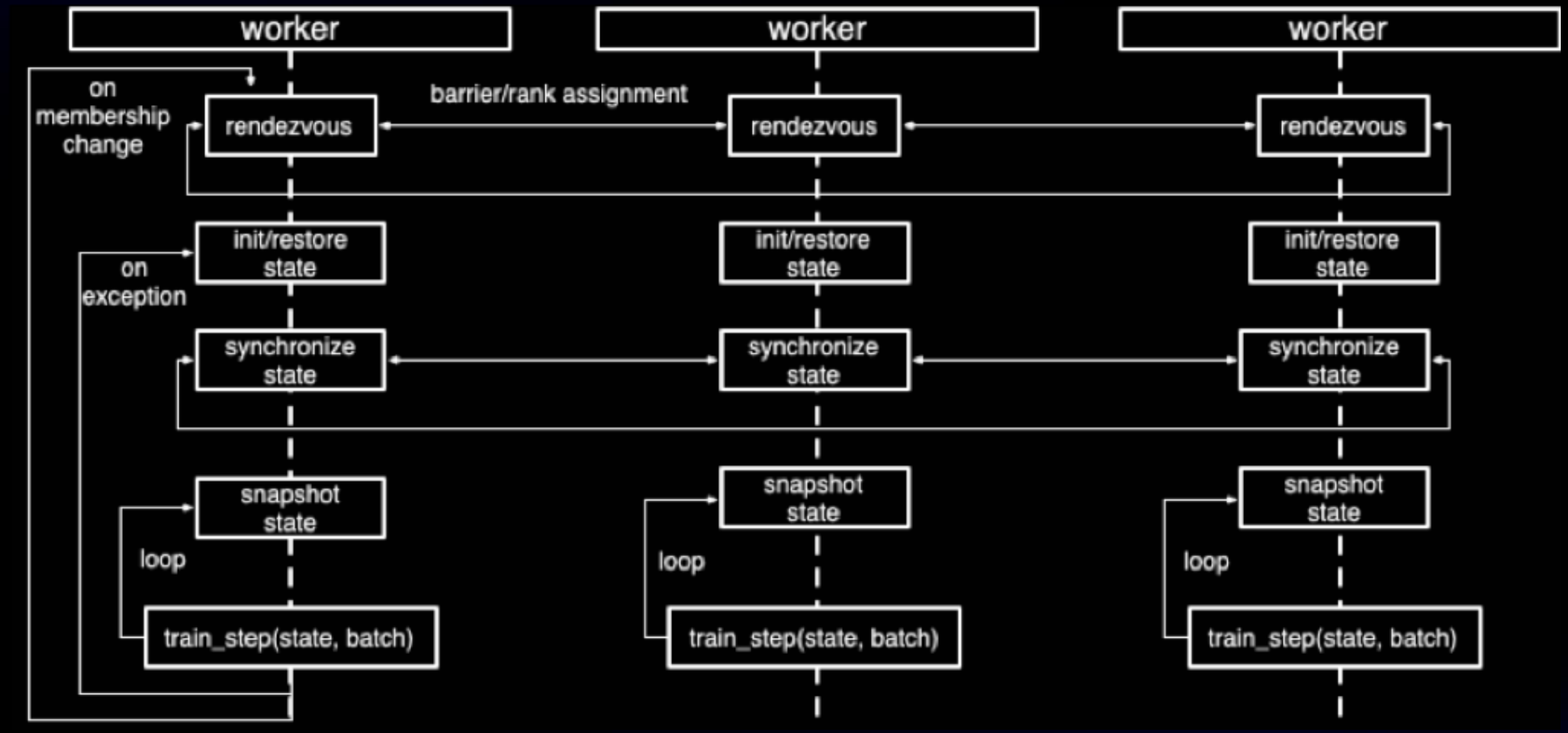
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Torch Elastic

Auto scaling fault tolerant distributed PyTorch jobs on Spot Instances



https://pytorch.org/docs/stable/distributed.elastic.html

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

PipeTransformer

Elastic resource scheduling for frozen layers

Table 1. Speedup for ViT and BERT Training

Dataset	Baseline		PipeTransformer		Training Speedup
	Accuracy	Training time	Accuracy	Training time	
ImageNet	80.83 ± 0.05	26h 30m	82.18 ± 0.32	9h 21m	**2.83** ×
CIFAR-100	91.21 ± 0.07	35m 6s	91.33 ± 0.05	12m 23s	2.44 ×
SQuAD 1.1	90.71 ± 0.18	5h 7m	90.69 ± 0.23	2h 26m	2.10 ×

```python
# Step 1: build a model including two linear layers
fc1 = nn.Linear(16, 8).cuda(0)
fc2 = nn.Linear(8, 4).cuda(1)

# Step 2: wrap the two layers with nn.Sequential
model = nn.Sequential(fc1, fc2)

# Step 3: build Pipe (torch.distributed.pipeline.sync.Pipe)
model = Pipe(model, chunks=8)

# do training/inference
input = torch.rand(16, 16).cuda(0)
output_rref = model(input)
```

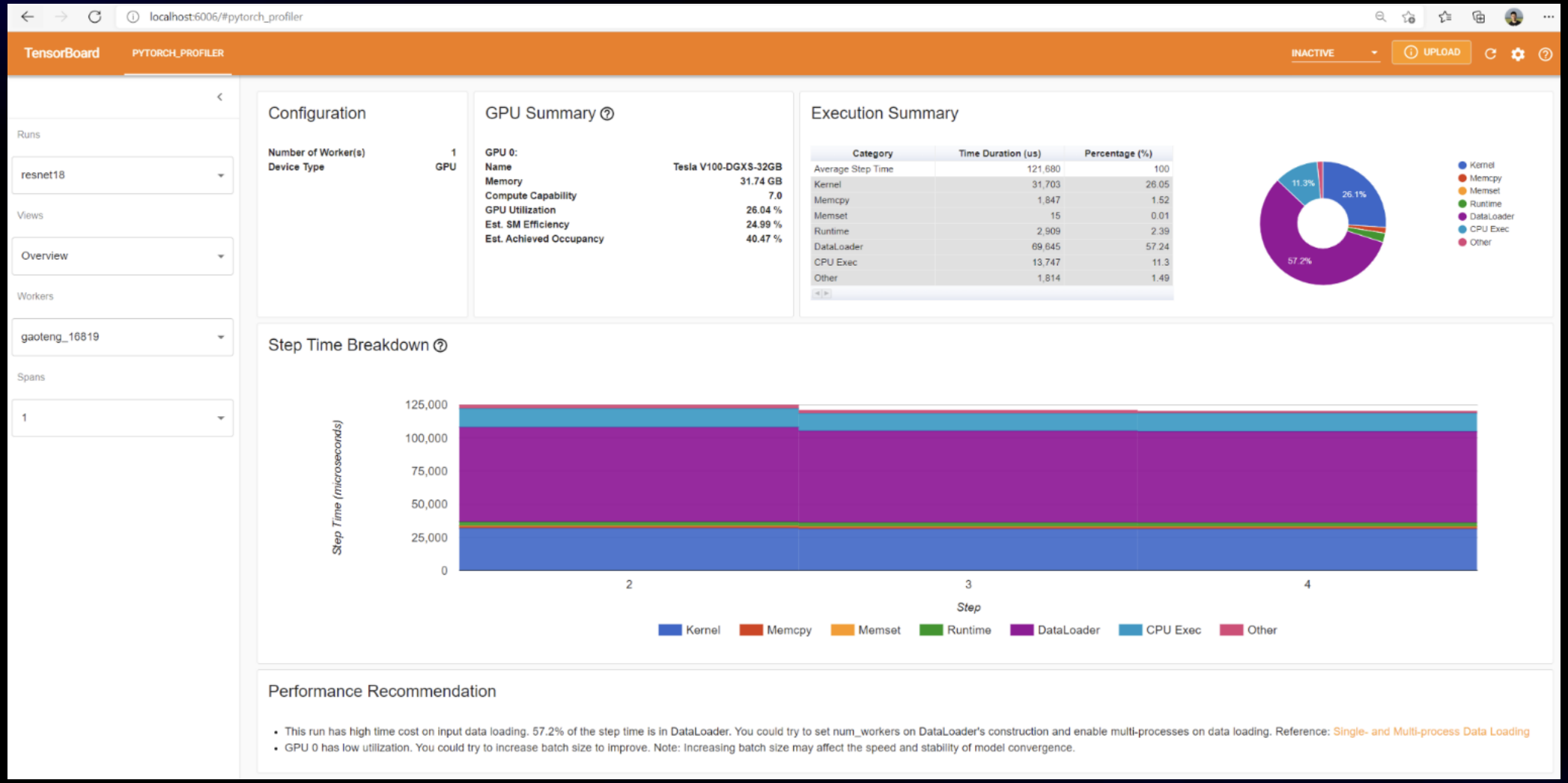
https://pytorch.org/blog/pipetransformer-automated-elastic-pipelining/

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Torch Profiler



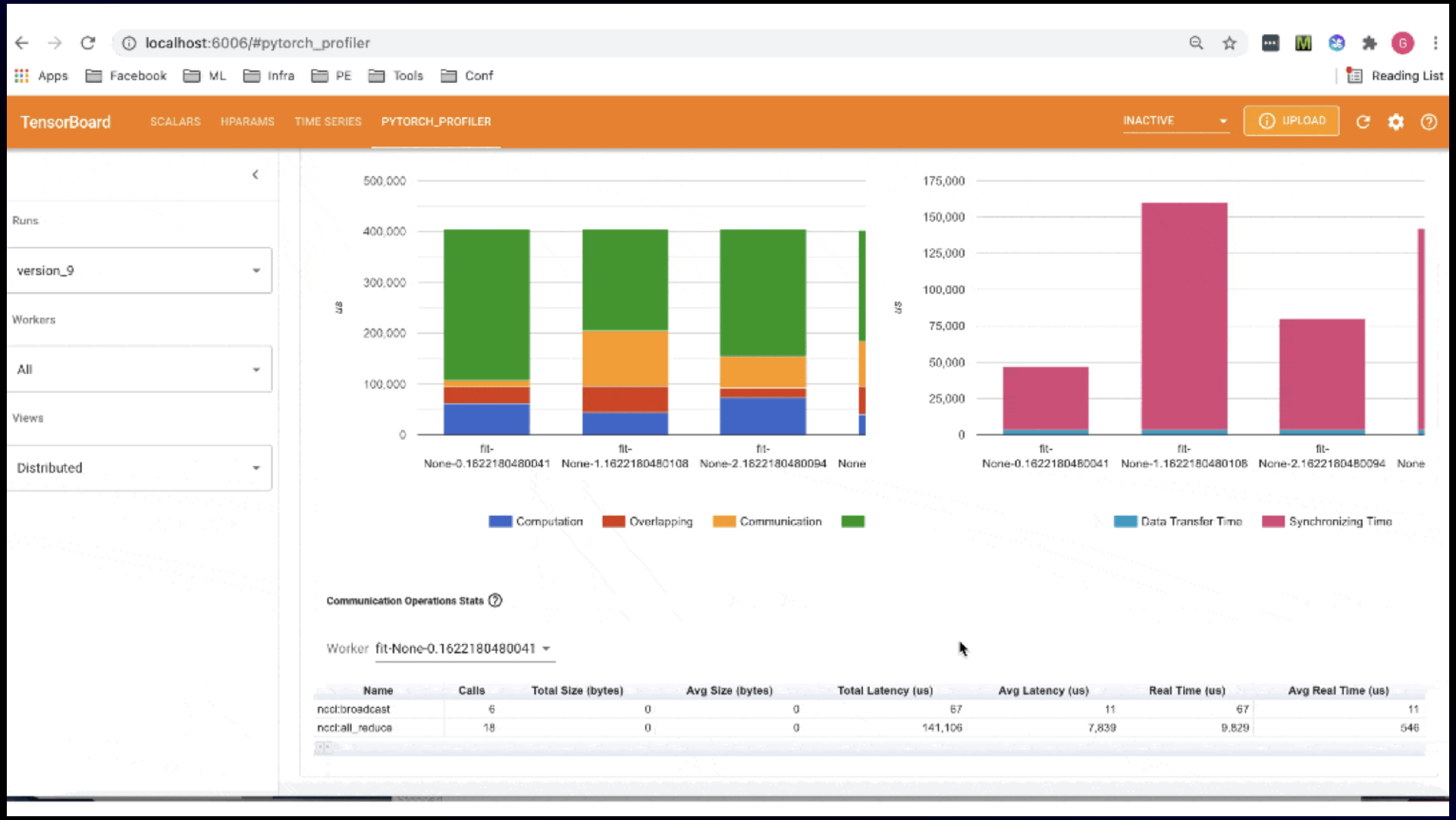
https://pytorch.org/blog/introducing-pytorch-profiler-the-new-and-improved-performance-tool/

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Distributed profiling



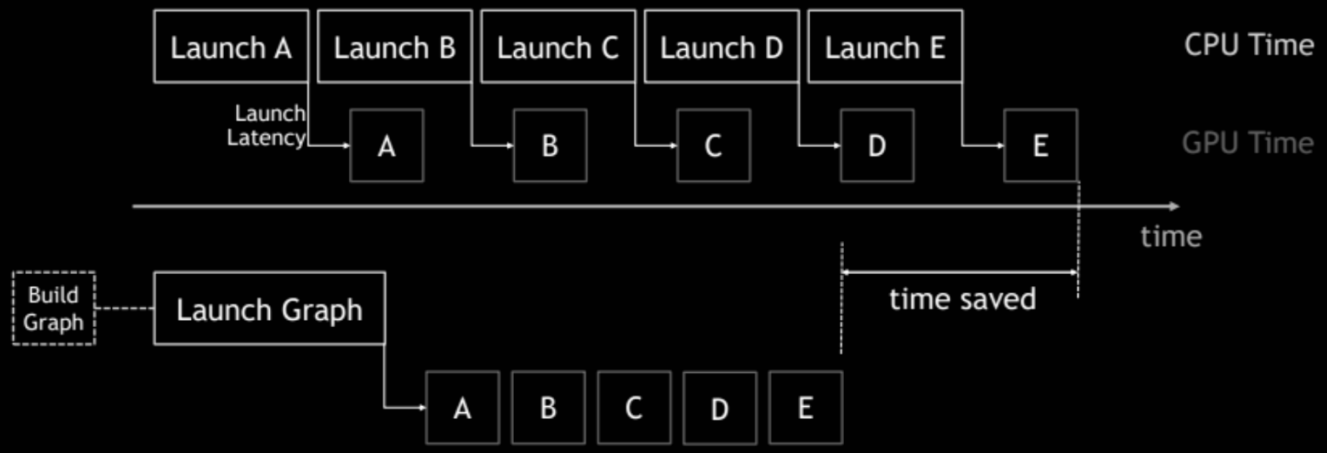
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

https://pytorch.org/blog/pytorch-profiler-1.9-released/

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CUDA Graphs

Coalesce small CUDA kernels




```python
N, D_in, H, D_out = 640, 4096, 2048, 1024

module1 = torch.nn.Linear(D_in, H).cuda()
module2 = torch.nn.Linear(H, D_out).cuda()
module3 = torch.nn.Linear(H, D_out).cuda()

loss_fn = torch.nn.MSELoss()
optimizer = torch.optim.SGD(chain(module1.parameters() +
                                  module2.parameters() +
                                  module3.parameters()),
                            lr=0.1)

# Sample inputs used for capture
# requires_grad state of sample inputs must match
# requires_grad state of real inputs each callable will see.
x = torch.randn(N, D_in, device='cuda')
h = torch.randn(N, H, device='cuda', requires_grad=True)

module1 = torch.cuda.make_graphed_callables(module1, (x,))
module2 = torch.cuda.make_graphed_callables(module2, (h,))
module3 = torch.cuda.make_graphed_callables(module3, (h,))

real_inputs = [torch.rand_like(x) for _ in range(10)]
real_targets = [torch.randn(N, D_out, device="cuda") for _ in range(10)]

for data, target in zip(real_inputs, real_targets):
    optimizer.zero_grad(set_to_none=True)

    tmp = module1(data)  # forward ops run as a graph

    if tmp.sum().item() > 0:
        tmp = module2(tmp)  # forward ops run as a graph
    else:
        tmp = module3(tmp)  # forward ops run as a graph

    loss = loss_fn(tmp, y)
    # module2's or module3's (whichever was chosen) backward ops,
    # as well as module1's backward ops, run as graphs
    loss.backward()
    optimizer.step()
```

https://pytorch.org/blog/accelerating-pytorch-with-cuda-graphs

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

torch.fx

Compiler transformations in Python

- Operator fusion
- Export to runtime
- TorchVision Feature Extraction
- Quantization

```python
import torch
import torch.fx

# Sample module
class M(torch.nn.Module):
    def forward(self, x, y):
        return torch.add(x, y)

def transform(m: torch.nn.Module,
              tracer_class : type = fx.Tracer) -> torch.nn.Module:
    graph : fx.Graph = tracer_class().trace(m)
    # FX represents its Graph as an ordered list of
    # nodes, so we can iterate through them.
    for node in graph.nodes:
        # Checks if we're calling a function (i.e:
        # torch.add)
        if node.op == 'call_function':
            # The target attribute is the function
            # that call_function calls.
            if node.target == torch.add:
                node.target = torch.mul

    graph.lint() # Does some checks to make sure the
                 # Graph is well-formed.

    return fx.GraphModule(m, graph)
```

https://pytorch.org/docs/stable/fx.html

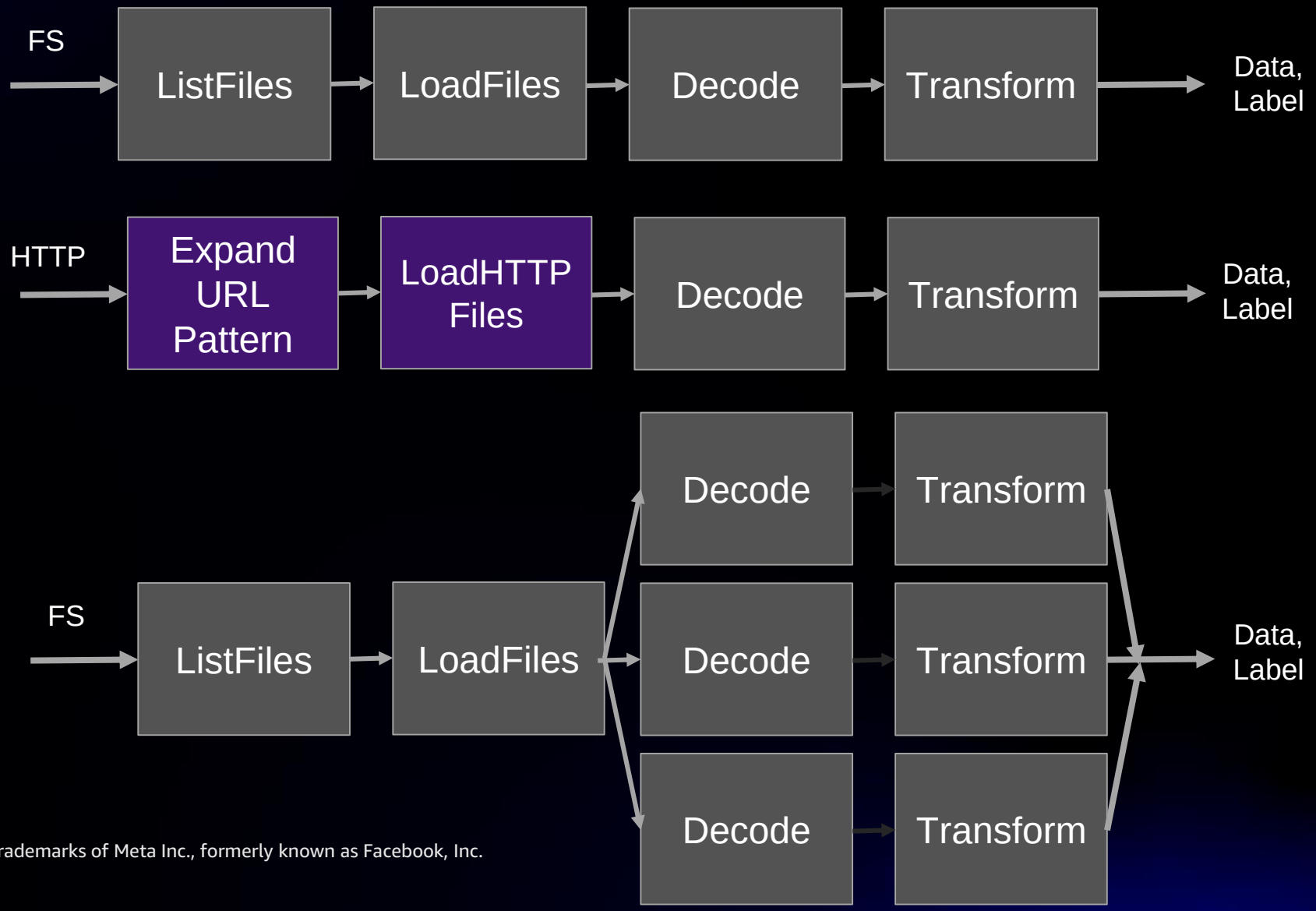
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Torch data

- Modular versus monolithic datasets
- Interchangeable data pipes
- Composable multi-threading and multi-processing



https://github.com/pytorch/data

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

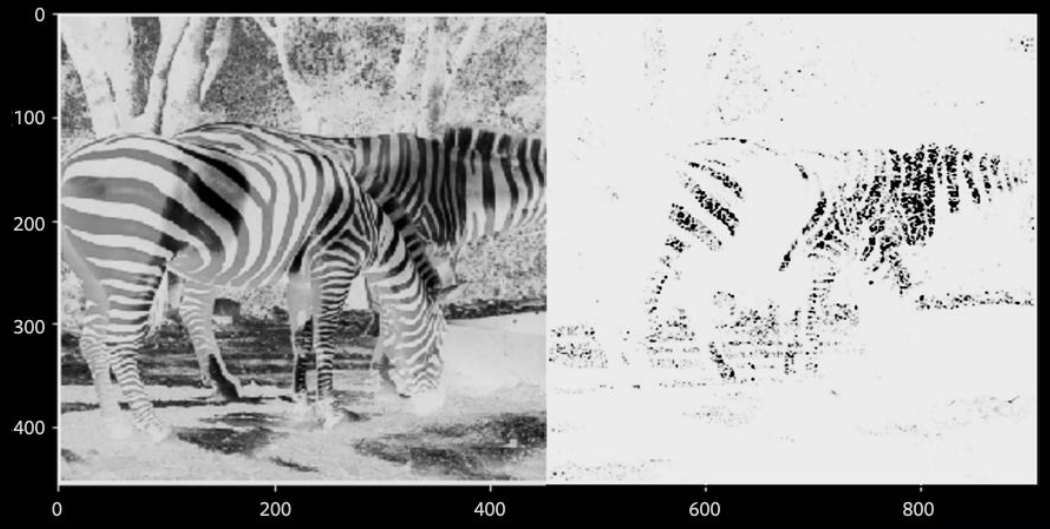
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CAPTUM

Interpretability and layer attribution

Target Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
zebra	zebra (0.60)	zebra	7.54	what is on the picture




```
from captum.attr import Saliency
from captum.attr import visualization as viz

attr_algo = Saliency(cifar_net)
original_image = np.transpose((image.cpu().detatch().numpy()/2) + 0.5, (1,2,0))

# compute attributions for plane w.r.t inputs
grads = attr_algo.attribute(image, target = plane_label_ind)

viz.visualize_image_attr(grads,
                         original_image,
                         method="blended_heat_map",
                         sign="absolute_value",
                         show_colorbar=True,
                         title="Overlayed Gradient Magnitudes")
```

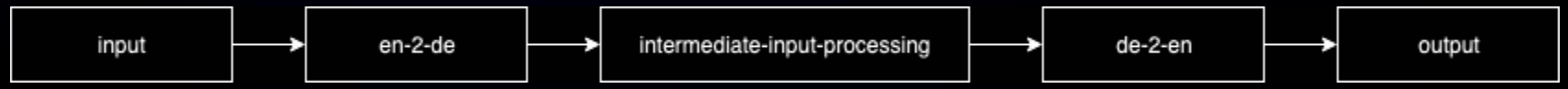
https://captum.ai/api/saliency.html

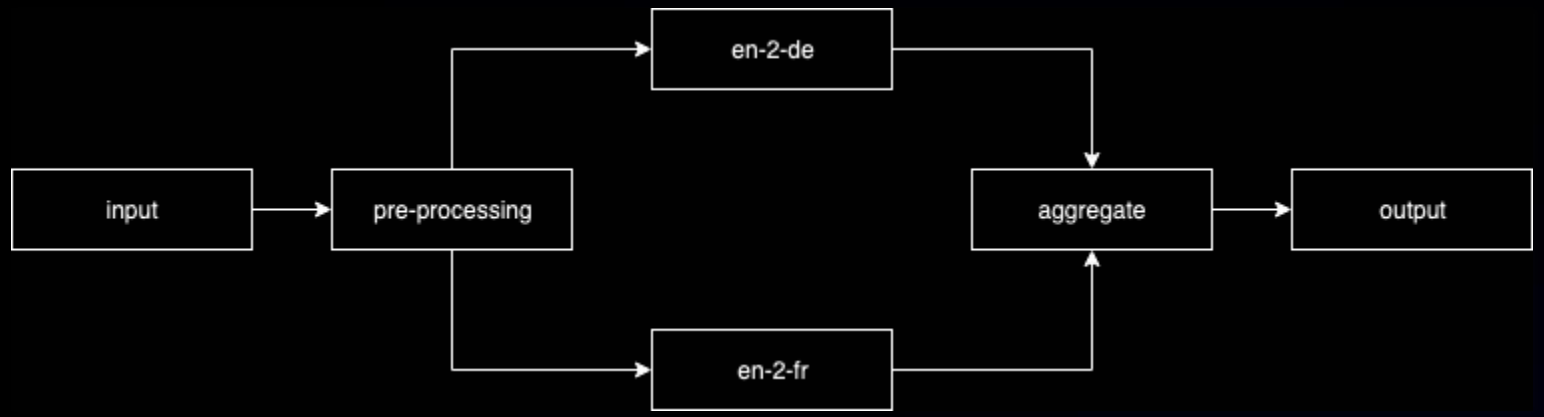
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

TorchServe ensemble






```
models:
    min-workers: 1
    max-workers: 1
    batch-size: 4
    max-batch-delay: 100
    retry-attempts: 5
    timeout-ms: 300000

    nmt_en_de:
      url: TransformerEn2De.mar

    nmt_de_en:
      url: TransformerDe2En.mar

dag:
  nmt_en_de: [prep_intermediate_input]
  prep_intermediate_input: [nmt_de_en]
  nmt_de_en: [post_processing]
```

https://pytorch.org/serve/workflows.html

PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Arun Subramaniyan

Arunaws@amazon.com

Mark Saroufim

Marksaroufim@fb.com
twitter.com /marksaroufim
Github.com/msaroufim

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.