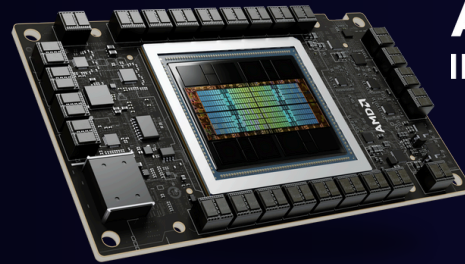


# Entering the Era of Choice in AI: Putting Dell™ PowerEdge™ XE9680 Server with AMD® Instinct™ MI300X Accelerators to the Test by Fine-tuning and Deploying Llama 2 70B Chat Model.

| March 2024

In this blog, Metrum AI™ will show you how to fine-tune large language models (LLMs), deploy 70B parameter models, and run a chatbot on the Dell™ PowerEdge™ XE9680 Server equipped with AMD® Instinct™ MI300X Accelerators.



**AMD**  
**INSTINCT**

**DELL**Technologies

**AMD**

**M** **TRUM** AI

 **Hugging Face**

With the release of the AMD Instinct MI300X Accelerator, we are now entering an era of choice for leading AI Accelerators that power today's generative AI solutions. Dell has paired the accelerators with its flagship PowerEdge XE9680 server for high performance AI applications. To put this leadership combination to the test, Metrum AI™ received early access and developed a fine-tuning stack with industry leading open-source components and deployed the Llama 2 70B Chat Model with FP16 precision in an enterprise chatbot scenario. In doing so, Metrum AI™ uncovered three critical value drivers:

- Deployed the Llama 2 70B parameter model on a single AMD Instinct MI300X Accelerator on the Dell PowerEdge XE9680 Server.
- Deployed eight concurrent instances of the model by utilizing all eight available AMD Instinct MI300X Accelerators on the Dell PowerEdge XE9680 Server.
- Fine-tuned the Llama 2 70B parameter model with FP16 precision on one Dell PowerEdge XE9680 Server with eight AMD Instinct MI300X accelerators.

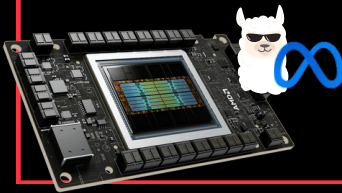
## INFERENCE

1

1x AMD® Instinct™  
MI300X Accelerator



Llama 2 70B Chat Model



Dell™ PowerEdge™  
XE9680 Server



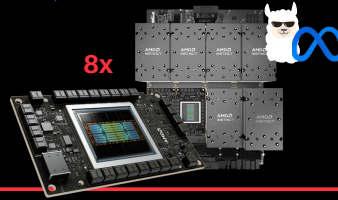
2

8x AMD® Instinct™  
MI300X Accelerators



8x Instance

Llama 2 70B Chat Model



Dell™ PowerEdge™  
XE9680 Server



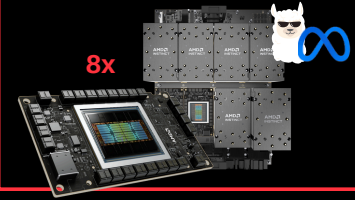
## FINE-TUNING

3

8x AMD® Instinct™  
MI300X Accelerators



Llama 2 70B Chat Model



Dell™ PowerEdge™  
XE9680 Server



This showcases industry leading total cost of ownership value for enterprises looking to fine-tune state of the art large language models with their own proprietary data, and deploy them on a single Dell PowerEdge XE9680 server equipped with AMD Instinct MI300X Accelerators.

*"The PowerEdge XE9680 paired with AMD Instinct MI300X Accelerators delivers industry leading capability for fine-tuning and deploying eight concurrent instances of the Llama 2 70B FP16 model on a single server."*



- Chetan Gadgil, CTO, Metrum AI

**To recreate, start with Dell PowerEdge XE9680 Server configurations as such.**

OS: Ubuntu 22.04.4 LTS  
Kernel version: 5.15.0-94-generic  
Docker Version: Docker version 25.0.3, build 4debf41  
ROCm™ version: 6.0.2  
Server: Dell™ PowerEdge™ XE9680  
GPU: 8x AMD® Instinct™ MI300X Accelerators

## Setup Steps

**1. Install the AMD® ROCm™ driver, libraries, and tools. Follow the detailed [installation instructions](#) for your Linux based platform.**

To ensure these installations are successful, check the GPU info using rocm-smi.

**2. Clone the vLLM GitHub repository for 0.3.2 version as below:**

```
git clone -b v0.3.2 https://github.com/vllm-project/vllm.git
```

**3. Build the Docker container from the Dockerfile.rocm file inside the cloned vLLM repository.**

```
cd vllm  
sudo docker build -f Dockerfile.rocm -t vllm-rocm:latest .
```

**4. Use the command below to start the vLLM ROCm docker container and open the container shell.**

```
sudo docker run -it \  
--name vllm \  
--network=host \  
--device=/dev/kfd \  
--device=/dev/dri \  
--shm-size 16G \  
--group-add=video \  
--workdir=/ \  
vllm-rocm:latest bash
```

**5. Use the command below to start the vLLM ROCm docker container and open the container shell.**

```
huggingface-cli login
```

Part 1.0 - Let's start by showcasing how you can run the Llama 2 70B Chat Model on one AMD Instinct MI300X Accelerator on the PowerEdge XE9680 server. Previously we would use two cutting edge GPUs to complete this task.

## INFERENCE

On Dell PowerEdge™ XE9680 Server  
with 1x AMD® Instinct™ MI300X Accelerator



Deploying Llama 2 70B Chat Model with vLLM 0.3.2 on a single AMD Instinct MI300X Accelerator with Dell PowerEdge XE9680 Server.

### Run vLLM Serving with Llama 2 70B Chat Model

1. Start the vLLM server for Llama 2 70B Chat model with FP16 precision loaded on a single AMD Instinct MI300X Accelerator.

```
python3 -m vllm.entrypoints.openai.api_server --model meta-llama/Llama-2-70b-chat-hf --dtype float16 --tensor-parallel-size 1
```

## 2. Execute the following curl request to verify if vLLM is successfully serving the model at the chat completion endpoint.

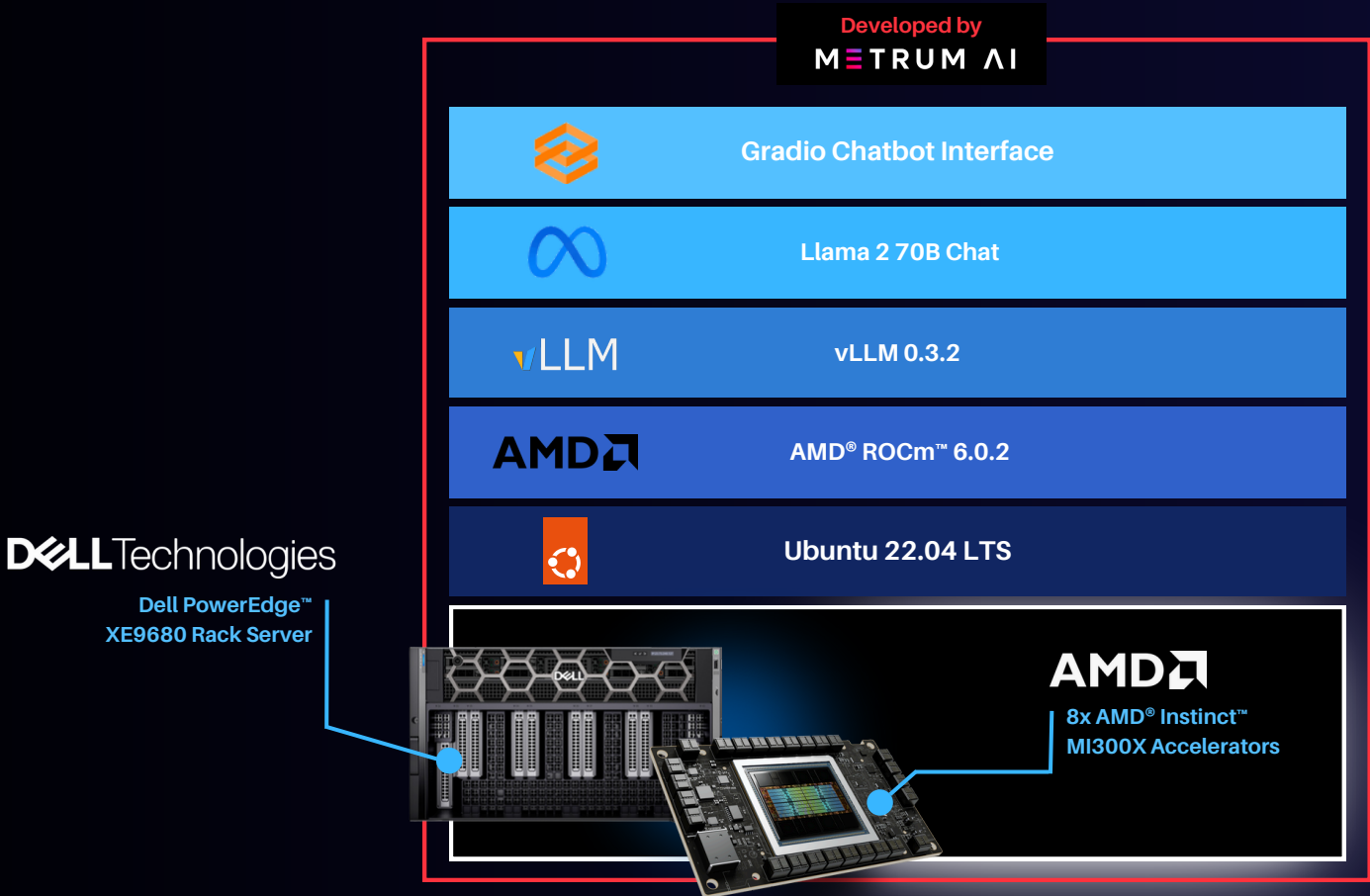
```
curl http://localhost:8000/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "meta-llama/Llama-2-70b-chat-hf",
    "max_tokens":256,
    "temperature":1.0,
    "messages": [
      {"role": "system", "content": "You are a helpful
assistant."},
      {"role": "user", "content": "Describe AMD ROCm in 180
words."}
    ]
  }'
```

The response should look as follows.

```
{"id":"cml-
42f932f6081e45fa8ce7a7212cb19adb","object":"chat.completion","create
d":1150766,"model":"meta-llama/Llama-2-70b-chat-hf","choices":
[{"index":0,"message":{"role":"assistant","content":" AMD ROCm
(Radeon Open Compute MTV) is an open-source software platform
developed by AMD for high-performance computing and deep learning
applications. It allows developers to tap into the massive parallel
processing power of AMD Radeon GPUs, providing faster performance
and more efficient use of computational resources. ROCm supports a
variety of popular deep learning frameworks, including TensorFlow,
PyTorch, and Caffe, and is designed to work seamlessly with AMD's
GPU-accelerated hardware. ROCm offers features such as low-level
hardware control, GPU Virtualization, and support for multi-GPU
configurations, making it an ideal choice for demanding workloads
like artificial intelligence, scientific simulations, and data
analysis. With ROCm, developers can take full advantage of AMD's GPU
capabilities and achieve faster time-to-market and better
performance for their
applications."},"finish_reason":"stop"}],"usage":
{"prompt_tokens":42,"total_tokens":237,"completion_tokens":195}}
```

Part 1.1. - Let's start by showcasing how you can run the Llama 2 70B Chat Model on one AMD Instinct MI300X Accelerator on the PowerEdge XE9680 server. Previously we would use two cutting edge GPUs to complete this task.

## Running Gradio Chatbot with Llama 2 70B Chat Model



This Gradio chatbot works by sending the user input query received through the user interface to the Llama 2 70B Chat Model being served using vLLM. The vLLM server is compatible with the OpenAI Chat API hence the request is sent in the OpenAI Chat API compatible format. The model generates the response based on the request which is sent back to the client. This response is displayed on the Gradio chatbot user interface.

## Deploying Gradio Chatbot

1. If not already done, follow the instructions in the Setup Steps section to install the AMD ROCm driver, libraries, and tools, clone the vLLM repository, build and start the vLLM ROCm Docker container, and request access to the Llama 2 Models from Meta.

2. Install the prerequisites for running the chatbot.

```
pip3 install -U pip
pip3 install openai==1.13.3 gradio==4.20.1
```

3. Log in to the Hugging Face CLI and enter your HuggingFace access token when prompted:

```
huggingface-cli login
```

4. Start the vLLM server for Llama 2 70B Chat model with data type FP16 on one AMD Instinct MI300X Accelerator.

```
python3 -m vllm.entrypoints.openai.api_server --model meta-llama/Llama-2-70b-chat-hf --dtype float16
```

5. Run the `gradio_openai_chatbot_webserver.py` from the `/app/vllm/examples` directory within the container with the default configurations.

```
cd /app/vllm/examples
python3 gradio_openai_chatbot_webserver.py --model meta-llama/Llama-2-70b-chat-hf
```

The Gradio chatbot will be running on the port 8001 and can be accessed using the URL <http://localhost:8001>. The query passed to the chatbot is "How does AMD ROCm contribute to enhancing the performance and efficiency of enterprise AI workflows?" The output conversation with the chatbot is shown below:

How does AMD ROCm contribute to enhancing the performance and efficiency of enterprise AI workflows?

AMD ROCm (Radeon Open Compute Platform) is a software platform designed to enhance the performance and efficiency of enterprise AI workflows by leveraging the power of AMD GPUs. ROCm provides a set of tools and libraries that enable developers to easily integrate AI workloads onto AMD GPUs, improving the performance and efficiency of their AI applications.

Here are some ways in which AMD ROCm contributes to enhancing the performance and efficiency of enterprise AI workflows:

1. **GPU Acceleration:** ROCm allows developers to offload compute-intensive AI workloads to AMD GPUs, which are designed to handle large amounts of data in parallel, making them ideal for AI workloads. By leveraging the massively parallel architecture of GPUs, ROCm enables faster processing of AI workloads, resulting in significant performance gains.
2. **Optimized Memory Access:** ROCm optimizes memory access patterns for AI workloads, minimizing data transfer between the GPU and CPU, and reducing memory latency. This results in improved performance and reduced power consumption, making it ideal for data-intensive AI applications.
3. **Scalability:** ROCm allows developers to scale their AI applications across multiple GPUs, providing the necessary resources to handle large and complex AI workloads. This enables enterprises to train larger models, process larger datasets, and perform more complex AI tasks.
4. **Flexibility:** ROCm supports a variety of popular AI frameworks and libraries, including TensorFlow, PyTorch, and Caffe. This allows developers to use their framework of choice and easily integrate ROCm into their existing AI workflows.
5. **Ease of Use:** ROCm provides a set of tools and libraries that simplify the development process, making it easier for developers to integrate AI workloads onto AMD GPUs. This reduces the time and effort required to develop and deploy AI applications, allowing enterprises to quickly adopt and deploy AI solutions.
6. **HPC Support:** ROCm is designed to work seamlessly with AMD's high-performance computing (HPC) solutions, enabling enterprises to leverage the power of GPU acceleration for HPC workloads. This allows enterprises to perform complex simulations, data analysis, and machine learning tasks with improved performance and efficiency.
7. **Security:** ROCm provides advanced security features, including secure boot, secure memory, and secure data transfer, ensuring that sensitive data is protected throughout the AI workflow. This is particularly important for enterprises in highly regulated industries, such as finance, healthcare, and government.

In summary, AMD ROCm enhances the performance and efficiency of enterprise AI workflows by providing a software platform that leverages the power of AMD GPUs, optimizes memory access, scales across multiple GPUs, supports a variety of frameworks and libraries, and provides ease of use, HPC support, and advanced security features.



Retry

Undo

Clear

Type a message...

SUBMIT

Use via API  · Built with Gradio 

## 6. To observe the GPU utilization, use the rocm-smi command as shown below.

```
===== ROCm System Management Interface =====
===== Concise Info =====
Device [Model : Revision] Temp Power Partitions SCLK MCLK Fan Perf PwrCap VRAM% GPU%
^[3m Name (20 chars) (Junction) (Socket) (Mem, Compute)
=====
0 [0x74a1 : 0x00] 43.0°C 444.0W NPS1, SPX 1665Mhz None 0% auto 750.0W 91% 100%
AMD Instinct MI300X
1 [0x74a1 : 0x00] 38.0°C 143.0W NPS1, SPX 132Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
2 [0x74a1 : 0x00] 38.0°C 149.0W NPS1, SPX 132Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
3 [0x74a1 : 0x00] 37.0°C 140.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
4 [0x74a1 : 0x00] 36.0°C 138.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
5 [0x74a1 : 0x00] 34.0°C 134.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
6 [0x74a1 : 0x00] 36.0°C 144.0W NPS1, SPX 131Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
7 [0x74a1 : 0x00] 35.0°C 138.0W NPS1, SPX 132Mhz 900Mhz 0% auto 750.0W 0% 0%
AMD Instinct MI300X
=====
===== End of ROCm SMI Log =====
```



## 7. Use the command below to access various vLLM serving metrics through the /metrics endpoint.

```
curl http://127.0.0.1:8000/metrics
```

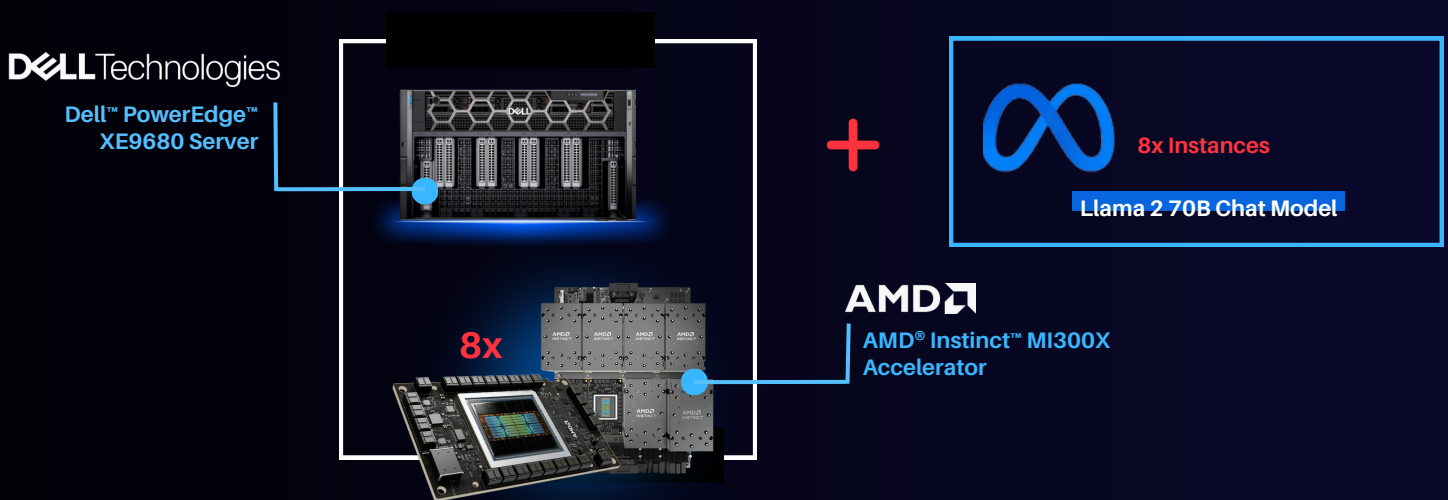
The output should look as follows.

```
# HELP exceptions_total_counter Total number of requested which
generated an exception
# TYPE exceptions_total_counter counter
# HELP requests_total_counter Total number of requests received
# TYPE requests_total_counter counter
requests_total_counter{method="POST",path="/v1/chat/completions"} 1
# HELP responses_total_counter Total number of responses sent
# TYPE responses_total_counter counter
responses_total_counter{method="POST",path="/v1/chat/completions"} 1
# HELP status_codes_counter Total number of response status codes
# TYPE status_codes_counter counter
status_codes_counter{method="POST",path="/v1/chat/completions",status_co
de="200"} 1
# HELP vllm:avg_generation_throughput_toks_per_s Average generation
throughput in tokens/s.
# TYPE vllm:avg_generation_throughput_toks_per_s gauge
vllm:avg_generation_throughput_toks_per_s{model_name="meta-llama/Llama-
2-70b-chat-hf"} 4.222076684555402
# HELP vllm:avg_prompt_throughput_toks_per_s Average prefill throughput
in tokens/s.
# TYPE vllm:avg_prompt_throughput_toks_per_s gauge
vllm:avg_prompt_throughput_toks_per_s{model_name="meta-llama/Llama-2-
70b-chat-hf"} 0.0
...
# HELP vllm:prompt_tokens_total Number of prefill tokens processed.
# TYPE vllm:prompt_tokens_total counter
vllm:prompt_tokens_total{model_name="meta-llama/Llama-2-70b-chat-hf"} 44
...
vllm:time_per_output_token_seconds_count{model_name="meta-llama/Llama-2-
70b-chat-hf"} 136.0
vllm:time_per_output_token_seconds_sum{model_name="meta-llama/Llama-2-
70b-chat-hf"} 32.18783768080175
...
vllm:time_to_first_token_seconds_count{model_name="meta-llama/Llama-2-
70b-chat-hf"} 1.0
vllm:time_to_first_token_seconds_sum{model_name="meta-llama/Llama-2-70b-
chat-hf"} 0.2660619909875095
```

Part 2: Now that we have deployed the Llama 2 70B Chat Model on a single GPU, let's take full advantage of the Dell PowerEdge XE9680 server and deploy eight concurrent instances of the Llama 2 70B Chat Model with FP16 precision. To handle more simultaneous users and generate higher throughput, the 8x AMD Instinct MI300X Accelerators can be leveraged to deploy 8 vLLM serving deployments in parallel.

## INFERENCE

On Dell PowerEdge™ XE9680 Server  
with 8x AMD® Instinct™ MI300X Accelerators



Serving Llama 2 70B Chat model with FP16 precision using vLLM 0.3.2 on 8x AMD Instinct MI300X Accelerators with the PowerEdge XE9680 Server.

To enable the multi GPU vLLM deployment, we use a Kubernetes based stack. The stack consists of a Kubernetes Deployment with 8 vLLM serving replicas and a Kubernetes Service to expose all vLLM serving replicas through a single endpoint. The Kubernetes Service utilizes a round robin based strategy to distribute the requests across the vLLM serving replicas.

### Prerequisites

1. Any Kubernetes distribution on the server.
2. AMD GPU device plugins for Kubernetes setup on the installed Kubernetes distribution.
3. A Kubernetes secret that grants access to the container registry, facilitating Kubernetes deployment.

## Deploying the multi vLLM serving on 8x AMD Instinct MI300X Accelerators.

1. If not already done, follow the instructions in the Setup Steps section to install the AMD ROCm driver, libraries, and tools, clone the vLLM repository, build the vLLM ROCm Docker container, and request access to the Llama 2 Models from Meta. Push the built vllm-rocm:latest image to the container registry of your choice.
2. Create a deployment yaml file "multi-vllm.yaml" based on the sample provided below.

```
# vllm deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vllm-serving
  namespace: default
  labels:
    app: vllm-serving
spec:
  selector:
    matchLabels:
      app: vllm-serving
  replicas: 8
  template:
    metadata:
      labels:
        app: vllm-serving
    spec:
      containers:
      - name: vllm
        image: container-registry/vllm-rocm:latest # update the
container registry name
        args: [
          "python3", "-m", "vllm.entrypoints.openai.api_server",
          "--model", "meta-llama/Llama-2-70b-chat-hf"
```

```

    ]
    env:
      - name: HUGGING_FACE_HUB_TOKEN
        value: "" # add your huggingface token with Llama 2
models access
  resources:
    requests:
      cpu: 15
      memory: 150G
      amd.com/gpu: 1 # each replica is allocated 1 GPU
    limits:
      cpu: 15
      memory: 150G
      amd.com/gpu: 1
  imagePullSecrets:
    - name: cr-login # kubernetes container registry secret
---
# nodeport service with round robin load balancing
apiVersion: v1
kind: Service
metadata:
  name: vllm-serving-service
  namespace: default
spec:
  selector:
    app: vllm-serving
  type: NodePort
  ports:
    - name: vllm-endpoint
      port: 8000
      targetPort: 8000
      nodePort: 30800 # the external port endpoint to access the
serving

```

**3. Deploy the multi vLLM serving using the deployment configuration with kubectl. This will deploy eight replicas of vLLM serving using the Llama 2 70B Chat model with FP16 precision.**

```
kubectl apply -f multi-vllm.yaml
```

#### 4. Execute the following curl request to verify whether the model is being successfully served at the chat completion endpoint at port 30800.

```
curl http://localhost:30800/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "meta-llama/Llama-2-70b-chat-hf",
    "max_tokens":256,
    "temperature":1.0,
    "messages": [
      {"role": "system", "content": "You are a helpful
assistant."},
      {"role": "user", "content": "Describe AMD ROCm in 180
words."}
    ]
  }'
```

The response should look as follows:

```
{"id":"cml-
42f932f6081e45fa8ce7dnjmc769ab","object":"chat.completion","created
":1150766,"model":"meta-llama/Llama-2-70b-chat-hf","choices":
[{"index":0,"message":{"role":"assistant","content":" AMD ROCm
(Radeon Open Compute MTV) is an open-source software platform
developed by AMD for high-performance computing and deep learning
applications. It allows developers to tap into the massive parallel
processing power of AMD Radeon GPUs, providing faster performance
and more efficient use of computational resources. ROCm supports a
variety of popular deep learning frameworks, including TensorFlow,
PyTorch, and Caffe, and is designed to work seamlessly with AMD's
GPU-accelerated hardware. ROCm offers features such as low-level
hardware control, GPU Virtualization, and support for multi-GPU
configurations, making it an ideal choice for demanding workloads
like artificial intelligence, scientific simulations, and data
analysis. With ROCm, developers can take full advantage of AMD's GPU
capabilities and achieve faster time-to-market and better
performance for their
applications."},"finish_reason":"stop"}],"usage":
{"prompt_tokens":42,"total_tokens":237,"completion_tokens":195}}
```

5. We used load testing tools similar to [Apache Bench](#) to simulate concurrent user requests to the serving endpoint. The screenshot below showcases the output of rocm-smi while Apache Bench is running 2048 concurrent requests.

```

===== ROCm System Management Interface =====
===== Concise Info =====
Device [Model : Revision] Temp Power Concise Info SCLK MCLK Fan Perf PwrCap VRAM% GPU%
      Name (20 chars) (Junction) (Socket) (Mem, Compute)
=====
0 [0x74a1 : 0x00] 43.0°C 444.0W NPS1, SPX 1596Mhz None 0% auto 750.0W 91% 98%
  AMD Instinct MI300X
1 [0x74a1 : 0x00] 45.0°C 447.0W NPS1, SPX 1574Mhz None 0% auto 750.0W 91% 99%
  AMD Instinct MI300X
2 [0x74a1 : 0x00] 45.0°C 452.0W NPS1, SPX 1577Mhz None 0% auto 750.0W 91% 99%
  AMD Instinct MI300X
3 [0x74a1 : 0x00] 43.0°C 451.0W NPS1, SPX 1627Mhz None 0% auto 750.0W 91% 100%
  AMD Instinct MI300X
4 [0x74a1 : 0x00] 45.0°C 437.0W NPS1, SPX 1601Mhz None 0% auto 750.0W 91% 99%
  AMD Instinct MI300X
5 [0x74a1 : 0x00] 42.0°C 435.0W NPS1, SPX 1616Mhz None 0% auto 750.0W 91% 100%
  AMD Instinct MI300X
6 [0x74a1 : 0x00] 45.0°C 444.0W NPS1, SPX 1513Mhz None 0% auto 750.0W 91% 96%
  AMD Instinct MI300X
7 [0x74a1 : 0x00] 45.0°C 454.0W NPS1, SPX 1562Mhz None 0% auto 750.0W 91% 100%
  AMD Instinct MI300X
=====
===== End of ROCm SMI Log =====

```

Part 3: Now that we have deployed the Llama 2 70B Chat model on both one GPU and eight concurrent GPUs, let's try fine-tuning Llama 2 70B Chat with Hugging Face Accelerate.

## FINE-TUNING

On Dell PowerEdge™ XE9680 Server  
with 8x AMD® Instinct™ MI300X Accelerators

Dell Technologies

Dell™ PowerEdge™  
XE9680 Server



+

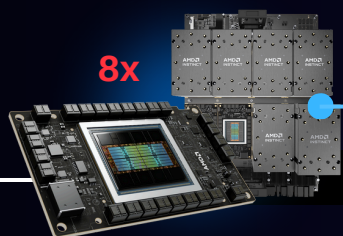


8x Instances

Llama 2 70B Chat Model



Hugging Face Accelerate

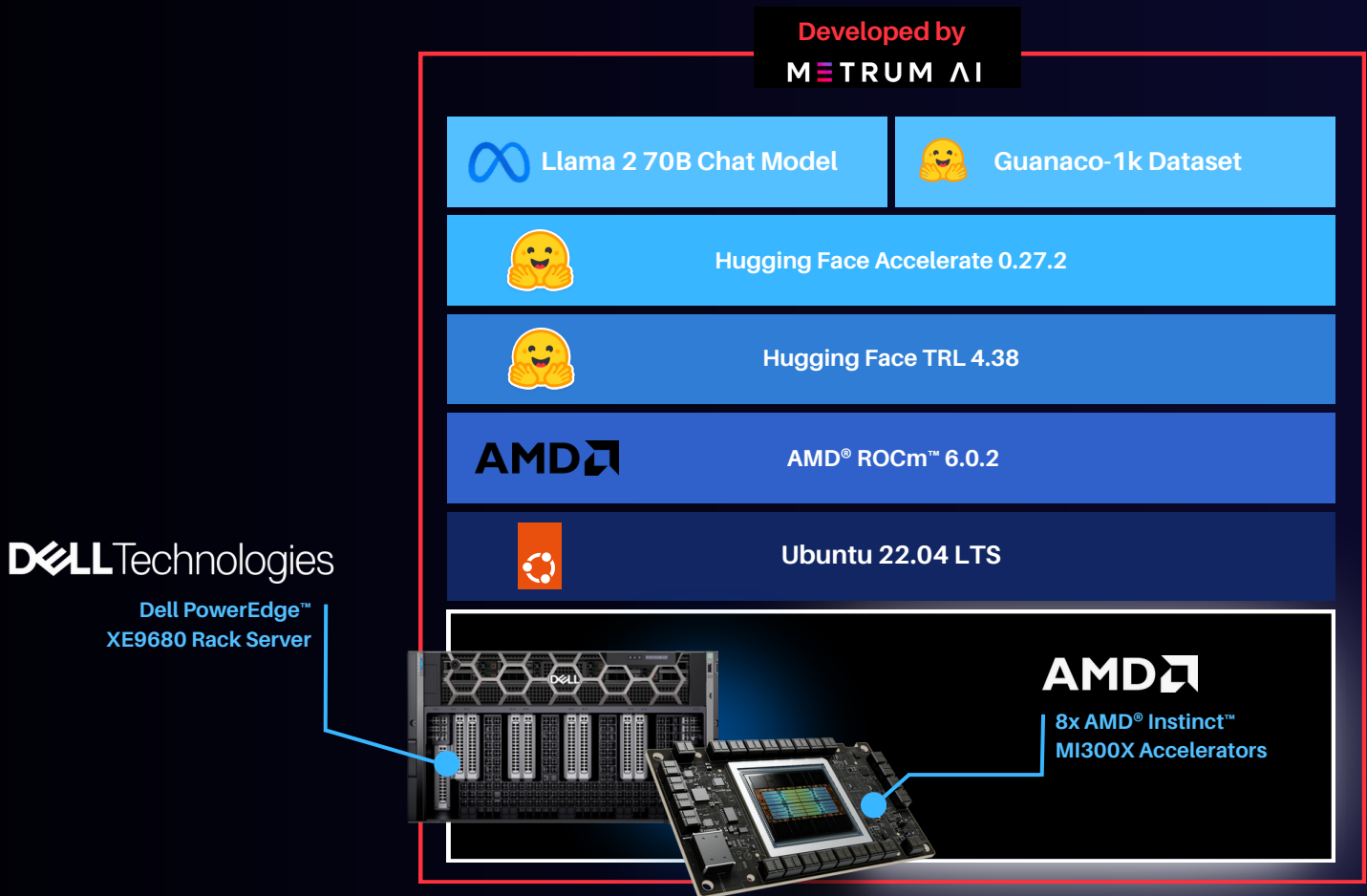


8x

AMD

AMD® Instinct™ MI300X  
Accelerator

## Fine-tuning



As shown above, the fine-tuning software stack begins with the AMD ROCm PyTorch image serving as the base, offering a tailored PyTorch library for optimal fine-tuning. Leveraging the Hugging Face Transformers library alongside Hugging Face Accelerate, facilitates multi-GPU fine-tuning capabilities. The Llama 2 70B Chat model will be fine-tuned with FP16 precision, utilizing the Guanaco-1k dataset from Hugging Face on eight AMD Instinct MI300X Accelerators.

In this scenario, we will perform full parameter fine-tuning of the Llama 2 70B Chat Model. While you can also implement fine-tuning using optimized techniques such as Low-Rank Adaptation of Large Language Models (LoRA) on accelerators with smaller memory footprints, performance tradeoffs exist on specific complex objectives. These nuances are addressed by full parameter fine-tuning methods, which generally require accelerators that support significant memory requirements.

## Fine-tuning Llama 2 70B Chat on 8x AMD Instinct MI300X Accelerators.

Fine-tune the Llama 2 70B Chat Model with FP16 precision for question and answer tasks by utilizing the [mlabonne/guanaco-llama2-1k](#) dataset on the 8X AMD Instinct MI300X Accelerators.

**1. If not already done, install the AMD ROCm driver, libraries, and tools and request access to the Llama 2 Models from Meta following the instructions in the Setup Steps section.**

**2. Start the fine-tuning docker container with the AMD ROCm PyTorch base image.**

The below command opens a shell within the docker container.

```
sudo docker run -it \  
--name fine-tuning \  
--network=host \  
--device=/dev/kfd \  
--device=/dev/dri \  
--shm-size 16G \  
--group-add=video \  
--workdir=/ \  
rocm/pytorch:rocm6.0.2_ubuntu22.04_py3.10_pytorch_2.1.2 bash
```

**3. Install the necessary Python prerequisites.**

```
pip3 install -U pip  
pip3 install transformers==4.38.2 trl==0.7.11 datasets==2.18.0
```

**4. Log in to Hugging Face CLI and enter your HuggingFace access token when prompted.**

```
huggingface-cli login
```

**5. Import the required Python packages.**

```
from datasets import load_dataset  
from transformers import (  
    AutoModelForCausalLM,  
    AutoTokenizer,  
    TrainingArguments,  
    pipeline  
)  
from trl import SFTTrainer
```



## 6. Load the Llama 2 70B Chat Model and the mlabonne/guanaco-llama2-1k dataset from Hugging Face.

```
# load the model and tokenizer
base_model_name = "meta-llama/Llama-2-70b-chat-hf"

# tokenizer parameters
llama_tokenizer = AutoTokenizer.from_pretrained(base_model_name,
trust_remote_code=True)
llama_tokenizer.pad_token = llama_tokenizer.eos_token
llama_tokenizer.padding_side = "right"

# load the based model
base_model = AutoModelForCausalLM.from_pretrained(
    base_model_name,
    device_map="auto",
)
base_model.config.use_cache = False
base_model.config.pretraining_tp = 1

# load the dataset from huggingface
dataset_name = "mlabonne/guanaco-llama2-1k"
training_data = load_dataset(dataset_name, split="train")
```

## 7. Define fine-tuning configurations and start fine-tuning for 1 epoch. The fine tuned model will be saved in finetuned\_llama2\_70b directory.

```
# fine tuning parameters
train_params = TrainingArguments(
    output_dir="./runs",
    num_train_epochs=1, # fine tuning for 1 epochs
    per_device_train_batch_size=8 # setting per GPU batch size
)
```

```

# define the trainer
fine_tuning = SFTTrainer(
    model=base_model,
    train_dataset=training_data,
    dataset_text_field="text",
    tokenizer=llama_tokenizer,
    args=train_params,
    max_seq_length=512
)

# start the fine tuning run
fine_tuning.train()

# save the fine tuned model
fine_tuning.model.save_pretrained("finetuned_llama2_70b")
print("Fine-tuning completed")

```

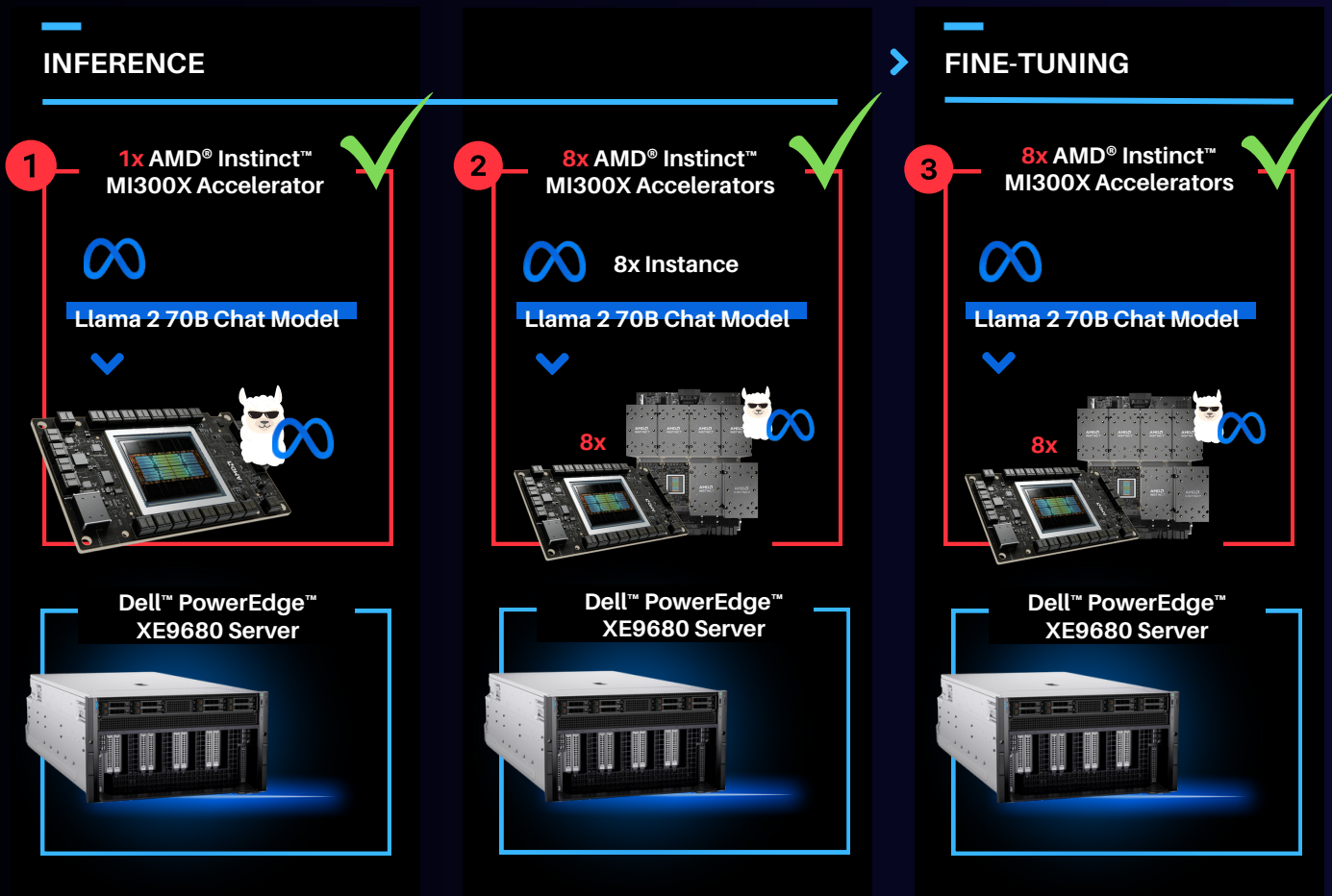
## 8. Use the `rocm-smi` command to observe GPU utilization while fine-tuning.

```

===== ROCm System Management Interface =====
===== Concise Info =====
Device [Model : Revision] Temp Power Partitions SCLK MCLK Fan Perf PwrCap VRAM% GPU%
      Name (20 chars) (Junction) (Socket) (Mem, Compute)
=====
0 [0x74a1 : 0x00] 47.0°C 268.0W NPS1, SPX 2100Mhz None 0% auto 750.0W 42% 100%
  AMD Instinct MI300X
1 [0x74a1 : 0x00] 48.0°C 254.0W NPS1, SPX 2108Mhz None 0% auto 750.0W 51% 100%
  AMD Instinct MI300X
2 [0x74a1 : 0x00] 49.0°C 257.0W NPS1, SPX 2104Mhz None 0% auto 750.0W 51% 100%
  AMD Instinct MI300X
3 [0x74a1 : 0x00] 49.0°C 255.0W NPS1, SPX 2100Mhz None 0% auto 750.0W 51% 100%
  AMD Instinct MI300X
4 [0x74a1 : 0x00] 46.0°C 256.0W NPS1, SPX 2079Mhz 1300Mhz 0% auto 750.0W 51% 100%
  AMD Instinct MI300X
5 [0x74a1 : 0x00] 44.0°C 626.0W NPS1, SPX 1312Mhz 1300Mhz 0% auto 750.0W 51% 100%
  AMD Instinct MI300X
6 [0x74a1 : 0x00] 39.0°C 244.0W NPS1, SPX 2100Mhz None 0% auto 750.0W 51% 100%
  AMD Instinct MI300X
7 [0x74a1 : 0x00] 38.0°C 238.0W NPS1, SPX 2100Mhz None 0% auto 750.0W 60% 100%
  AMD Instinct MI300X
=====
===== End of ROCm SMI Log =====

```

## Summary



Dell PowerEdge XE9680 Server equipped with AMD Instinct MI300X Accelerators offers enterprises industry leading infrastructure to create custom AI solutions using their proprietary data. In this blog, we showcased how enterprises deploying applied AI can take advantage of this solution in three critical use cases:

- Deploying the entire 70B parameter model on a single AMD Instinct MI300X Accelerator in Dell PowerEdge XE9680 Server
- Deploying eight concurrent instances of the model, each running on one of eight AMD Instinct MI300X accelerators on the Dell PowerEdge XE9680 Server
- Fine-tuning the 70B parameter model with FP16 precision on one PowerEdge XE9680 with all eight AMD Instinct MI300X accelerators

Metrum AI is excited to see continued advancements from Dell, AMD, and Hugging Face on hardware and software optimizations in the future.

## Additional Criteria for IT Decision Makers

### What is fine-tuning, and why is it critical for enterprises?

Fine-tuning enables enterprises to develop custom models with their proprietary data by leveraging the knowledge already encoded in pre-trained models. As a result, fine-tuning requires less labeled data and time for training compared to training a model from scratch, making it a more efficient approach for achieving competitive performance, particularly in the quantity of computational resources used and training time.

### Why is memory footprint critical for LLMs?

Large language models often have enormous numbers of parameters, leading to significant memory requirements. When working with LLMs, it is essential to ensure that the GPU has sufficient memory to store these parameters so that the model can run efficiently. In addition to model parameters, large language models require substantial memory to store input data, intermediate activations, and gradients during training or inference, and insufficient memory can lead to data loss or performance degradation.

### Why is the Dell PowerEdge XE9680 Server with AMD Instinct MI300X Accelerators well-suited for LLMs?

Designed especially for AI tasks, Dell PowerEdge XE9680 Server is a robust data-processing server equipped with eight AMD Instinct MI300X accelerators, making it well-suited for AI-workloads, especially for those involving training, fine-tuning, and conducting inference with LLMs. AMD Instinct MI300X Accelerator is a high-performance AI accelerator intended to operate in groups of eight within AMD's generative AI platform.

Running inference, specifically with a Large Language Model (LLM), requires approximately 1.2 times the memory occupied by the model on a GPU. In FP16 precision, the model memory requirement can be estimated as 2 bytes per parameter multiplied by the number of model parameters. For example, the Llama 2 70B model with FP16 precision requires a minimum of 168 GB of GPU memory to run inference. With 192 GB of GPU memory, a single AMD Instinct MI300X Accelerator can host an entire Llama 2 70B parameter model for inference. It is optimized for LLMs and can deliver up to 10.4 Petaflops of performance (BF16/FP16) with 1.5TB of total HBM3 memory for a group of eight accelerators.

Copyright © 2024 Metrum AI, Inc. All Rights Reserved. This project was commissioned by Dell Technologies. Dell and other trademarks are trademarks of Dell Inc. or its subsidiaries. AMD, Instinct™, ROCm™, and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other product names are the trademarks of their respective owners.

**\*\*\*DISCLAIMER** - Performance varies by hardware and software configurations, including testing conditions, system settings, application complexity, the quantity of data, batch sizes, software versions, libraries used, and other factors. The results of performance testing provided are intended for informational purposes only and should not be considered as a guarantee of actual performance.