DATA+AI SUMMIT BY databricks

Empowering Generative AI with Databricks and AMD

Keith Anderson & Avinash Sooriyarachchi
May 2024

Speaker Info

Keith Anderson

Senior Member of Technical Staff

AMD

Avinash Sooriyarachchi

Senior Solutions Architect

databricks

AMD + DATABRICKS FOR GENERATIVE AI

Session summary

AMD GPU Hardware & Software

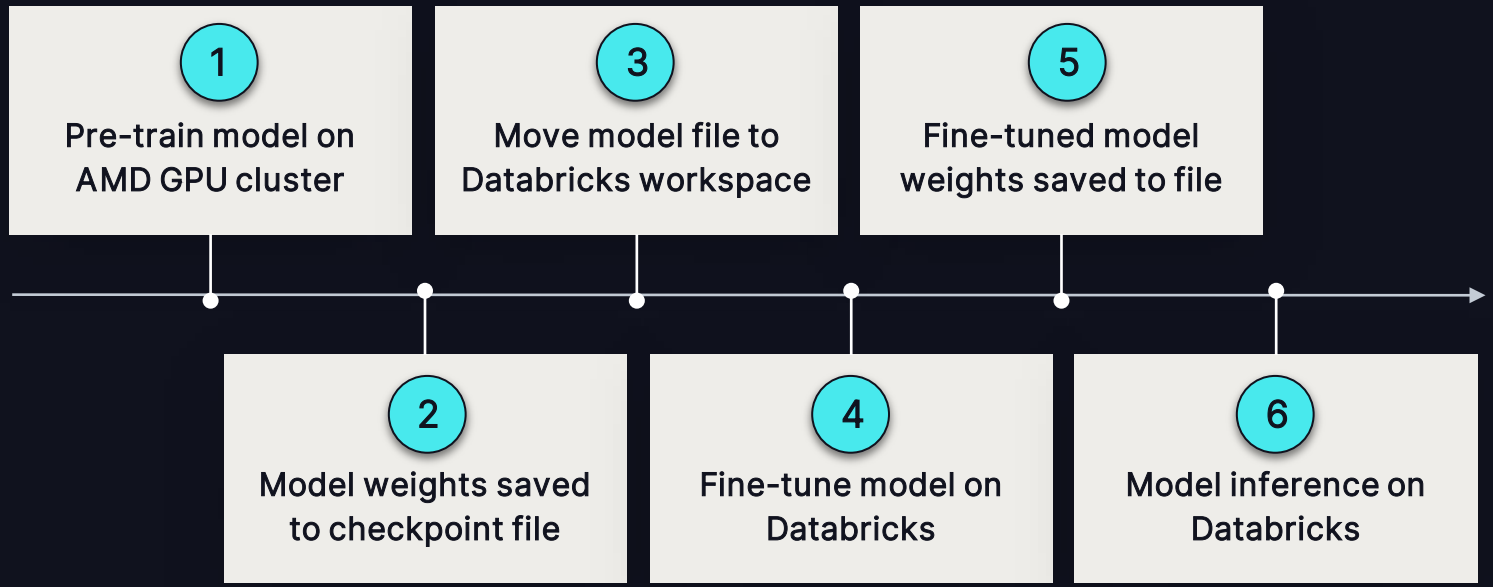
- AMD GPU HW & SW pre-training large AI models w/existing frameworks
- Flexible on-prem or GPU cloud pre-training for public/private datasets
- Scaling past single host workloads using RDMA & AMD GPUs

Databricks Model Management & Production Deployment

- AI model sharing with Databricks
- Fine-tuning and model inference on Databricks
- Better governance, privacy, and cost-effectiveness w/AMD & Databricks

TRAINING A GENERATIVE AI MODEL

High level end-to-end workflow



MODEL PRE-TRAINING ON AMD

AMD HARDWARE & SOFTWARE

Accelerate HPC & AI Workloads

INSTINCT ACCELERATORS

- MI300X series (1.3 PFLOPS bfloat16 peak, 1.5TB HBM3)
- xGMI for inter-GPU communications (896GB/s bidirectional peer-to-peer)
- Azure ND MI300X V5 instances (8 GPUs, 400 Gb/s IB link per GPU, 3.2Tb/s per VM)

ROCM

- Open-source stack for GPU compute
- HPC & AI workloads
- Datacenter & consumer-grade GPUs
- HIP = API for GPU accelerated apps
- RCCL = collective communications

ECOSYSTEM

- Support for popular AI & ML frameworks
- TensorFlow, JAX, Pytorch
- Hugging Face, DeepSpeed, ONNX
- AMD Infinity Hub for software containers

SCALING PRE-TRAINING

Moving past single GPU workloads

Hardware

Fast inter-node back-end RDMA network

- IB, RoCE (RDMA over converged ethernet)
- Benchmark with Linux RDMA `perftest`

Fast intra-node GPU connectivity

- PCIe, xGMI
- Benchmark with AMD `TransferBench`

Fast External storage

- Training data, model checkpoints, code
- Fast front-end network, fast storage (NFS)

Software

Linux RDMA networking stack

ROCm - RCCL

- Collective communications across multi-GPU
- Benchmark with AMD `rccl-test`

Distributed frameworks (Pytorch, DeepSpeed)

- Data (DDP)
- Model (FSDP)
- Pipeline (DeepSpeed + Megatron)

PRE-TRAINING GPT2

nanoGPT (Pytorch) with OpenWebText dataset

Environment

- AMD EPYC 7763 64-Core Processor, dual socket, two (2) NUMA nodes, 2TB RAM
- Eight (8) AMD MI250 GPUs per host, 68GB VRAM per GPU (544GB VRAM per host)
- 200Gbps NICs back-end RoCEv2
- ROCm 5.7 (includes RCCL), Pytorch 2.0.1

Model Architecture

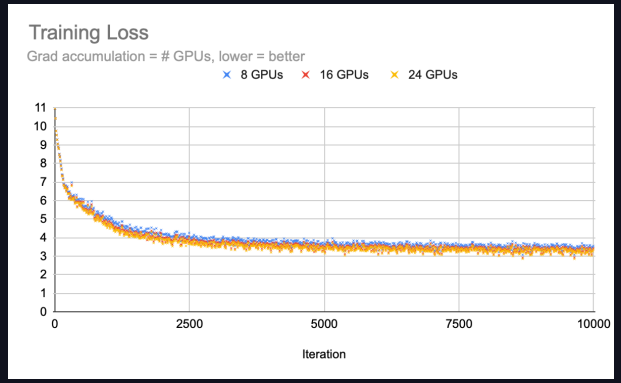
- Parameters
  - **Small** - 124M parameters (n_layer=12, n_head=12, n_embd=768)
  - **Medium** - 350M parameters (n_layer=24, n_head=16, n_embd=1024)
  - **Large** - 774M parameters, n_layer=36, n_head=20, n_embd=1280)
- Pytorch DDP (Distributed Data Parallel)
  - Full model replica on each GPU
  - Gradients sync over back-end network via RDMA

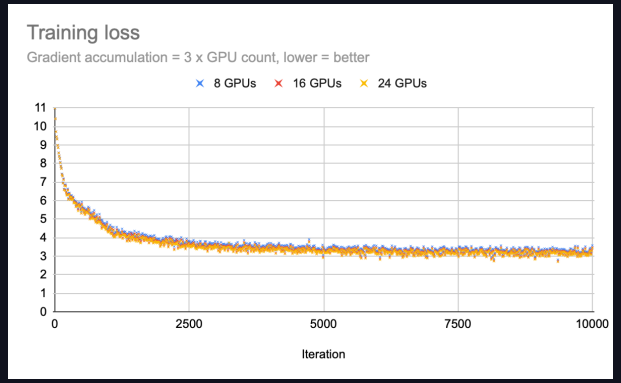
Training

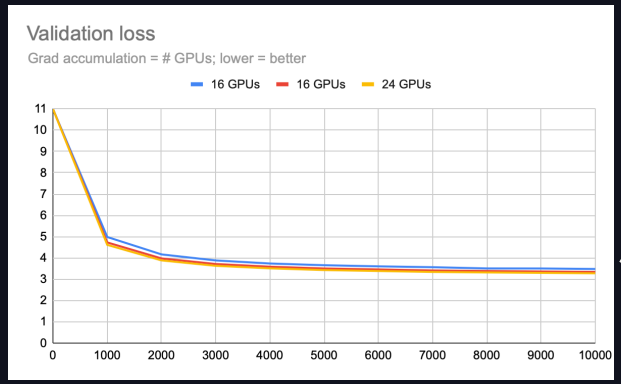
- Default batch size & block size, gradient accumulation steps = # GPUs and 3 x #GPUs
- OpenWebText = 9B tokens in training set (~17GB); 4M tokens in val set (~8.5MB)
- Pytorch ‘torchrun’ on each node with RCCL backend, each node rank unique

PRE-TRAINING SCALE

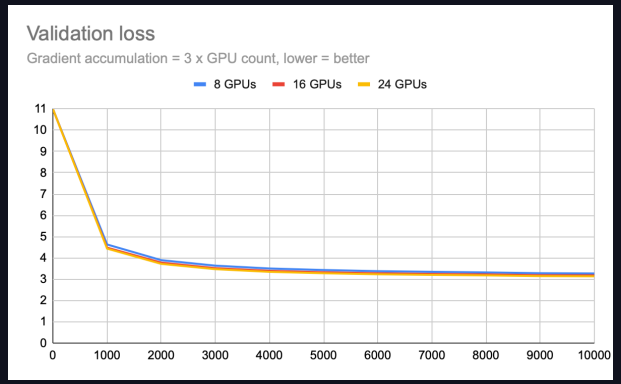
GPT2 - 124M parameters







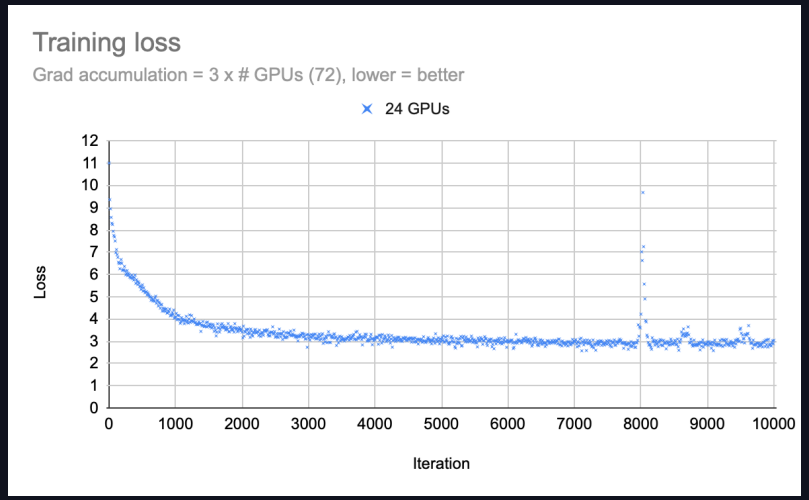
Best loss
24 GPUs
3.2776

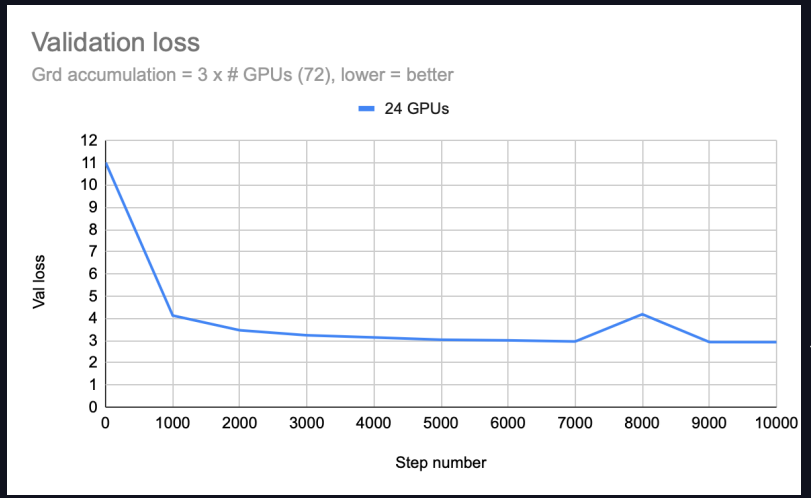


Best loss
24 GPUs
3.1366

PRE-TRAINING GPT2

GPT2 - 774M parameters





Best loss
24 GPUs
2.9266

TRANSITIONING MODEL TO DATABRICKS

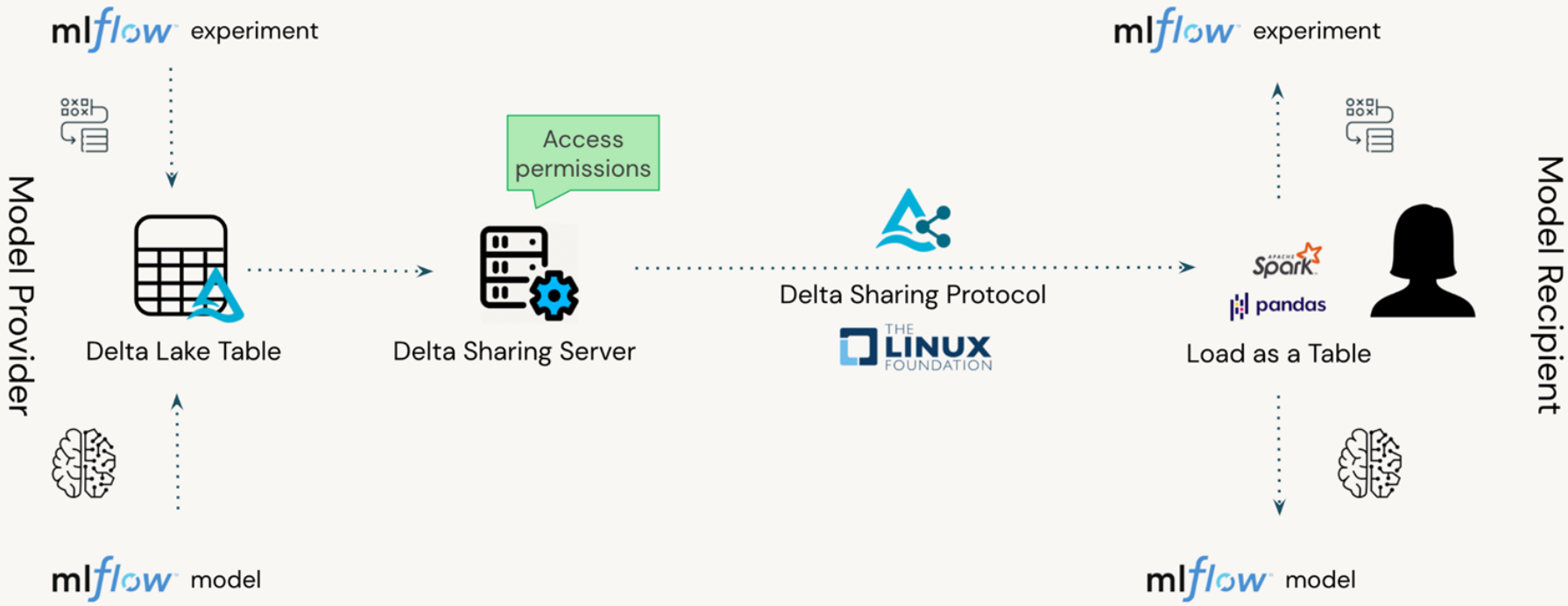
TRAINING & MODEL FILES

17GB training set -> 8.7GB model checkpoint

Training and validation sets	Model checkpoint file

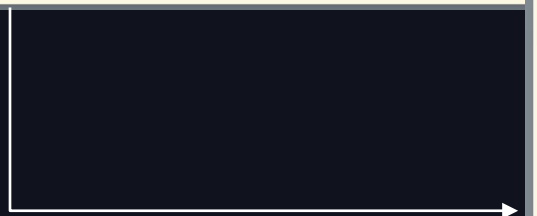
`#ls -lah openwebtext/`
`total 17G`
`drwxr-xr-x 2 root root  93 Apr  6 19:14 .`
`drwxr-xr-x 5 root root  84 Apr  5 13:44 ..`
`-rw-r--r-- 1 root root 3.1K Apr  5 13:44 prepare.py`
`-rw-r--r-- 1 root root  489 Apr  5 13:44 readme.md`
`-rw-r--r-- 1 root root  17G Apr  6 16:41 train.bin`
`-rw-r--r-- 1 root root 8.5M Apr  6 19:14 val.bin`
	`#ls -lah out/`
	`total 8.7G`
	`drwxr-xr-x  2 root root   21 Jun  5 21:39 .`
	`drwxr-xr-x 10 root root 4.0K Jun  5 18:43 ..`
	`-rw-r--r--  1 root root 8.7G Jun  5 21:37 ckpt.pt`

Model exchange via Delta Sharing



MODEL FINE TUNING

```
from peft import LoraConfig
...
...

#If only targeting attention blocks of the model
target_modules = ["q_proj", "v_proj"]

#If targeting all linear layers
target_modules = ['q_proj','k_proj','v_proj','o_proj','gate_proj','down_proj','up_proj','lm_head']

lora_config = LoraConfig(
r=16,
target_modules = target_modules,
lora_alpha=8,
lora_dropout=0.05,
bias="none",
task_type="CAUSAL_LM",}
```

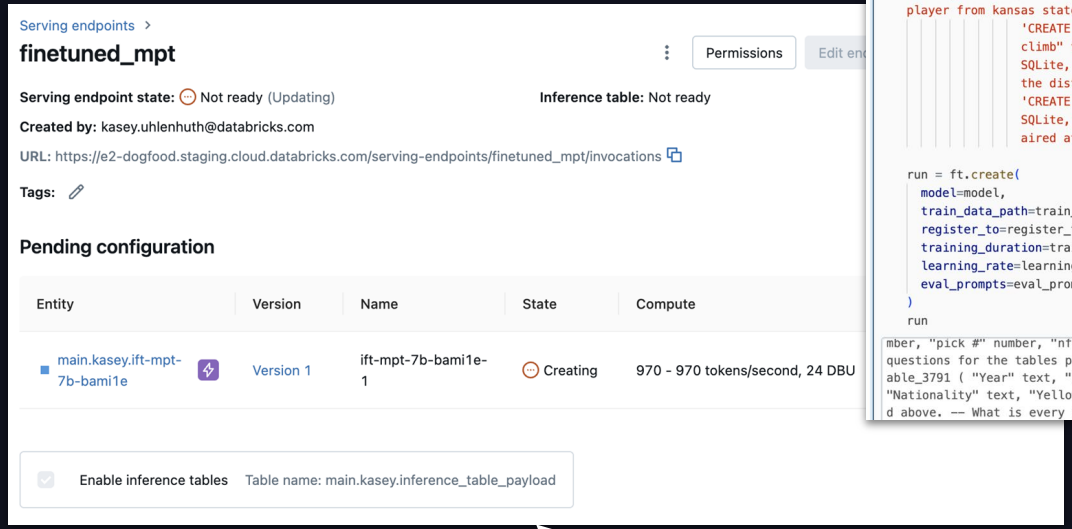


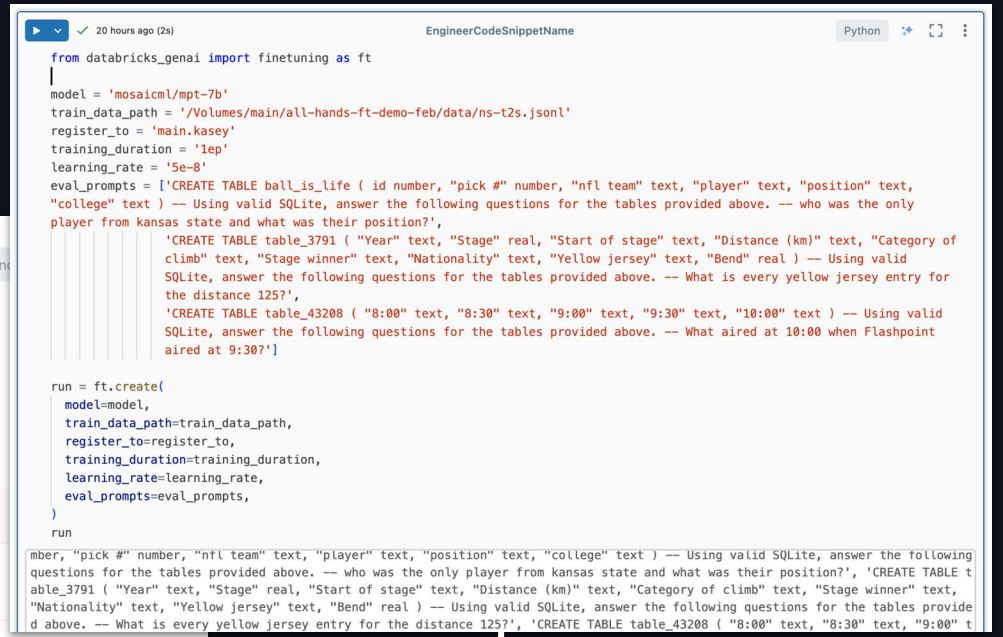
```
trainer = SFTTrainer(
model,
train_dataset=dataset['train'],
eval_dataset = dataset['test'],
dataset_text_field="text",
max_seq_length=256,
args=training_args,
)

# Initiate the training process
with mlflow.start_run(run_name= 'run_name_of_choice'):
trainer.train()
```

Fine-tune a model

Customize models on your proprietary data so you can maintain and improve quality over time

- **Databricks Fine-Tuning API (Private Preview now):** Simple tooling for fine-tuning common embeddings and instruction-following architectures.
- **Custom Code**: Fine-tune any GenAI model, using your standard Databricks workflow: GPU clusters, MLflow, notebooks/jobs...



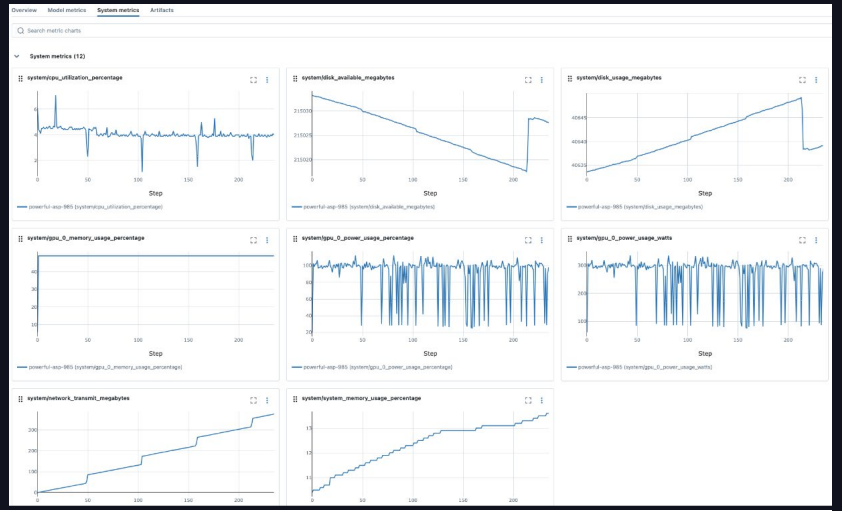


Easy to serve with FMAPI Provisioned Throughput

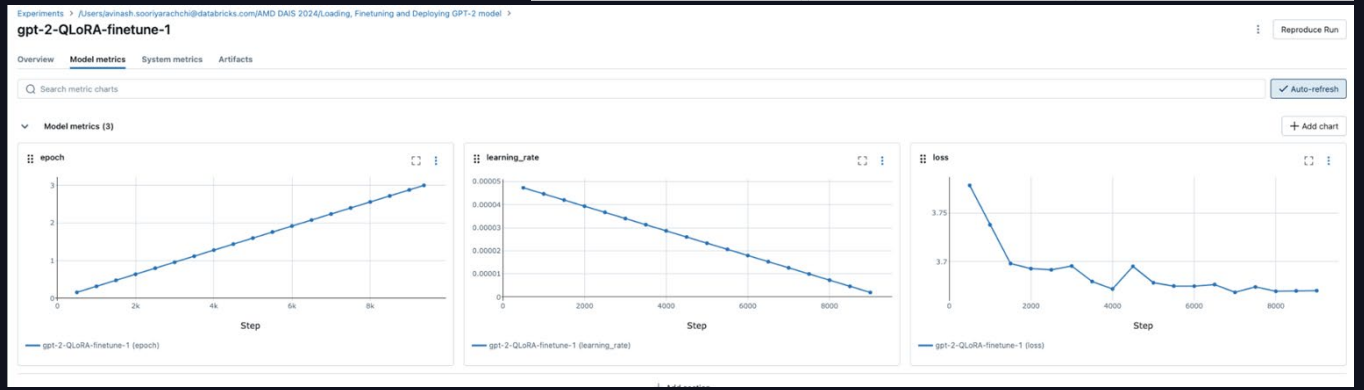
MLFlow for tracking and logging

1. System Metrics

```
import mlflow

mlflow.enable_system_metrics_logging()
```



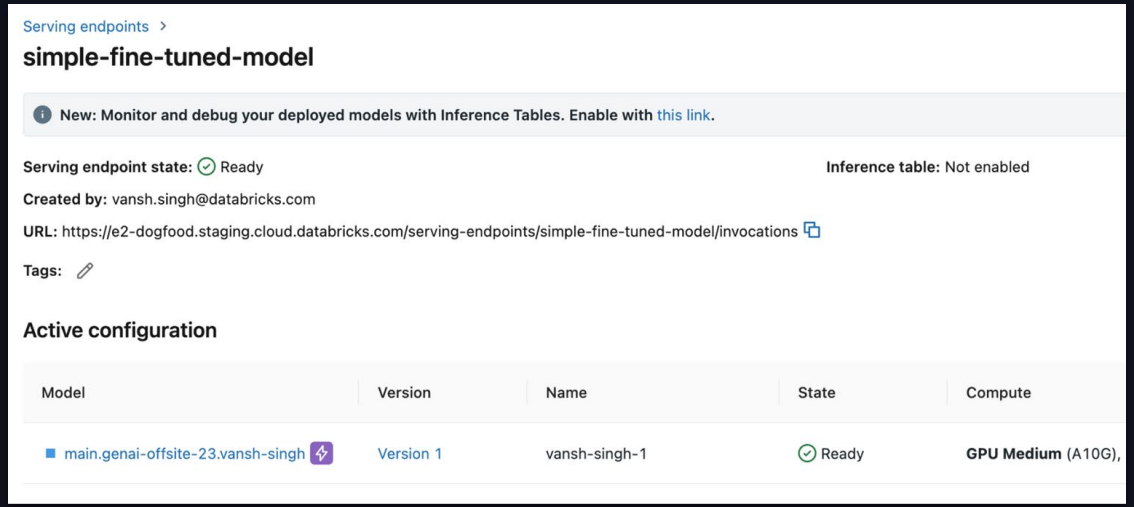
1. Training loss/ Metrics



MODEL SERVING

Model Deployment with Serverless Inference

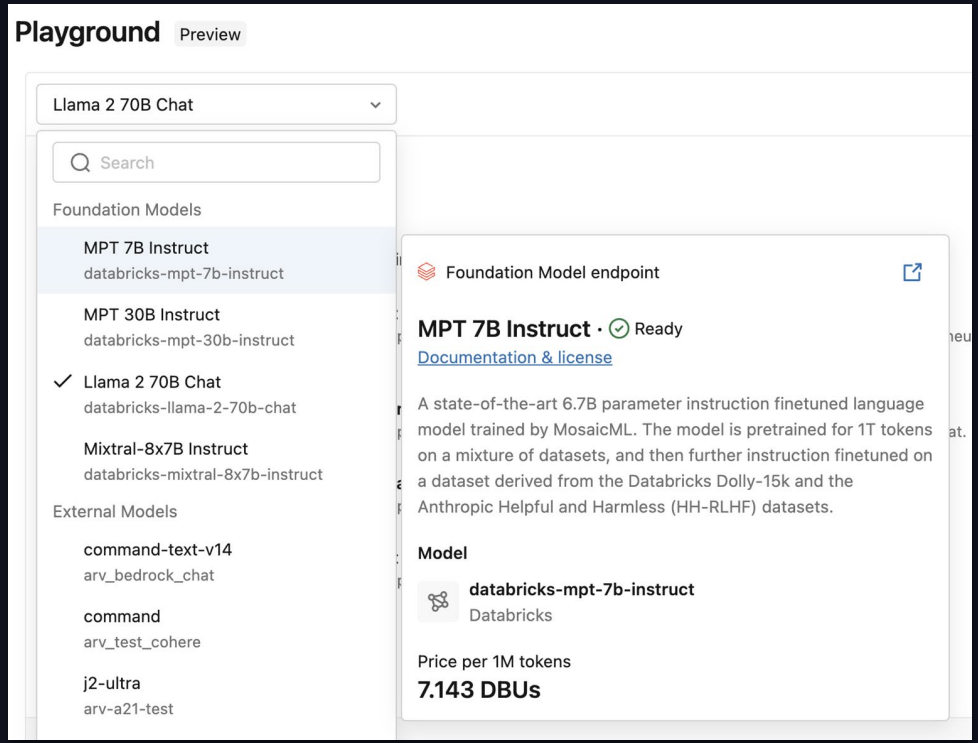
- Deploy the finetuned model as an API with Serverless GPU compute
- This includes any deep learning model including LLMs, Diffusion models and LLMs wrapped in chains (Langchain, LlamaIndex etc.)



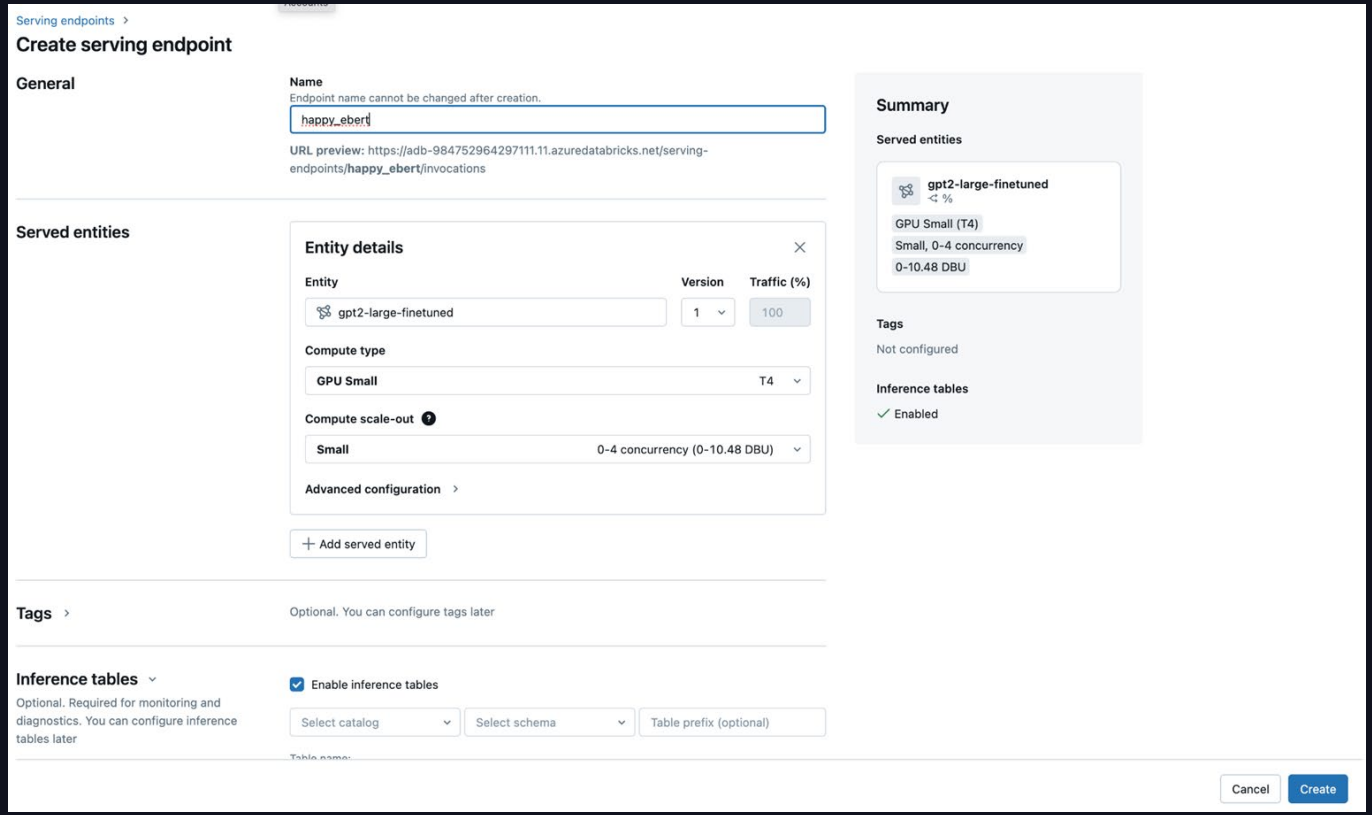
Foundation Model APIs

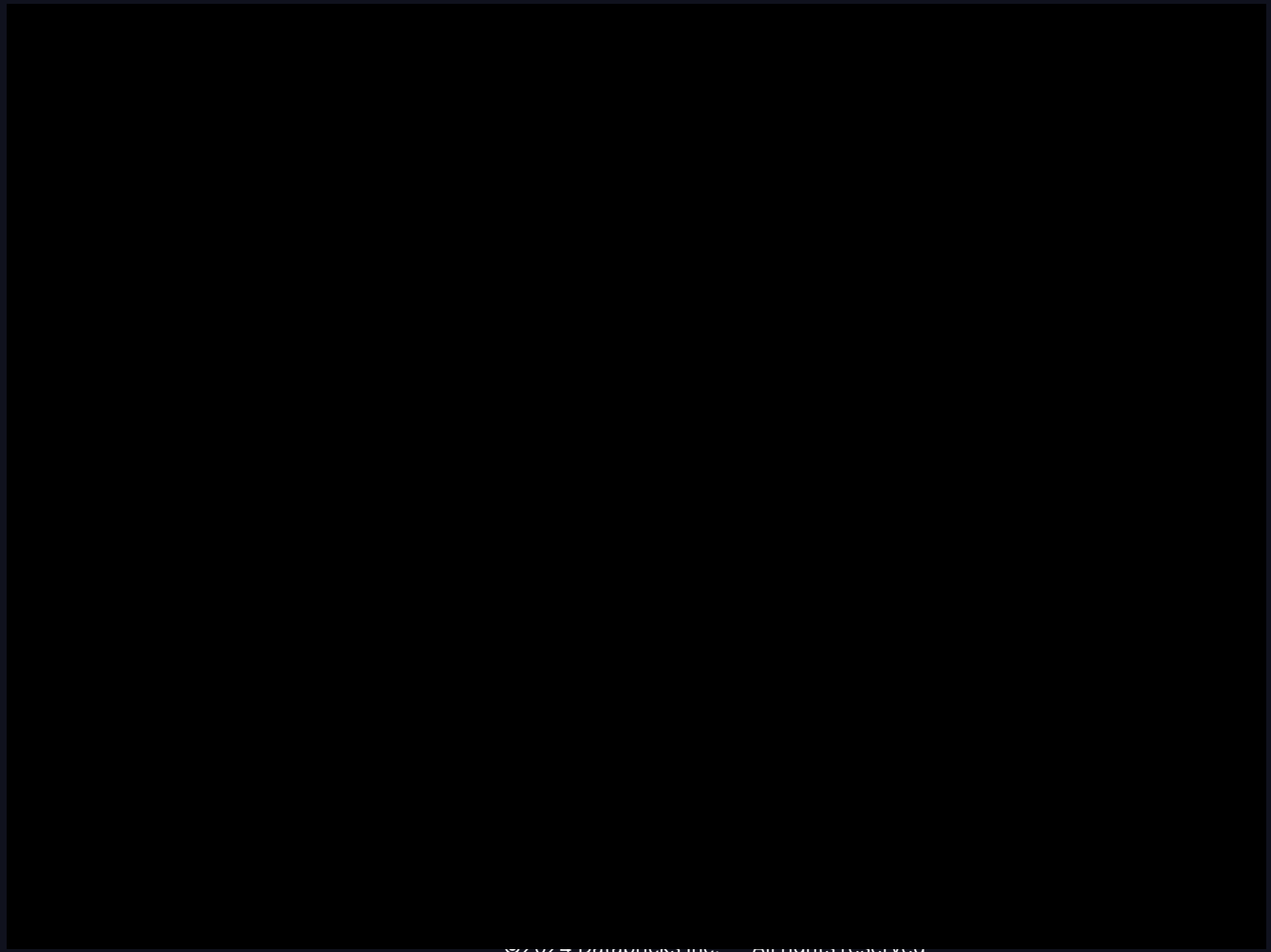
Get pay-per-token access to state-of-the-art open source models like the MPT and Llama families

- Use **pay-per-token** access for a curated set of foundation models
- Use **provisioned throughput** access to privately hosted foundation models on optimized infrastructure



Custom Model Deployment





SUMMARY

AMD + Databricks for Generative AI

- AMD hardware and software is ready to run your heaviest GPU workloads
- Run AMD GPU clusters on-prem or use Azure ND MI300X v5 instances
- Scale your pre-training using a dedicated fast RDMA network combined with AMD ROCm software and distributed ML frameworks
- Move your pre-trained models to Databricks for model lifecycle management tools
- Run fine-tuning and inference with Databricks to power your AI applications

DATA+AI SUMMIT BY databricks

THANK YOU