AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

AIM404-R2

Deep learning applications with PyTorch

Kris Skrinak
Global Machine Learning Segment Technical Lead
AWS

Hamid Shojanazeri
AI Partner Engineer
PyTorch

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

Agenda

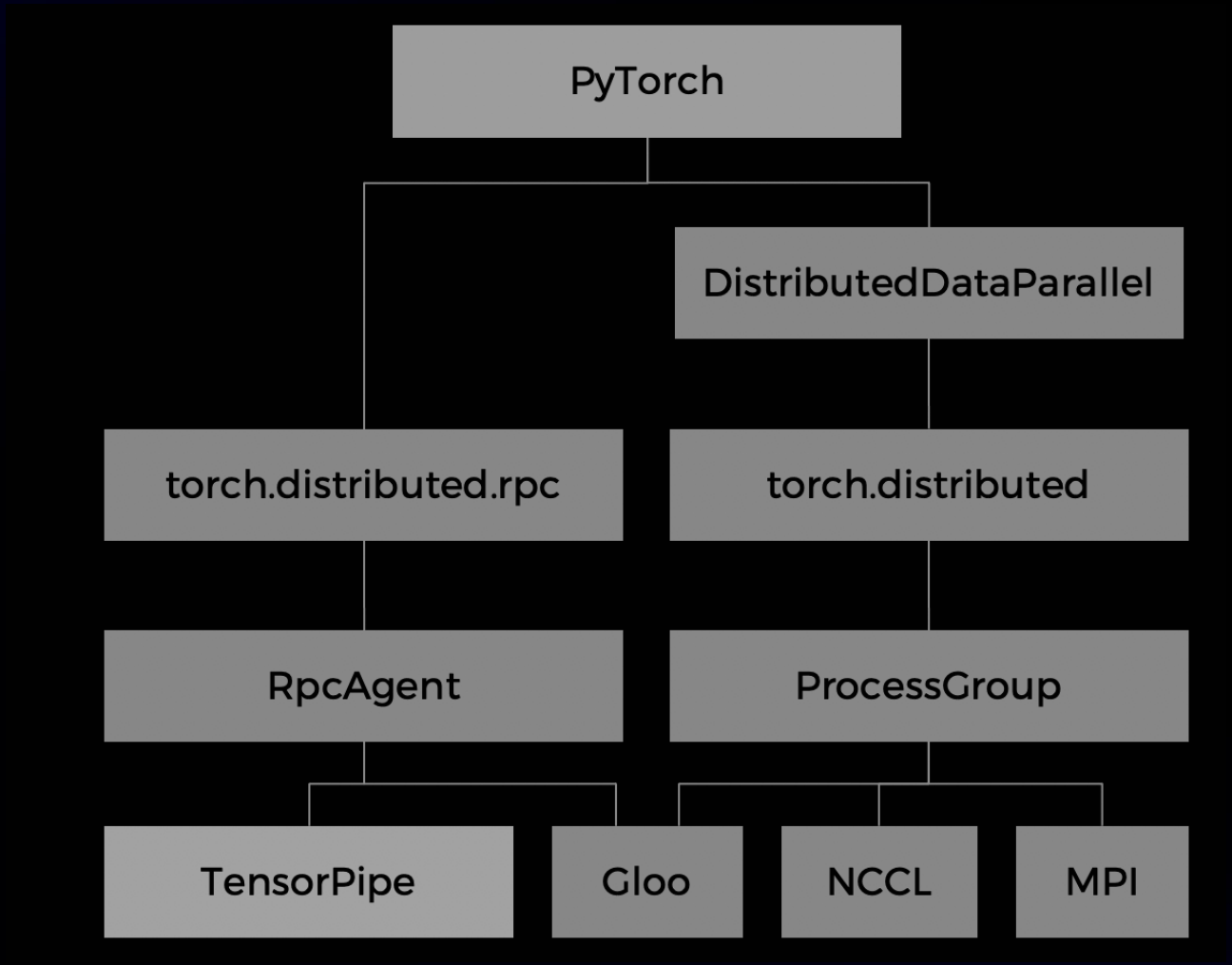
- What's new in PyTorch
- Running PyTorch models as container images in AWS Lambda
- Deploying multiple models for inference on AWS Lambda and EFS
- Amazon SageMaker Asynchronous Inference
- Amazon S3 plugin for PyTorch
- HuggingFace SageMaker Integration

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

Scaling PyTorch

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

Torch distributed stack



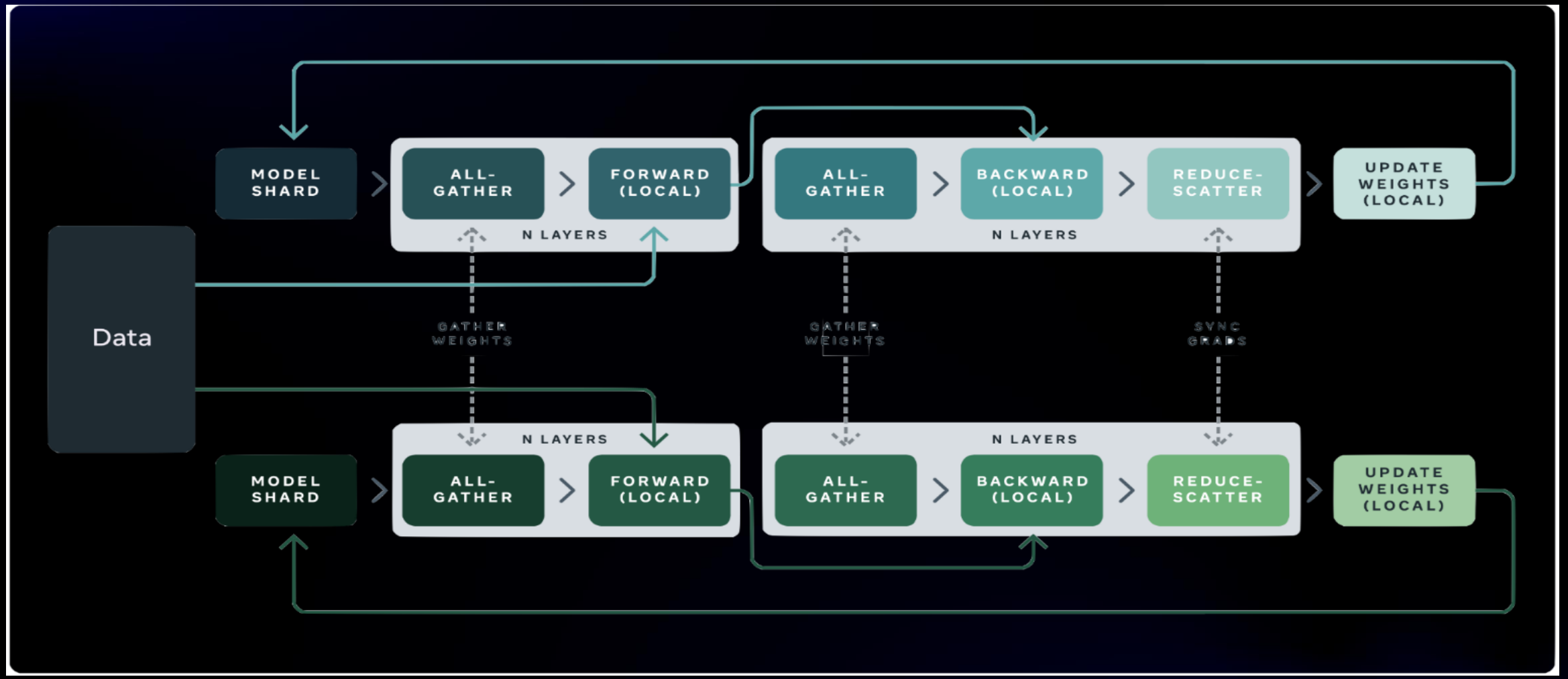
https://pytorch.org/docs/stable/distributed.html

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

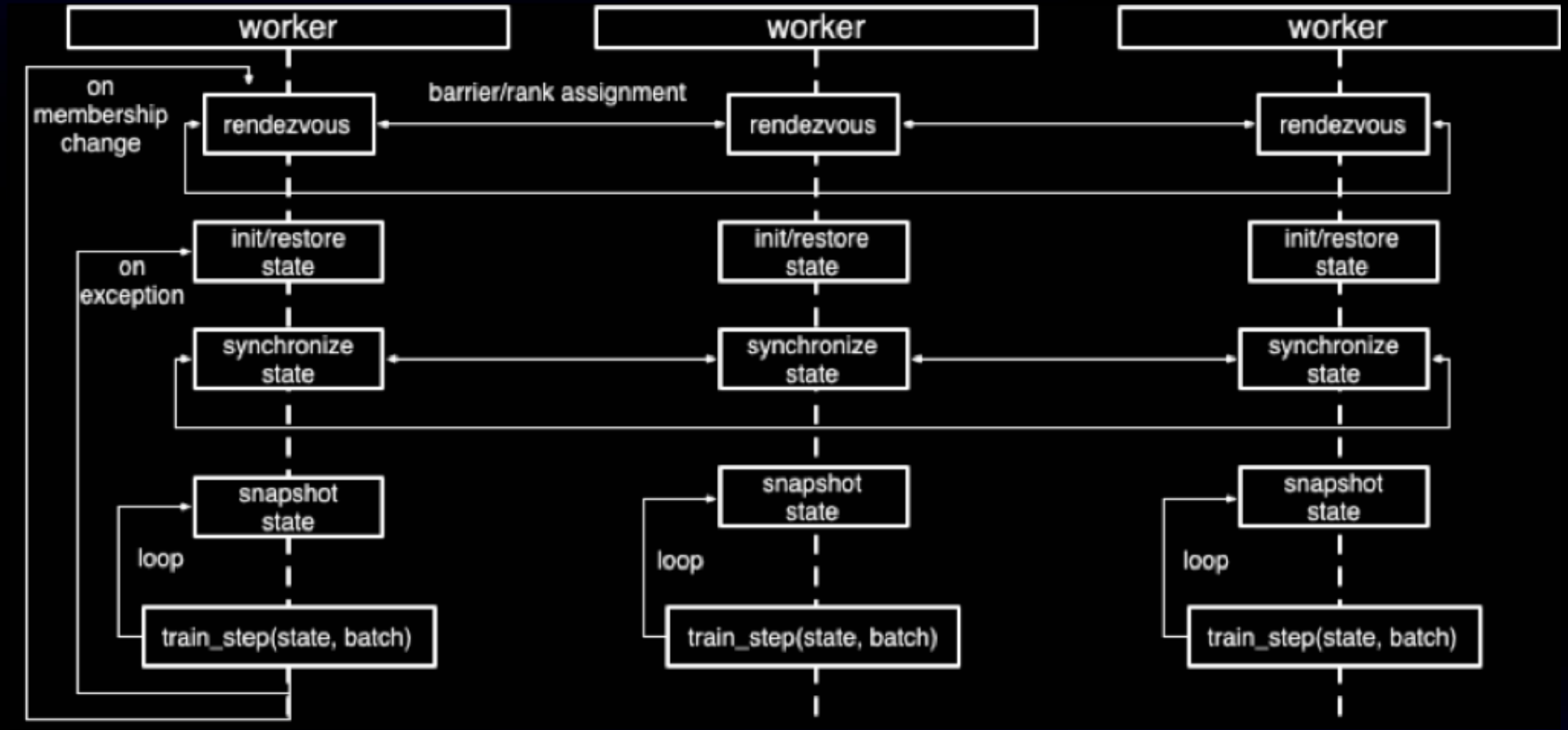
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

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)
```

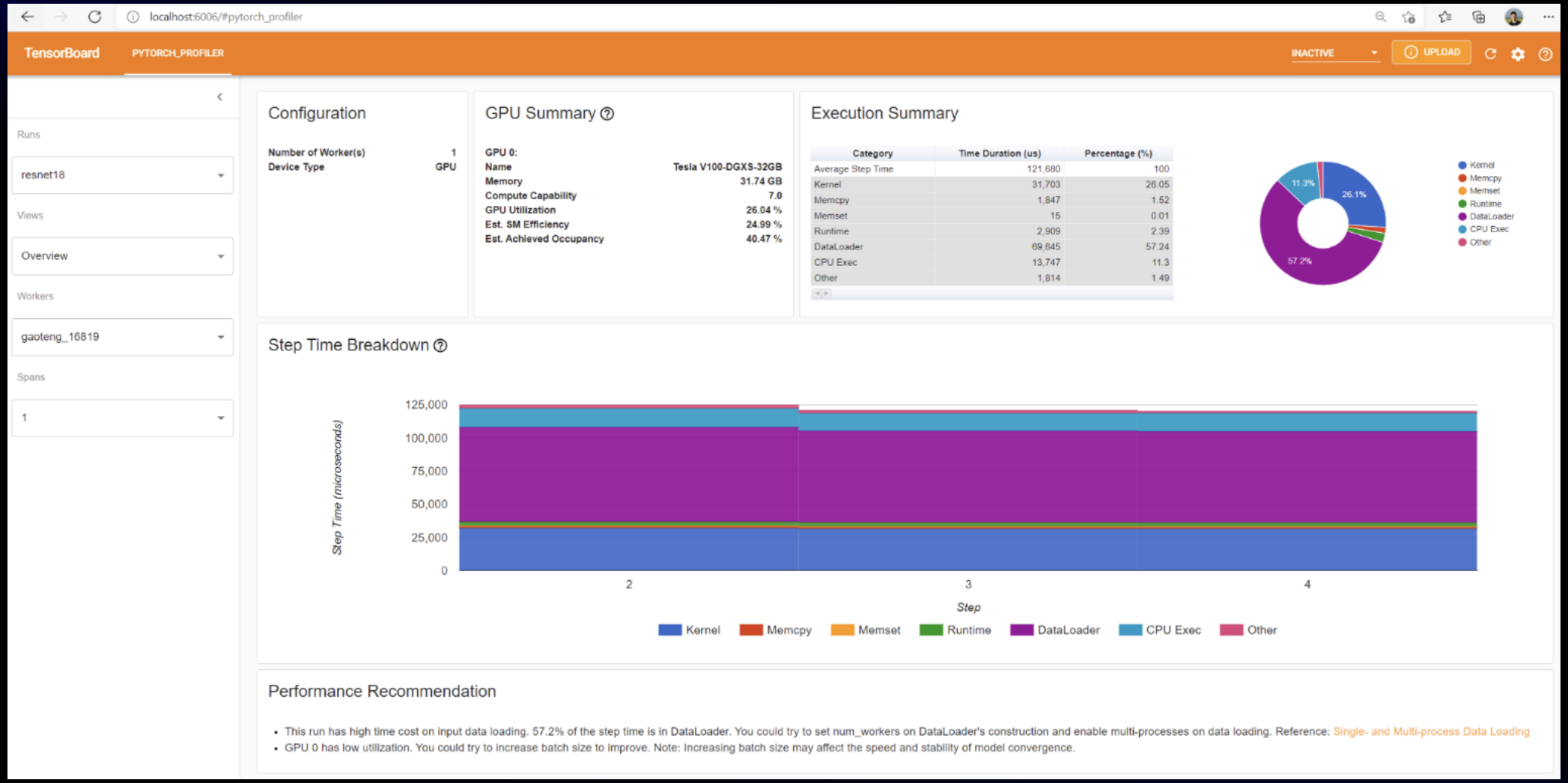
https://pytorch.org/blog/pipetransformer-automated-elastic-pipelining/

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



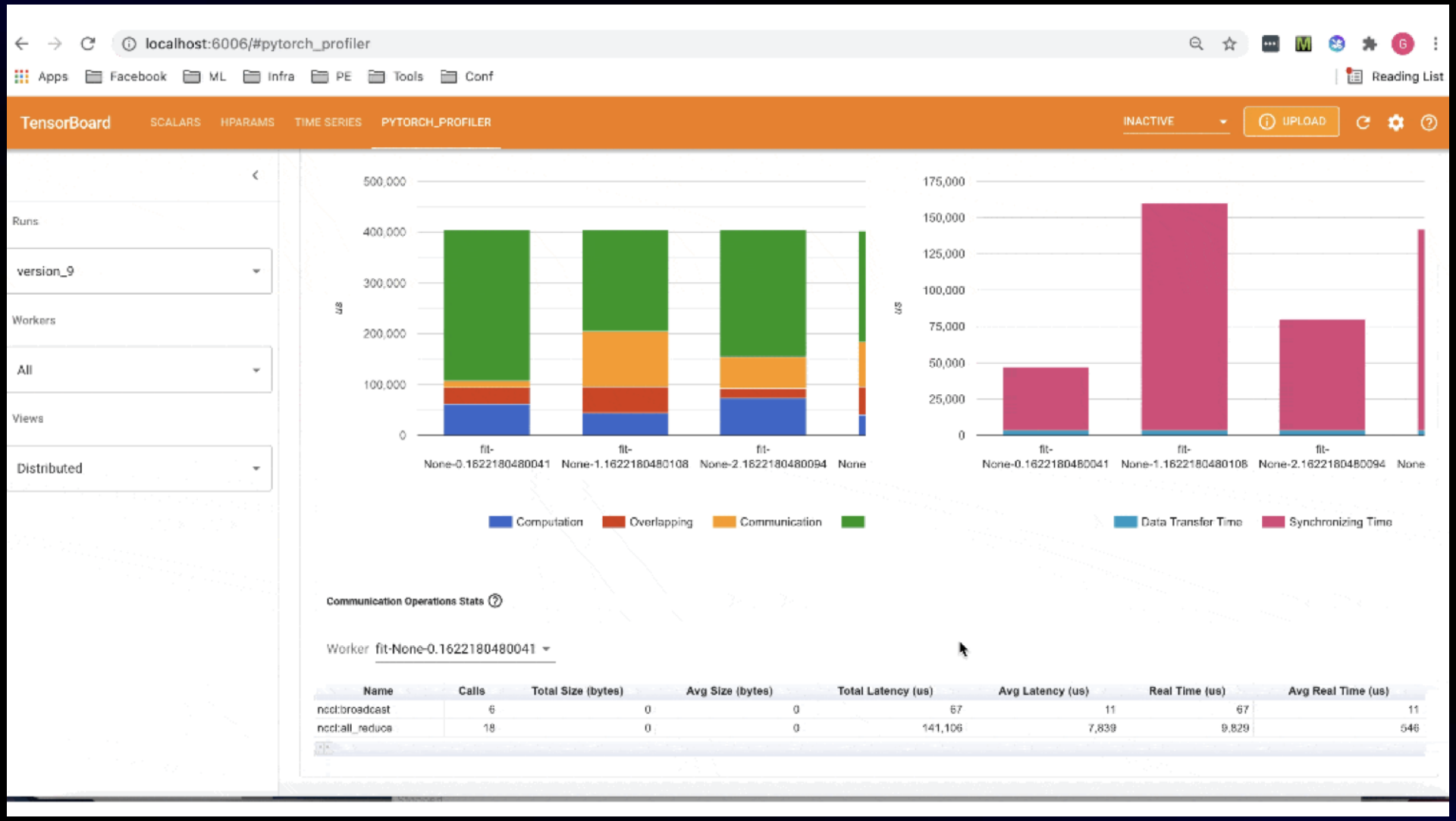
https://pytorch.org/blog/introducing-pytorch-profiler-the-new-and-improved-performance-tool/

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



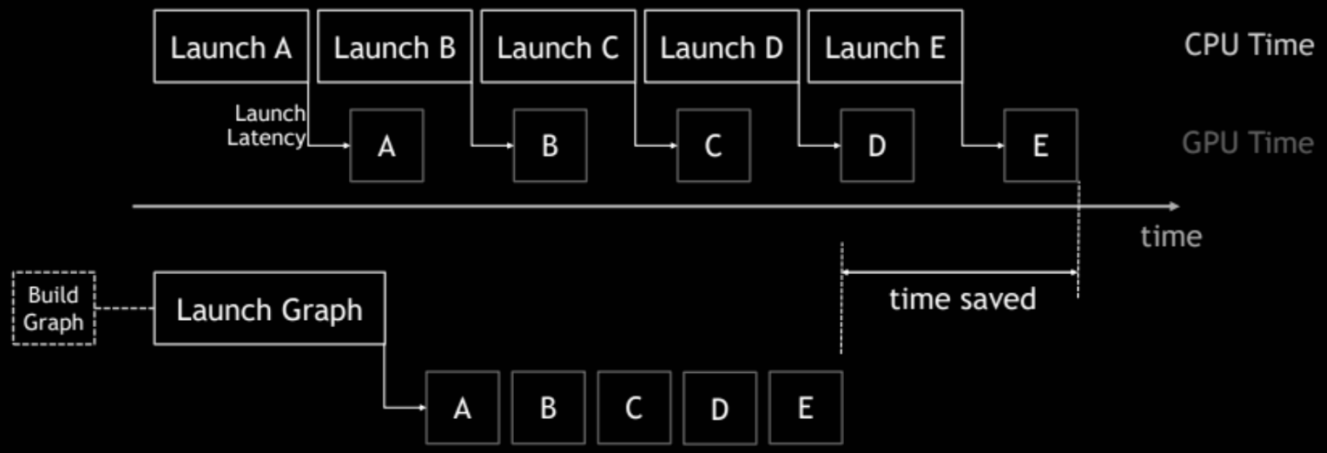
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

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




```
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

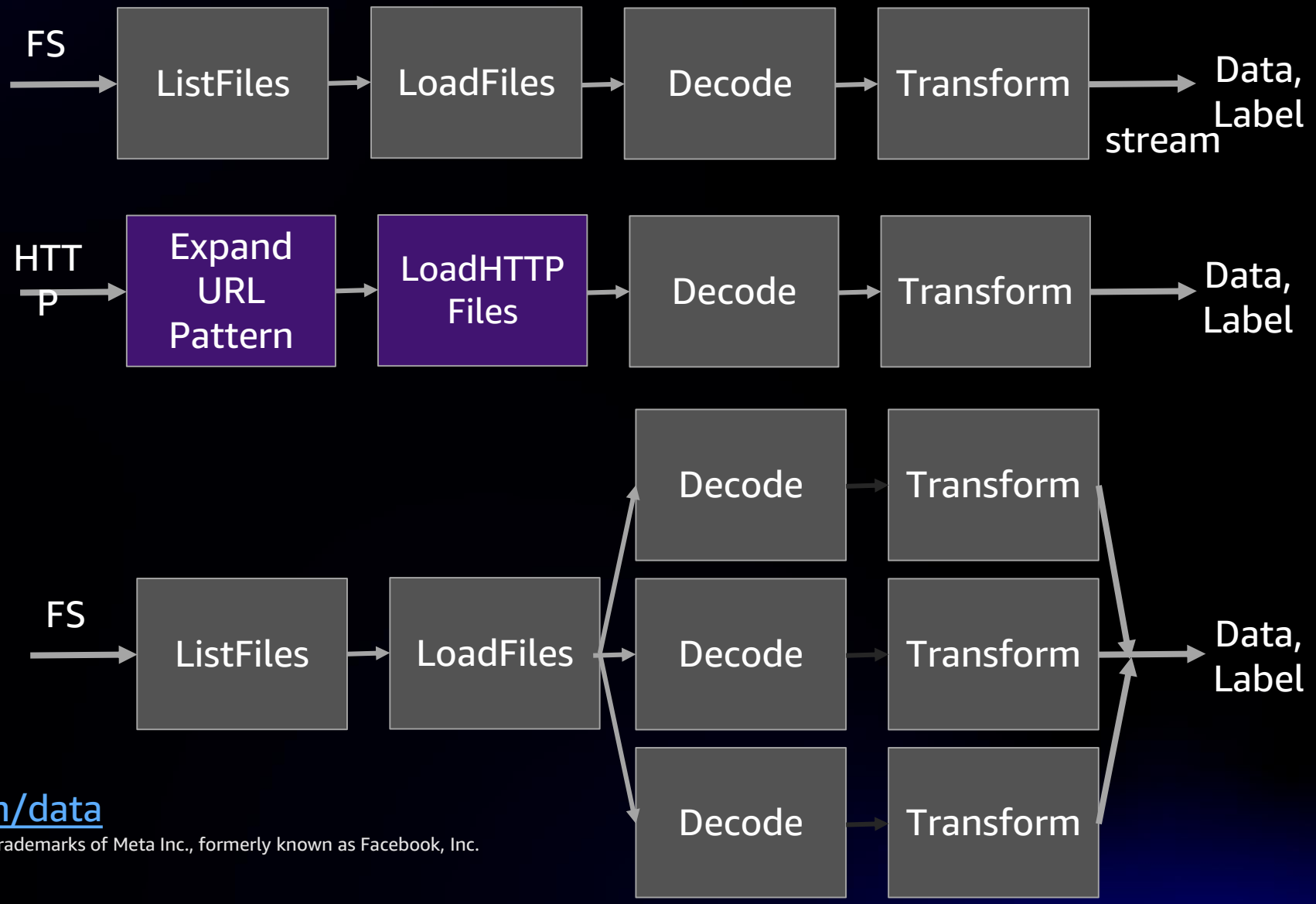
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

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.

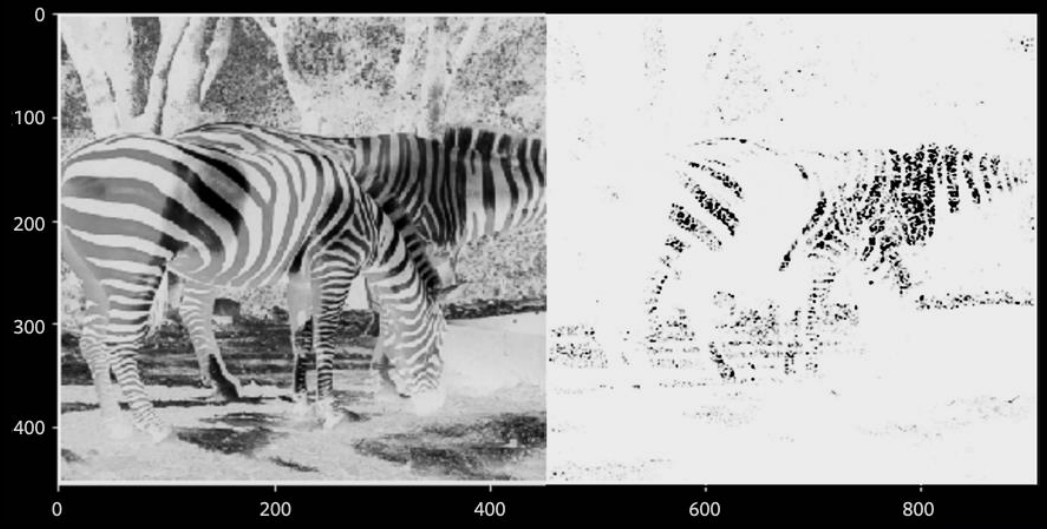
aws

© 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")
```

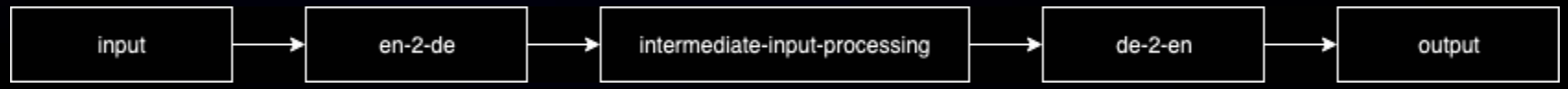
https://captum.ai/api/saliency.html

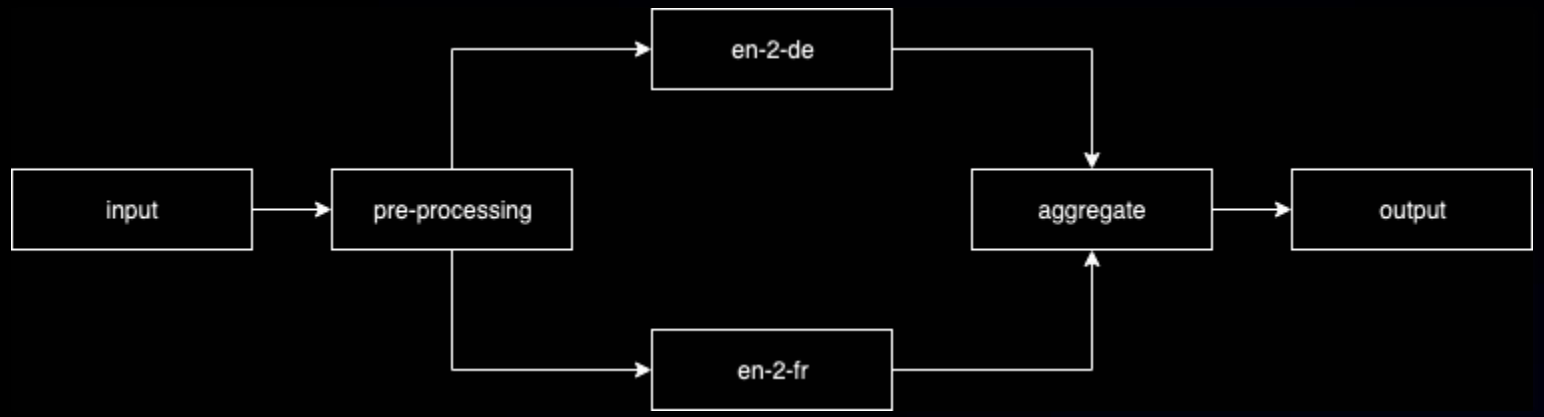
PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc., formerly known as Facebook, Inc.

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

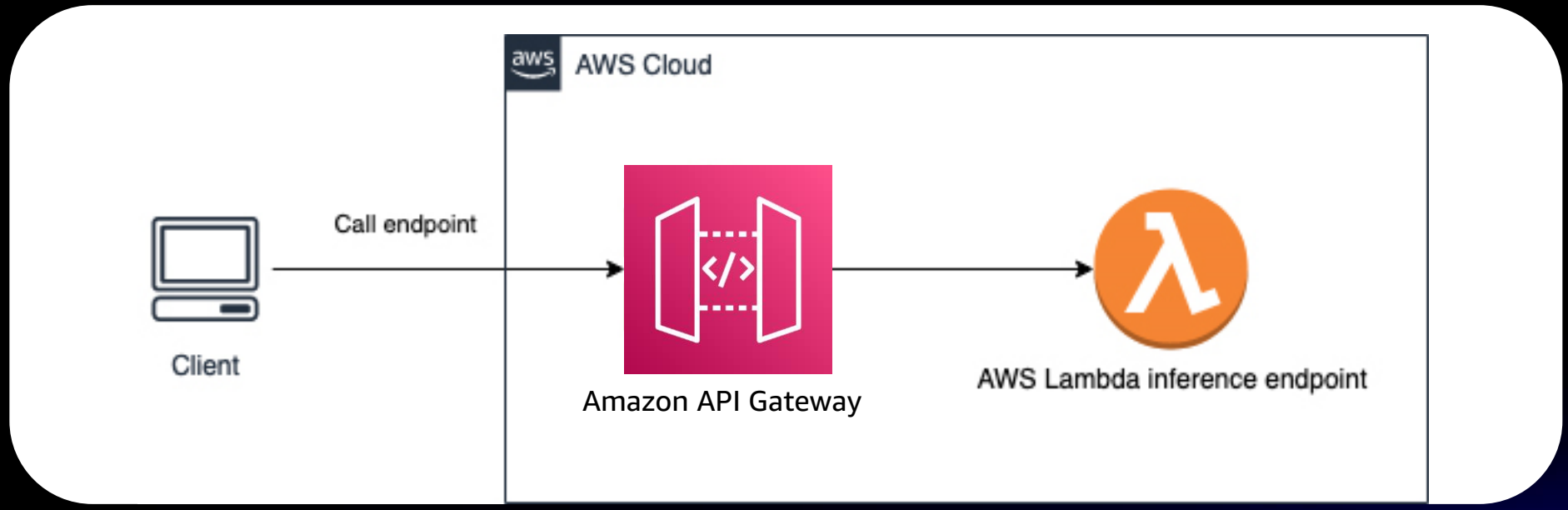
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

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

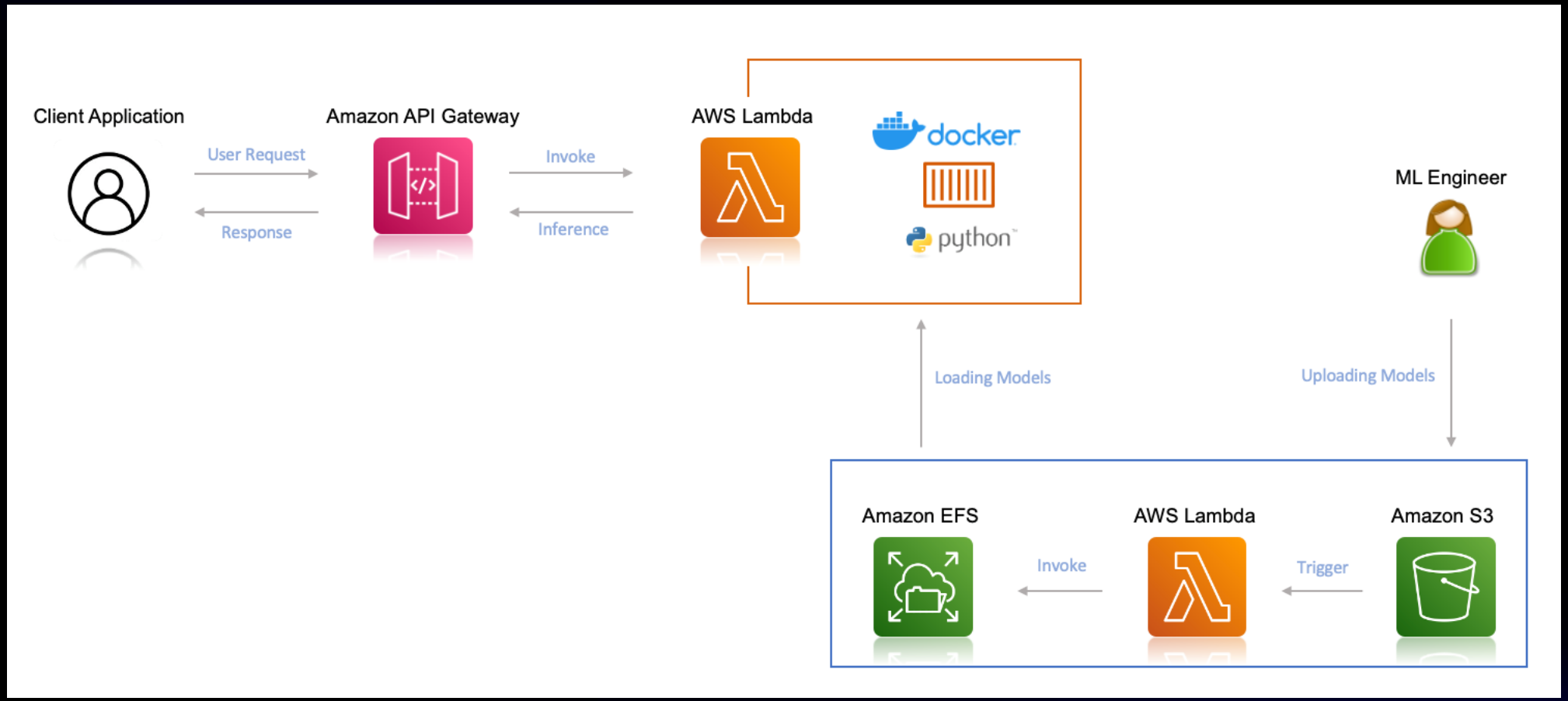
Run PyTorch models as AWS Lambda container images





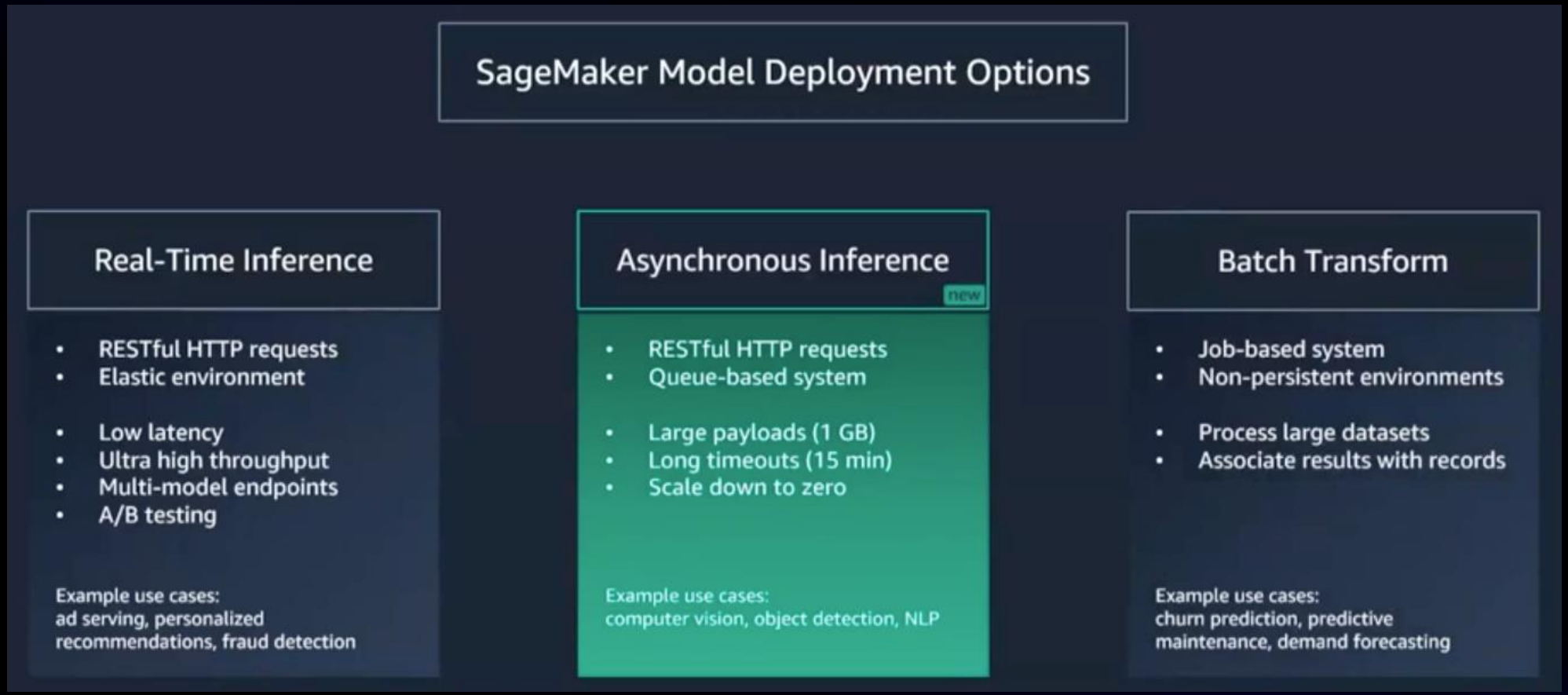
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

Deploy multiple models for inference on AWS Lambda and EFS



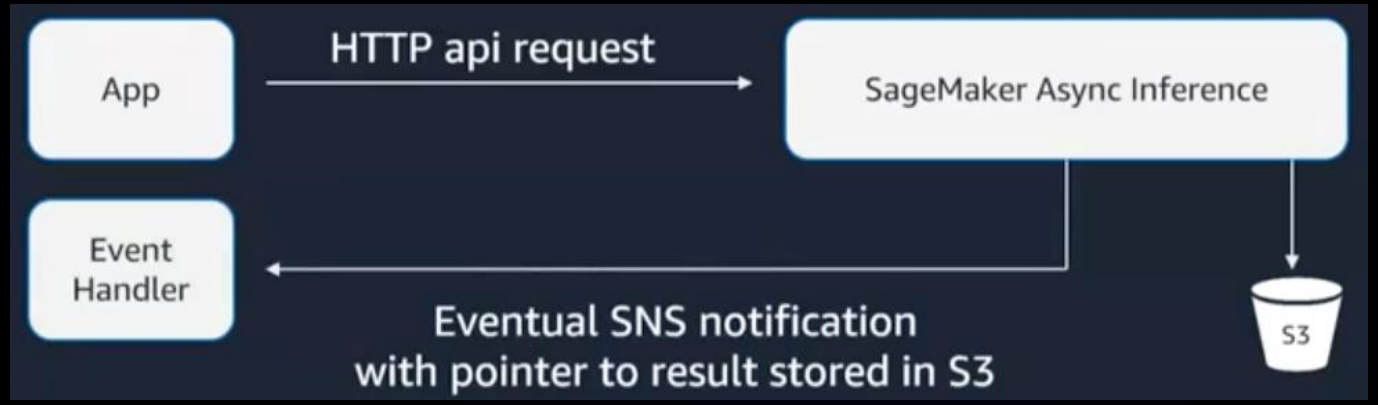
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

Amazon SageMaker Asynchronous Inference



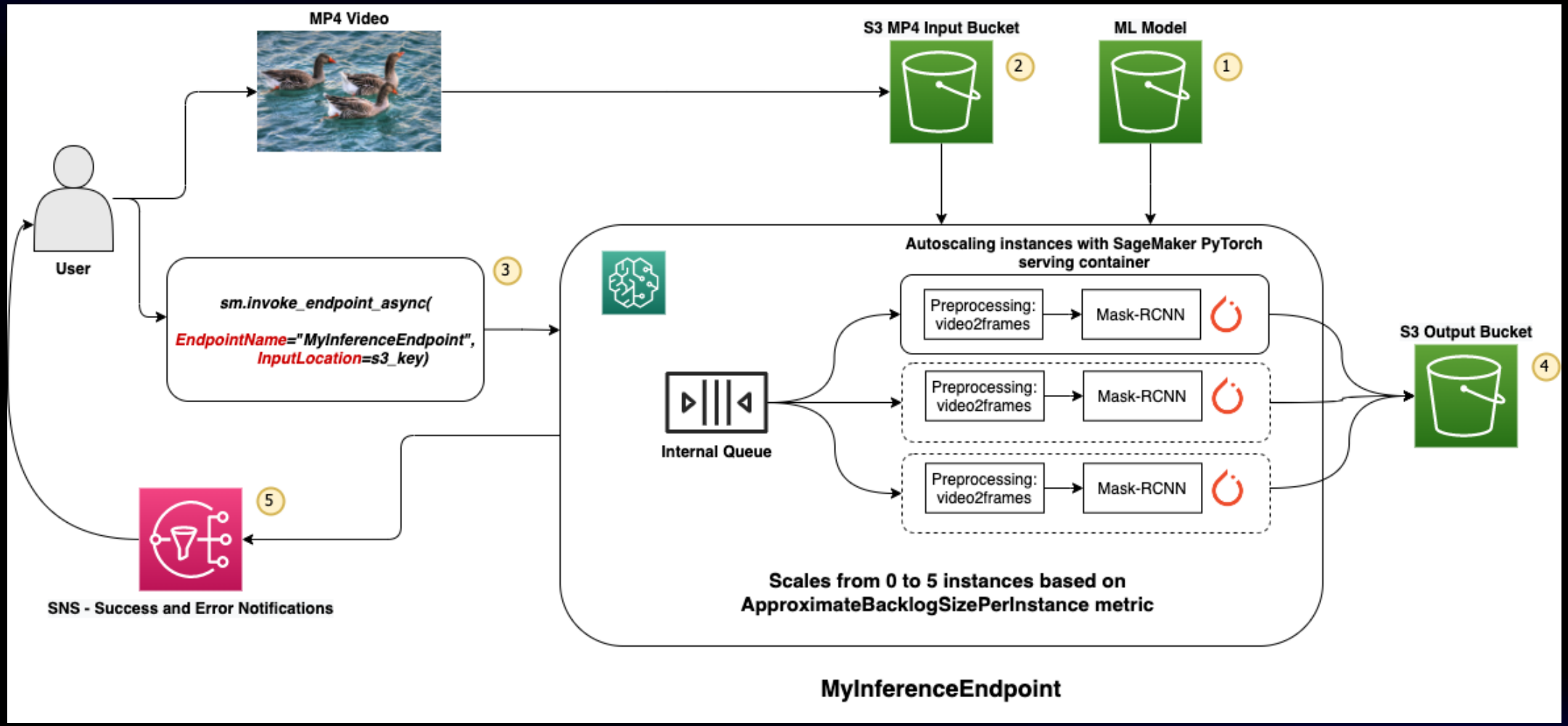
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

SageMaker Asynchronous Inference (cont.)



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

SageMaker Asynchronous Inference (cont.)



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

Amazon S3 plugin for PyTorch

You don’t need to change the existing code to make it work with Amazon S3

- `S3Dataset` and `S3IterarableDataset` provided by this plugin are implementations of PyTorch’s internal Dataset and IterableDataset interfaces
- Support for both map-style and iterable-style dataset interfaces
- File format agnostic: presents objects as a binary buffer (blob)
- Support for shuffling

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

HuggingFace SageMaker Integration

Hugging Face Estimator / Model / Predictor

```
from sagemaker.huggingface import HuggingFaceModel

# create Hugging Face Model Class
huggingface_model = HuggingFaceModel(
    transformers_version='4.6',
    pytorch_version='1.7',
    py_version='py36',
    model_data='s3://my-trained-model/artifacts/model.tar.gz',
    role=role,
)
# deploy model to SageMaker Inference
huggingface_model.deploy(initial_instance_count=1,instance_type="ml.m5.xlarge")
```

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

Resources

bit.ly/ptLambda

bit.ly/efsML

bit.ly/asyncInference

bit.ly/pytorchS3

bit.ly/sagemakerHF

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Disclaimer: PyTorch, the PyTorch logo and any related marks are trademarks of Meta Inc. formerly known Facebook, Inc.

Thank you!

Kris Skrinak

skrinak@amazon.com
@skrinak 

Hamid Shojanazeri

hamidnazeri@fb.com
twitter.com/nazeri2010
github.com/hamidshojanazeri

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