T1ALML001

Training ML models at scale with Amazon SageMaker

Vatsal Shah

Sr. Solutions Architect
AWS India

Session overview

- Benefits and challenges of large-scale machine learning
- Amazon SageMaker accelerates training ML models at scale
- Distributed training
- A sample walkthrough of training a LLM
- Useful resources

Rise of large-scale models

"a picture of a very clean living room"

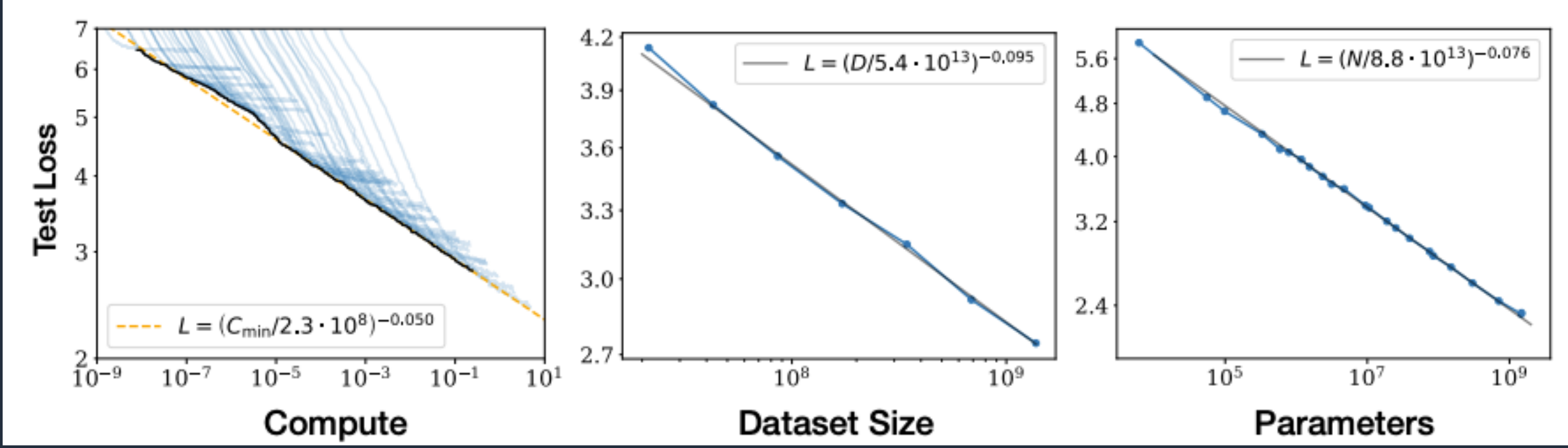
2017

StackGAN,
Zhang et al.

2022

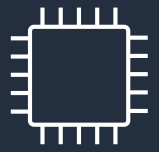
Stable Diffusion,
Rombach et al.

Large-scale models lead to better results



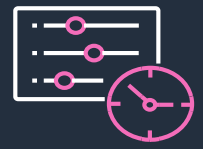
Scaling Laws for Neural Language Models
Kaplan et al., 2020

Challenges with training large-scale models



Hardware

Health checks



Orchestration

Data

Scaling up

Cost

Amazon SageMaker accelerates large-scale model training

Large-scale training on Amazon SageMaker

OPTIMIZED DISTRIBUTED TRAINING LIBRARIES & FRAMEWORKS

TensorFlow	PyTorch	**Hugging Face**	**Amazon SageMaker Distributed Training Libraries**	Bring your own library (e.g. DeepSpeed, Megatron)

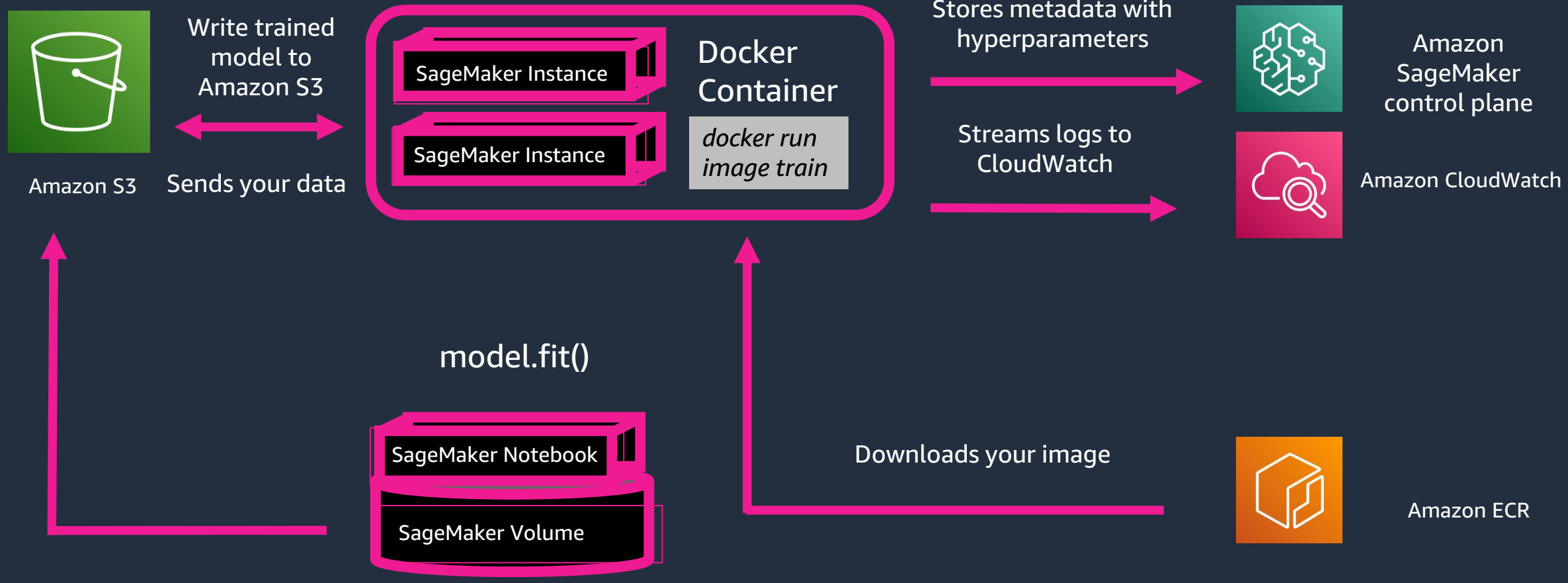
AMAZON SAGEMAKER TRAINING

Large Scale Cluster Orchestration	**NCCL Health Checks**	**Resilient training**	**SageMaker Compiler**	Warm pools	SSH to container
Data loading	Debugger	Profiling	Experiment tracking	Hyperparameter optimization	Pay for what you use

ML COMPUTE INSTANCES & ACCELERATORS

NVIDIA GPUS A100, V100, K80, T4, A10	**AWS Nitro**	400/800 Gbps EFA Networking	**CPU instances**	AWS Trainium

Amazon SageMaker ephemeral training clusters

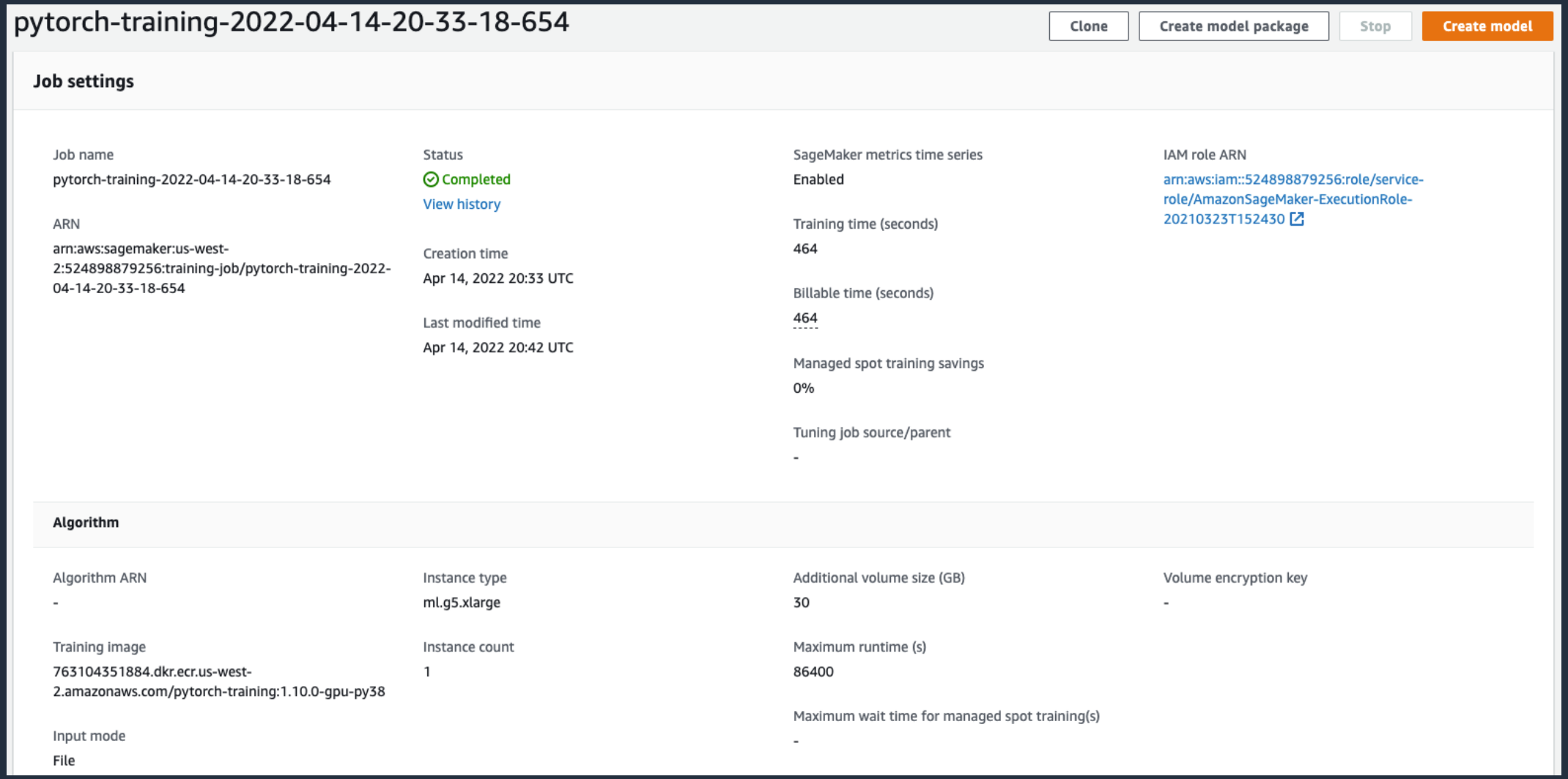


Fully managed by SageMaker

Train with your own deep learning model

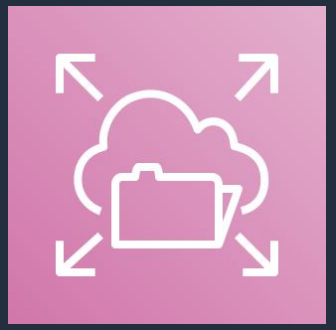
```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point = './cifar10.py',
                    role = role,
                    framework_version = '1.13',
                    py_version = 'py38',
                    instance_count = 1,
                    instance_type = 'ml.g5.xlarge',
                    hyperparameters = {'epochs': 50, 'batch_size': 32},
                    metric_definitions = [{'Name': 'train:loss', 'Regex': 'loss: (.*)'}])

estimator.fit("s3://bucket/path/to/training/data")
```

Replicate experimental results by default



Easy and performant data loading

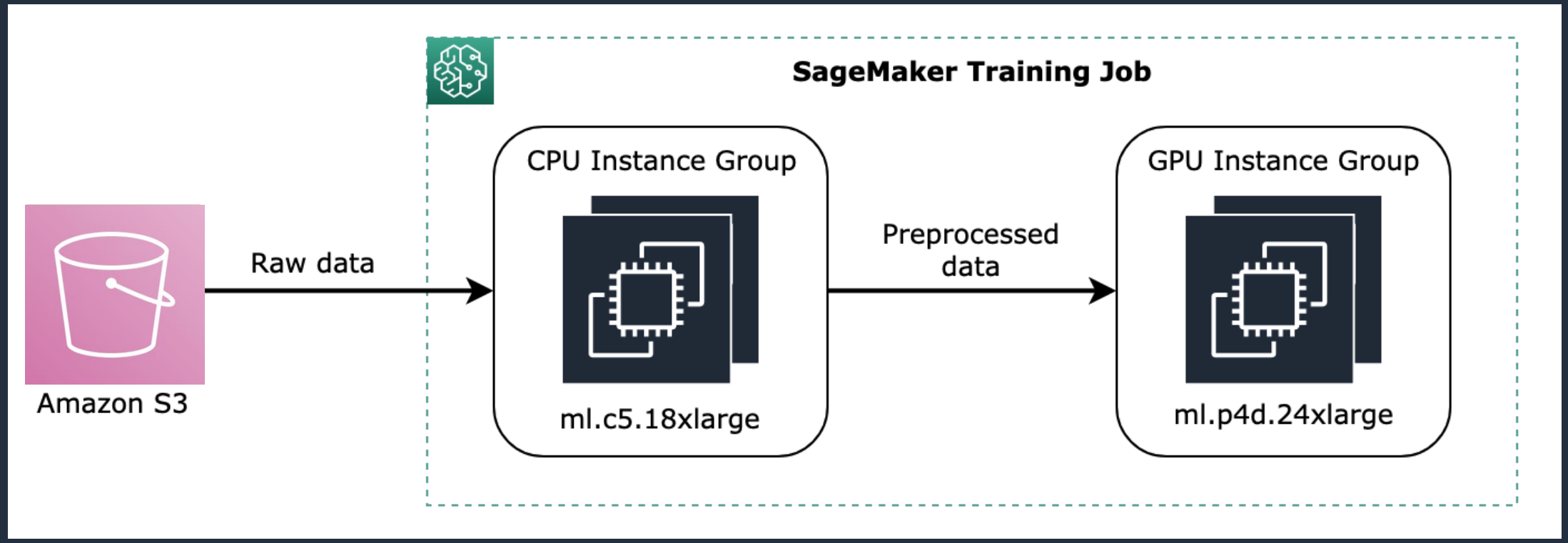
Amazon Simple Storage Service (S3)



Amazon Elastic File System (EFS)

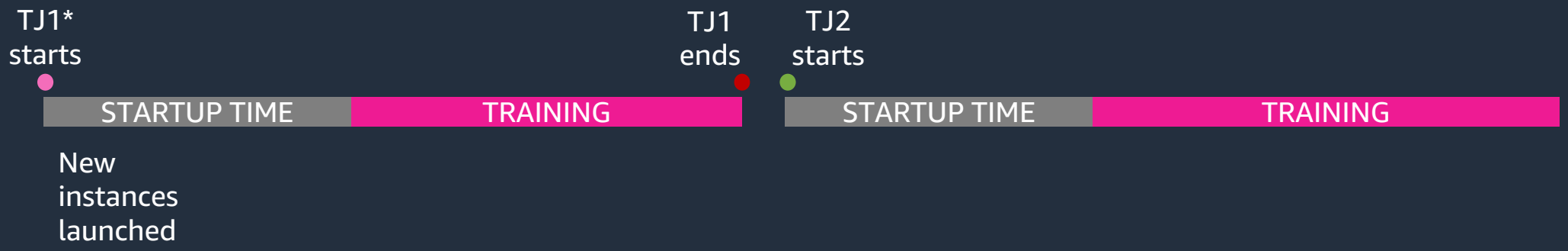
Amazon FSx for Lustre

Heterogeneous clusters: Better GPU utilization



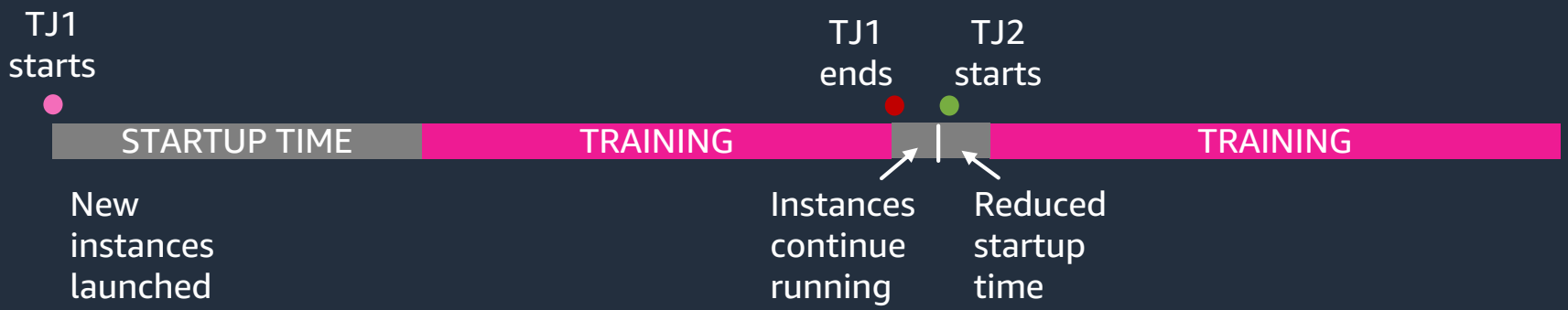
Warm Pools: Faster startup time

Before: Multi-minute wait between script updates



After: Multi-*second* wait between script updates

`Keep_alive_period_in_seconds=600`



*TJ=Training Job

Distributed Training with Amazon SageMaker

Why do we need distributed training?

INCREASING COMPLEXITY

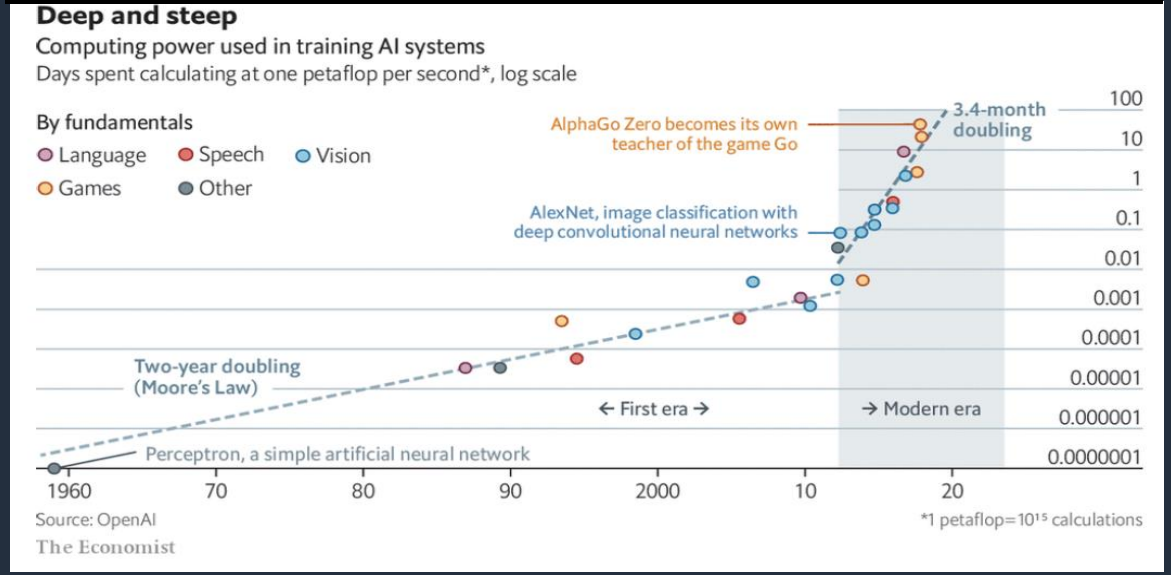
- Businesses need higher precision in their model predictions
- Results in larger and more complex models
- Requires frequent retraining of models

INCREASING COSTS

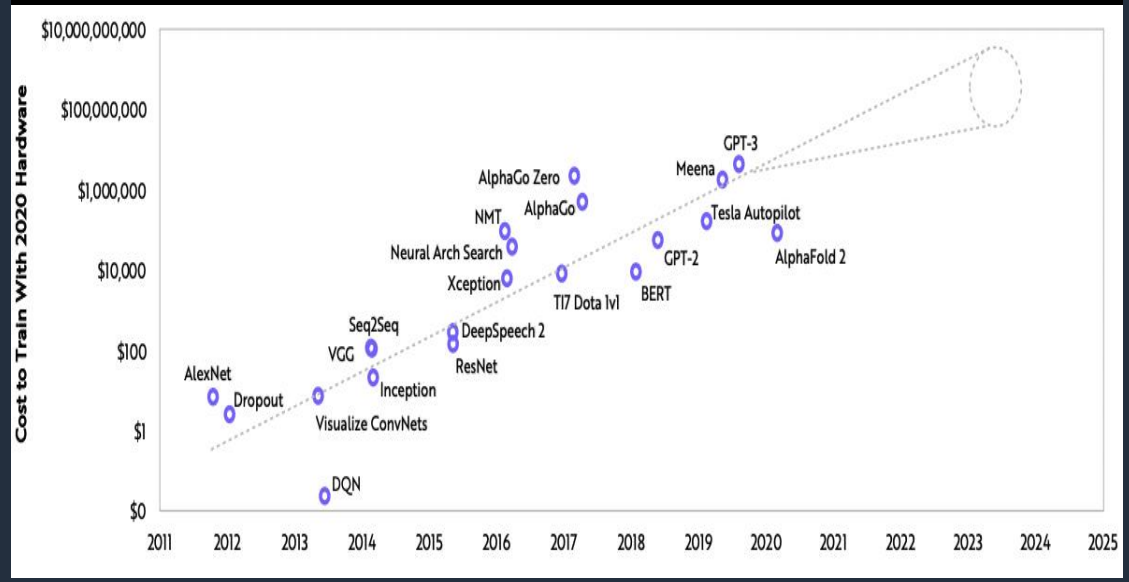
- Increasing compute power required for frequent training of larger models drives up cost to train
- Becomes a barrier for innovation and growth

MODELS GROW FASTER THAN HARDWARE, LEADING TO BOTTLENECKS

Compute power ~ 2x every 3.4 months



Model size increase ~ 10x/ year, Cost of Training increase ~ 100x by 2025



*According to Open AI as of 2018
*ARK Invest Big Ideas 2021

What are large language models (LLMs)?

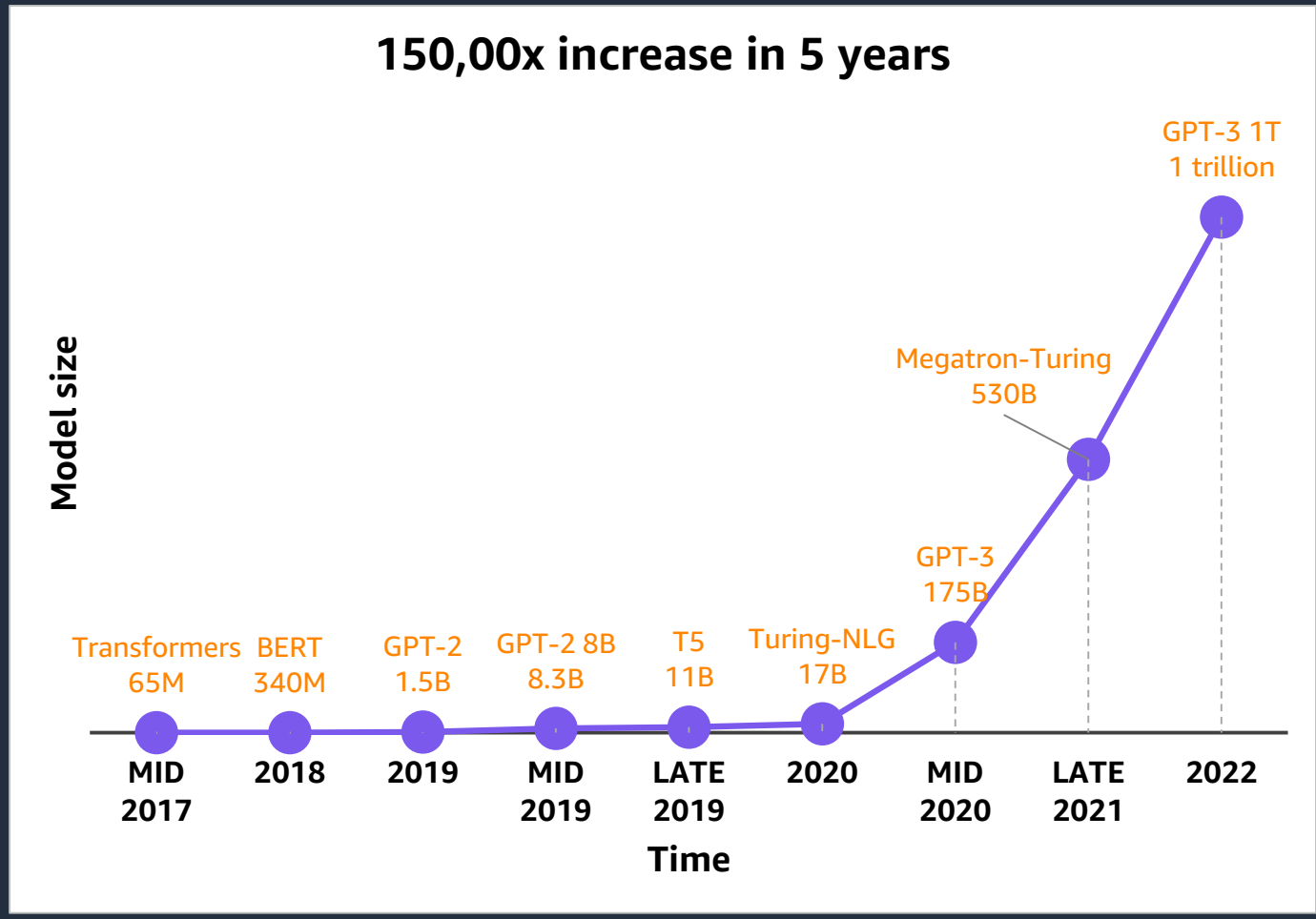
Size of large NLP models is increasing

Models are becoming more complex, with ML practitioners moving from classical ML to deep learning

State-of-the-art deep learning models are getting larger and larger as we find that larger models generalize better

They can be applied to a range of tasks to create more powerful digital assistants and generate better search results and product recommendations

They can be used for multi-modal use cases like answering visual questions from people who are vision-impaired, answering questions visually, emotion recognition, and many more



Pretraining vs. fine-tuning LLMs

	Pretraining	Fine tuning
Training duration (and cost)	Weeks to months	Minutes to hours
Customization	**FULL** • NN architecture and size • Vocabulary size • Context length • Training data	**Some** • Specific task tuning • Added domain-specific training data
Expertize needed	High	Low

Challenges in pretraining LLMs

Training LLMs in Amazon SageMaker

DISTRIBUTED TRAINING LIBRARIES

TensorFlow	PyTorch	**SageMaker Distributed Training**	OSS libraries

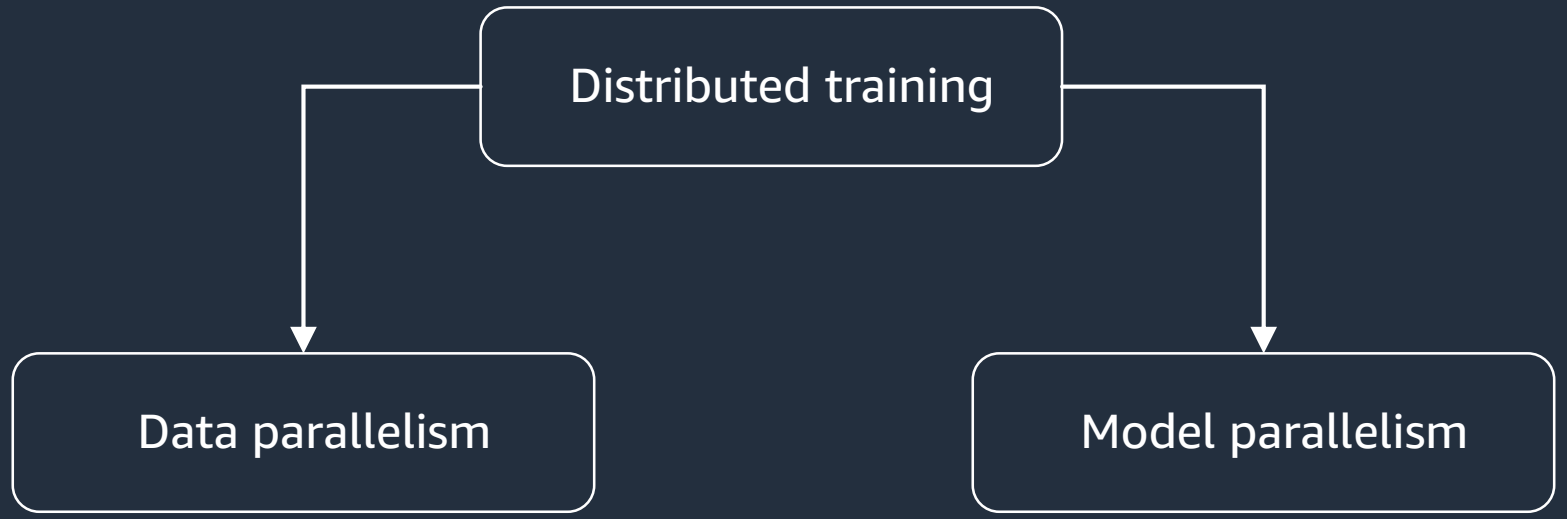
AMAZON SAGEMAKER PLATFORM

Large Scale Cluster Orchestration	Fault tolerant	Hugging Face integration	SageMaker Training Compiler

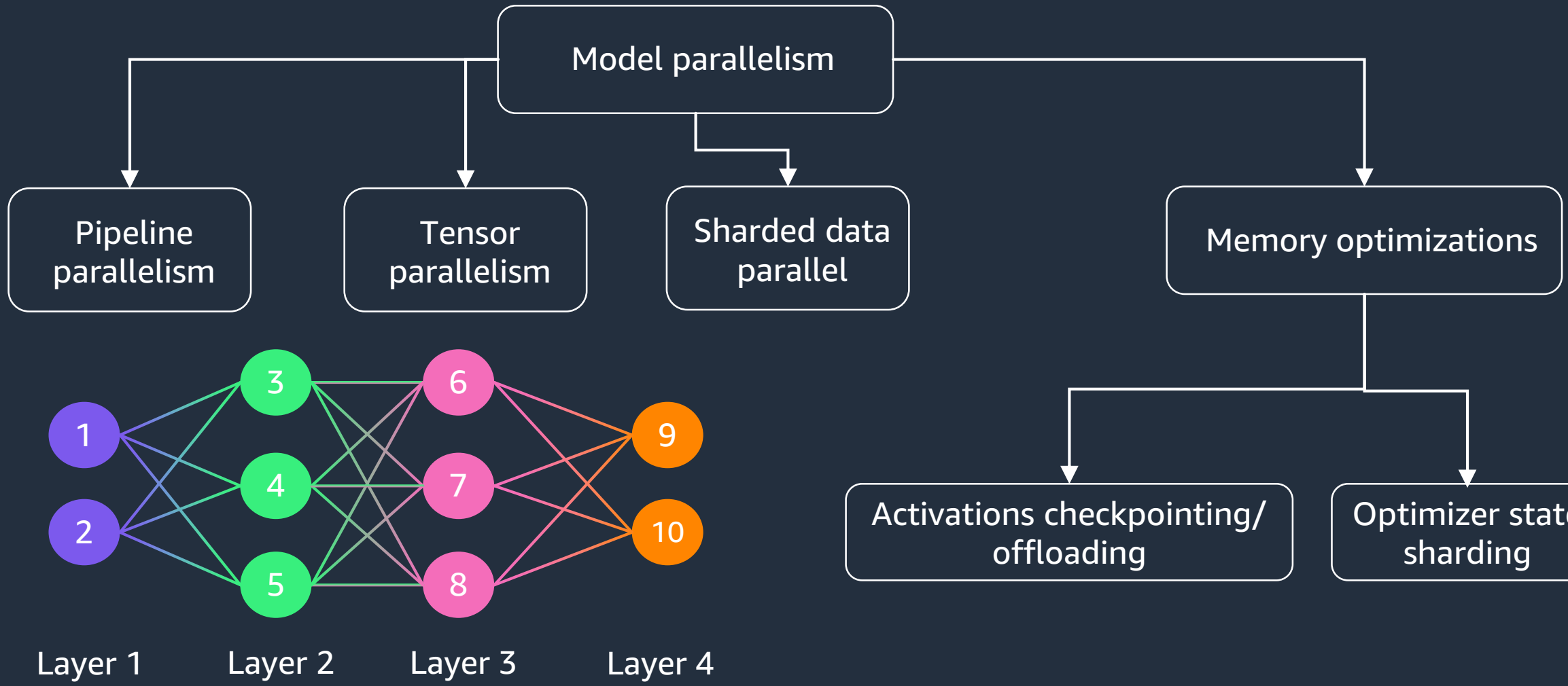
ML COMPUTE INSTANCES & ACCELERATORS

400/800 Gbps EFA networking	**NVIDIA A100 GPUs 40 GB/80 GB**	AWS Trainium	**Habana Gaudi**

Distributed training methods



Amazon SageMaker model parallelism options



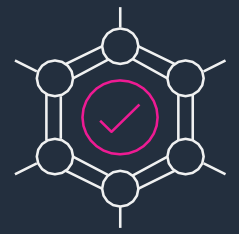
Data parallel, think “massive data”

Historical approaches to distributed gradient descent

Parameter server

TensorFlow

ParameterServerStrategy



MPI AllReduce

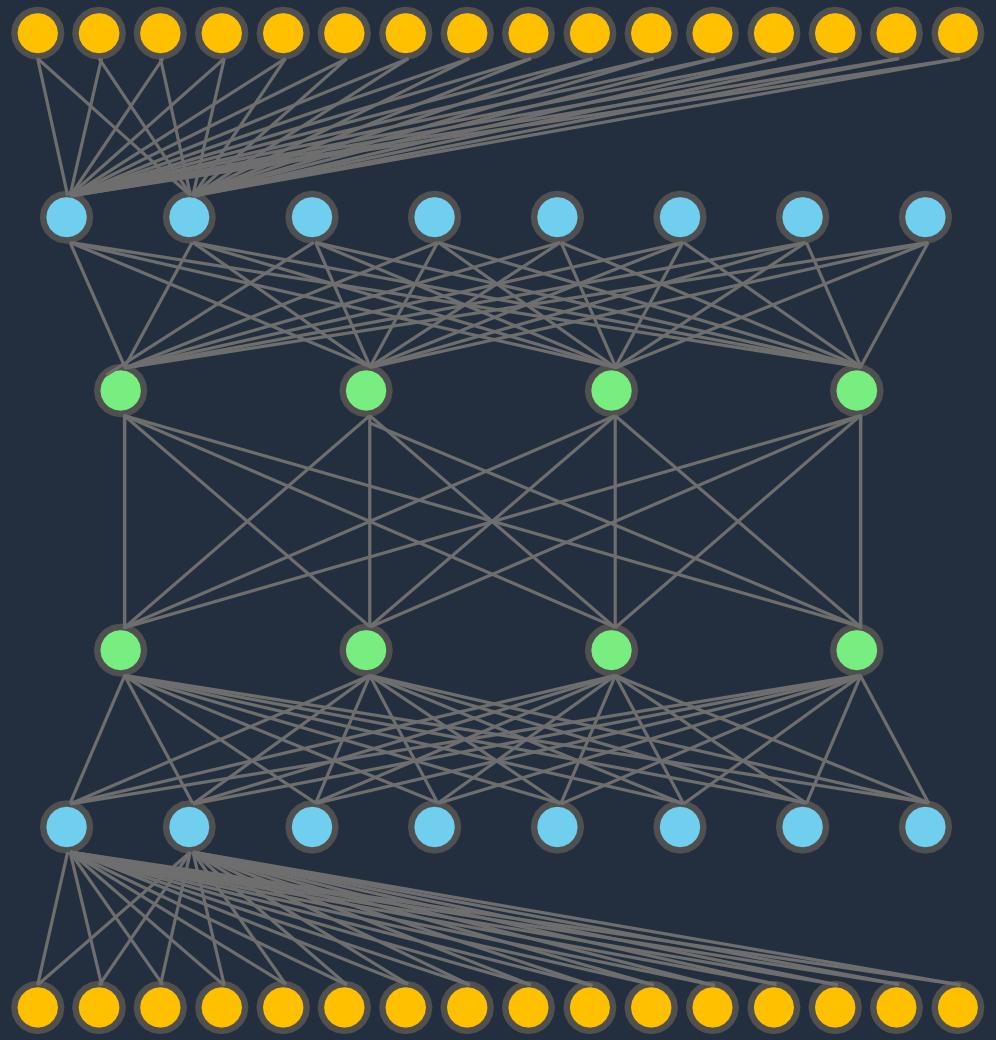
Horovod

PyTorch

DistributedDataParallel

Amazon SageMaker Distributed Data Parallel

- Optimized backend for distributed training of deep learning models in TensorFlow, PyTorch
- Accelerates training for network-bound workloads
- Built and optimized for AWS network topology and hardware
- 20%–40% faster and cheaper than NCCL and MPI-based solutions. **Best performance on AWS for large clusters.**



NEW!

AWS Machine Learning Blog

Run PyTorch Lightning and native PyTorch DDP on Amazon SageMaker Training, featuring Amazon Search

by Emily Webber, Abhinandan Patni, Mayank Jha, Karan Dhiman, Eiman Elnahrawy, and Vishwa Karia | on 18 AUG 2022 | in Amazon SageMaker | Permalink | Comments | Share

Number of Instances	Training Time (minutes)	Improvement
1	99	Baseline
2	55	1.8x
4	27	3.7x
8	13.5	7.3x

- 1 line of code to plug in PyTorch Lightning with SageMaker Distributed Data Parallel backend
- Easily run native PyTorch DDP on SageMaker Training
- Amazon Search saw 7.3x speedup moving to distributed training on SageMaker

Use the SageMaker Distributed Data Parallel Library as the Backend of `torch.distributed`

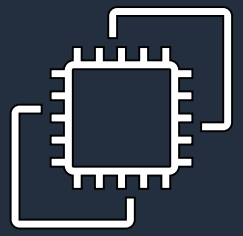
To use the SageMaker distributed data parallel library, the only thing you need to do is to import the SageMaker distributed data p
library's PyTorch client (`smdistributed.dataparallel.torch.torch_smddp`). The client registers `smddp` as a backend for
When you initialize the PyTorch distributed process group using the `torch.distributed.init_process_group` API, make su
`'smddp'` to the `backend` argument.

```
import smdistributed.dataparallel.torch.torch_smddp
import torch.distributed as dist

dist.init_process_group(backend='smddp')
```

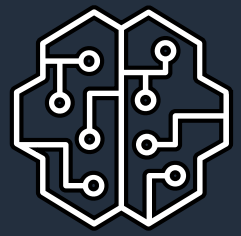
```
ddp = DDPStrategy(
    cluster_environment=env,
    process_group_backend="smddp",
    accelerator="gpu")
```

Model parallel, think “massive models”

Model parallelism on Amazon SageMaker (SMP)



Automated model partitioning



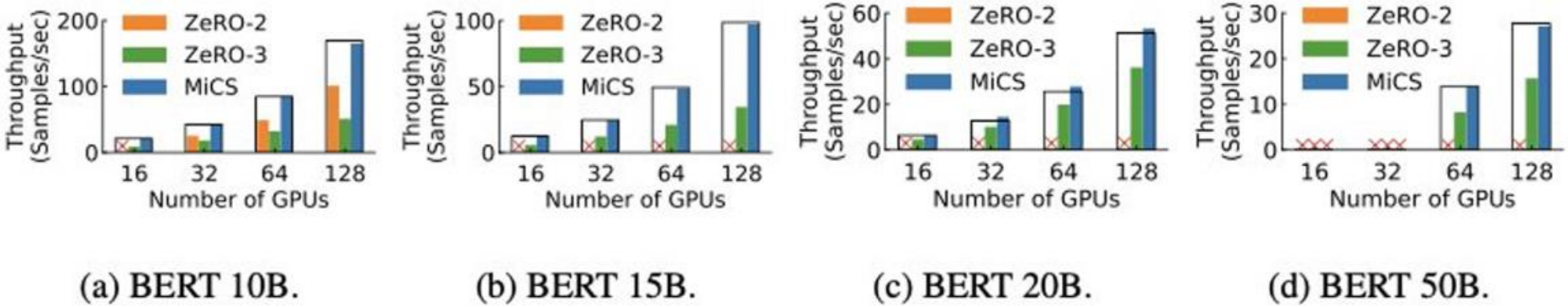
Interleaved pipelined training

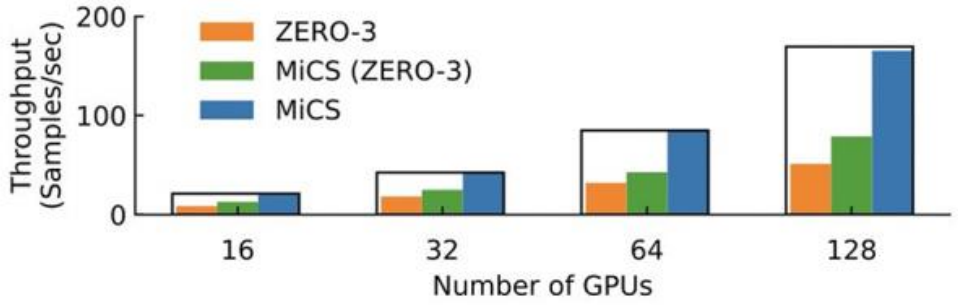
Managed Amazon SageMaker training

Clean framework integration

Scale near linearly with *Sharded Data Parallelism*

MiCS achieves 169 TFLOPS per GPU with 175B parameter model on AWS p4de.24xlarge instances





- MiCS hits 99.4% of linear-scaling efficiency from 128 to 512 GPUs
- DeepSpeed hits only 72% , saturates at 62 TFLOPS per GPU

Available within Amazon SageMaker Model Parallel
2.8x faster than DeepSpeed

NEW!

Steps to train 10 TB on 200 SageMaker GPUs

Data download

18 parallel CPU jobs

SageMaker Training

Local dev and test

2 CPUs <-> 96 CPUs

SageMaker Studio

Lightweight job test

1 Small GPU

SageMaker Training

FSx for Lustre

Set up VPC

Establish data repository

SageMaker Training

Build data index

2 CPUs <-> 96 CPUs

SageMaker Studio

Run final job

200 GPUs

SageMaker Training

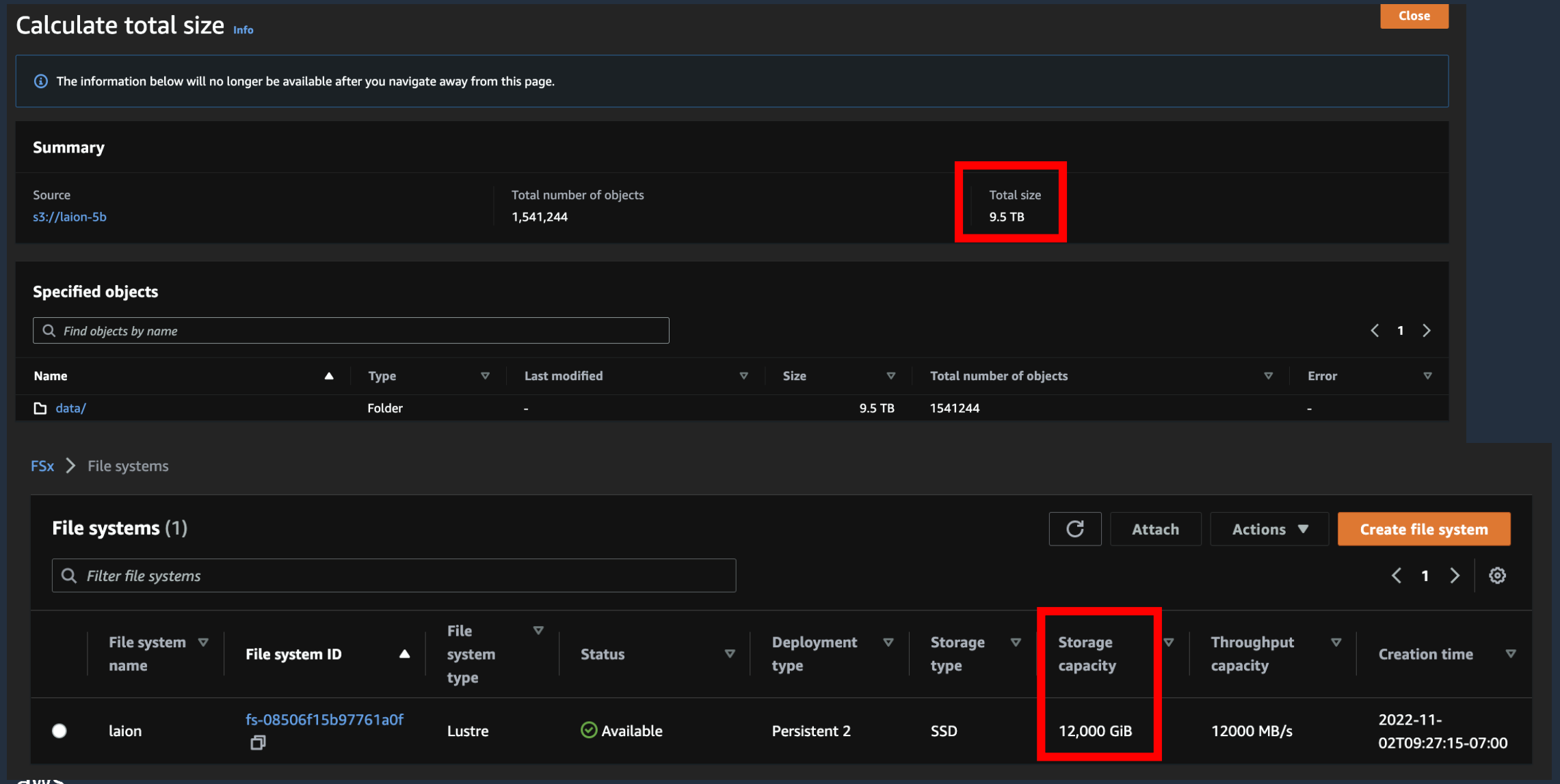
10x faster than 1 node baseline

Training loop completed in 15 minutes

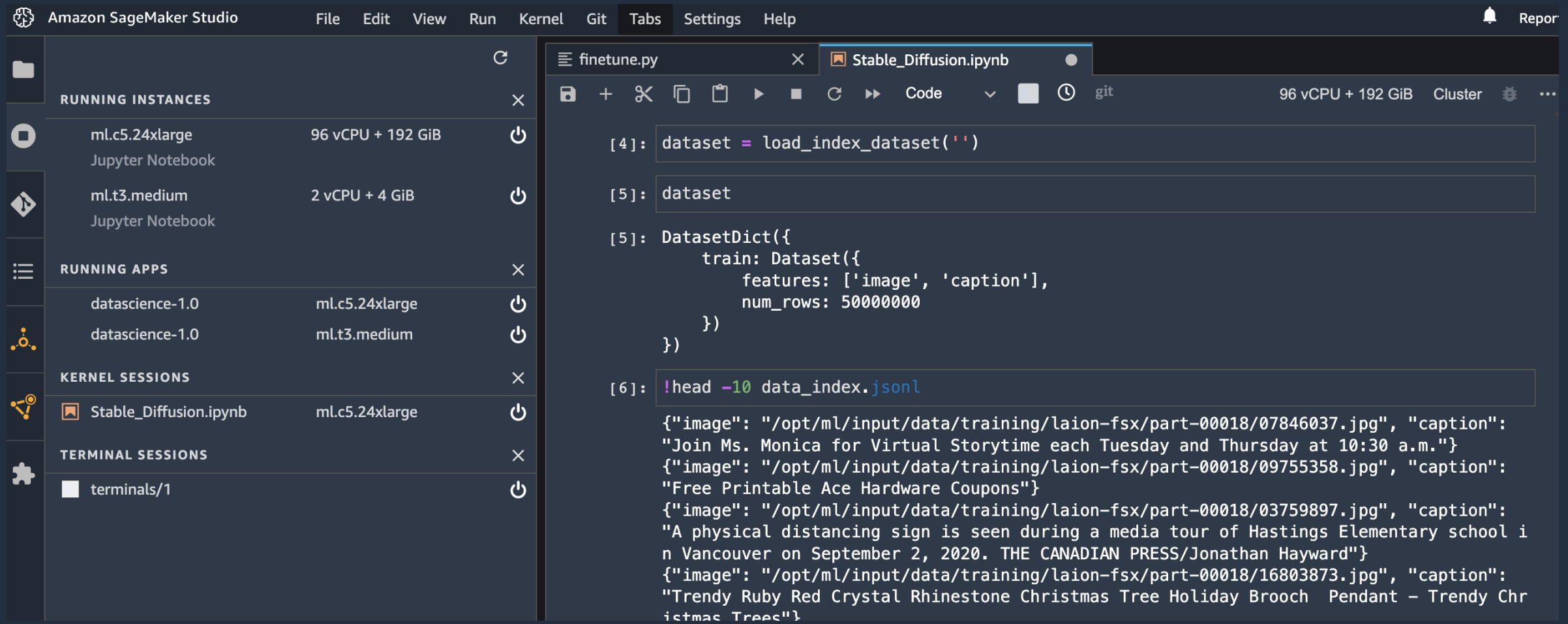
1/ Download data with CPU job parallelism

```
for p_file in parquet_list[:18]:

    part_number = p_file.split('-')[1]

    output_dir = "s3://{}/data/part-{}/".format(bucket, part_number)

    if is_open(output_dir):

        est = get_estimator(part_number, p_file, output_dir)

        est.fit({"parquet":"s3://{}/meta/{}".format(bucket, p_file)}, wait=False)
```

2/ Create Lustre in minutes from S3 path



3/ Run 96 CPUs on Studio to develop and test



4/ Scale to 200 GPUs with SageMaker Training

```
est = HuggingFace(entry_point='finetune.py',
                  source_dir='src',
                  image_uri=image_uri,
                  sagemaker_session=sess,
                  role=role,
                  output_path="s3://laion-5b/output/model/",
                  instance_type='ml.p4dn.24xlarge',
                  keep_alive_period_in_seconds = 60*60,
                  py_version='py38',
                  base_job_name='fsx-stable-diffusion',
                  instance_count=24,
                  enable_network_isolation=True,
                  encrypt_inter_container_traffic = True,
                  # all opt/ml paths point to SageMaker training
                  hyperparameters = hyperparameters,
                  distribution={"smdistributed": { "dataparallel": { "enabled": True } }},
                  max_retry_attempts = 30,
                  max_run = 4 * 60 * 60,
                  debugger_hook_config=False,
                  disable_profiler = True,
                  **kwargs)

est.fit(inputs=data_channels, wait=False)
```

5/ Host on SageMaker Real-Time Endpoints

```
import sagemaker
from sagemaker.deserializers import JSONDeserializer
from sagemaker.serializers import JSONSerializer


sess = sagemaker.Session()

endpoint_name = 'sd-inference-gpu-2022-10-16-14-46-56-112'

pred = sagemaker.predictor.Predictor(endpoint_name,
                                     sagemaker_session=sess,
                                     serializer=JSONSerializer(),
                                     deserializer=JSONDeserializer())
```

```
prompt = "a Christmas tree in Las Vegas"

output = pred.predict({'inputs':prompt})
process_result(output['images'][0])
```

Start your LLM journey today

Get started now
It's free!

Jurassic-1: Technical Details and Evaluation
White paper covering Large and Jumbo

Read the docs
API documentation, guides and more

Introducing J1-Grande
Blog post with benchmark results and samples

Build a sentiment analysis dashboard
Extract insights from reviews in 15 minutes

Build a CV Profile Generator
Teach Jurassic-1 Grande to generate a profile for a given role and skills from just a few examples

Getting started

Product page
aws.amazon.com/sagemaker/train/

Technical Documentation
docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

SageMaker examples on GitHub
github.com/aws/amazon-sagemaker-examples

AWS ML blogs
https://aws.amazon.com/blogs/machine-learning/

Thank you!

Vatsal Shah
vatsalbshah