AMD × Hugging Face

Deploying Optimized LLMs on AMD GPUs

Seungrok Jung
AI Group at AMD

AMD
together we advance_

Agenda

1. AMD Software and hardware for LLMs
2. LLM inference challenges and optimization directions
3. Commercial LLM serving deployment with Hugging Face
4. (Demo) Running Llama 3.1-405B
5. Useful AMD ROCm™ software resources & call for contributions
6. QA

AMD Software abstraction for LLMs

LLM libraries and toolkits 🤗 vLLM

PyTorch · TensorFlow · ONNX · Triton · deepspeed · OpenXLA

Ecosystem

Libraries

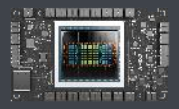
Compilers and Tools

Runtime

AMD AI Software

AMD ROCm

AMD GPUs



AMD Instinct™ Accelerators

AMD Radeon™ Graphics

AMD compute platforms

AMD Instinct™ Platform

Leading Memory and Competitive AI Performance to Power Generative AI Workloads

8x AMD Instinct™ MI300X Accelerators

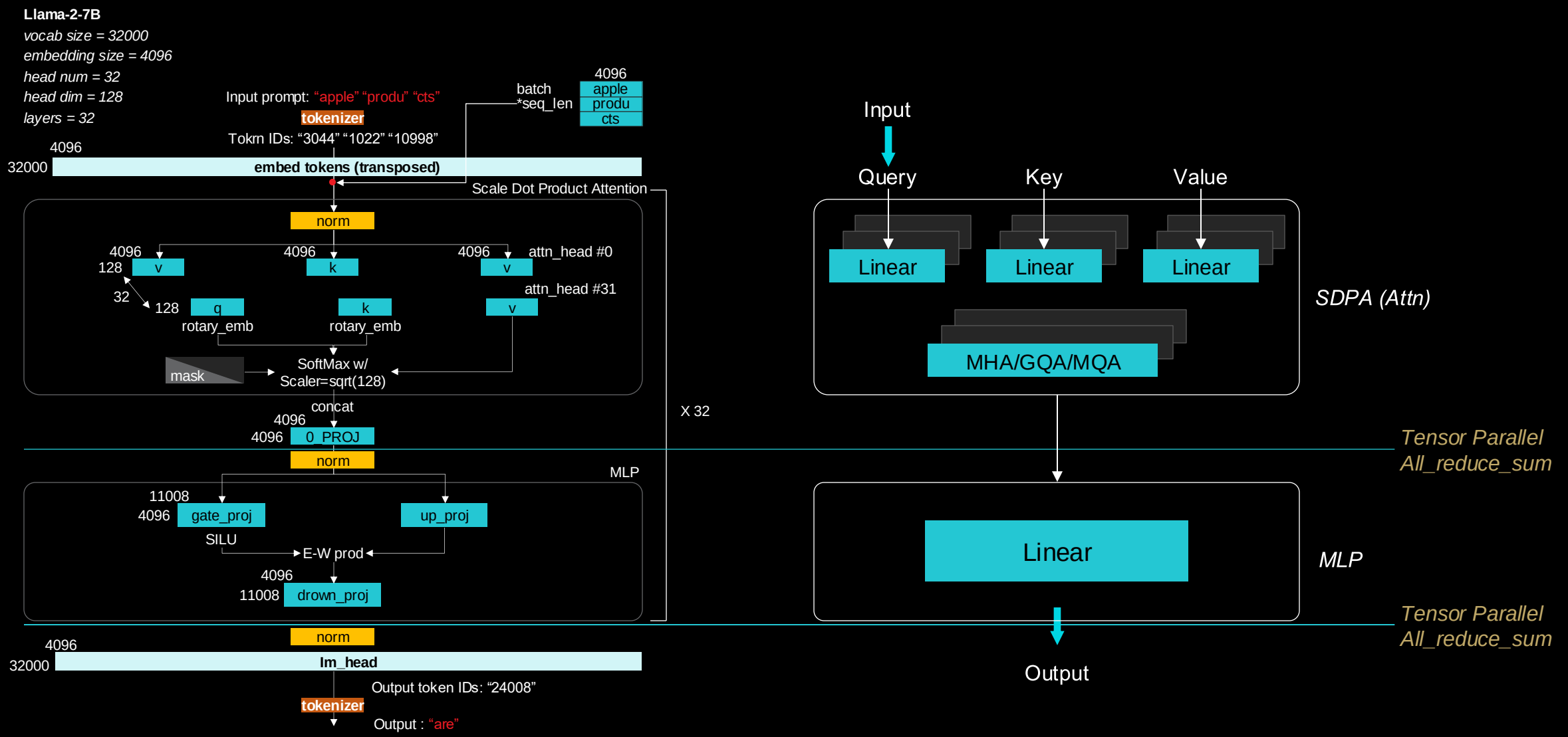
AMD Instinct™ MI300X Platform

1.5 TB HBM3 memory	10.4 PF FP16 / BF16 FLOPS
896 GB/s Aggregate bi-directional bandwidth	448 GB/s Single node ring bandwidth
Up to 400 GbE NIC / GPU	PCIe® Gen 5 128 GB/s

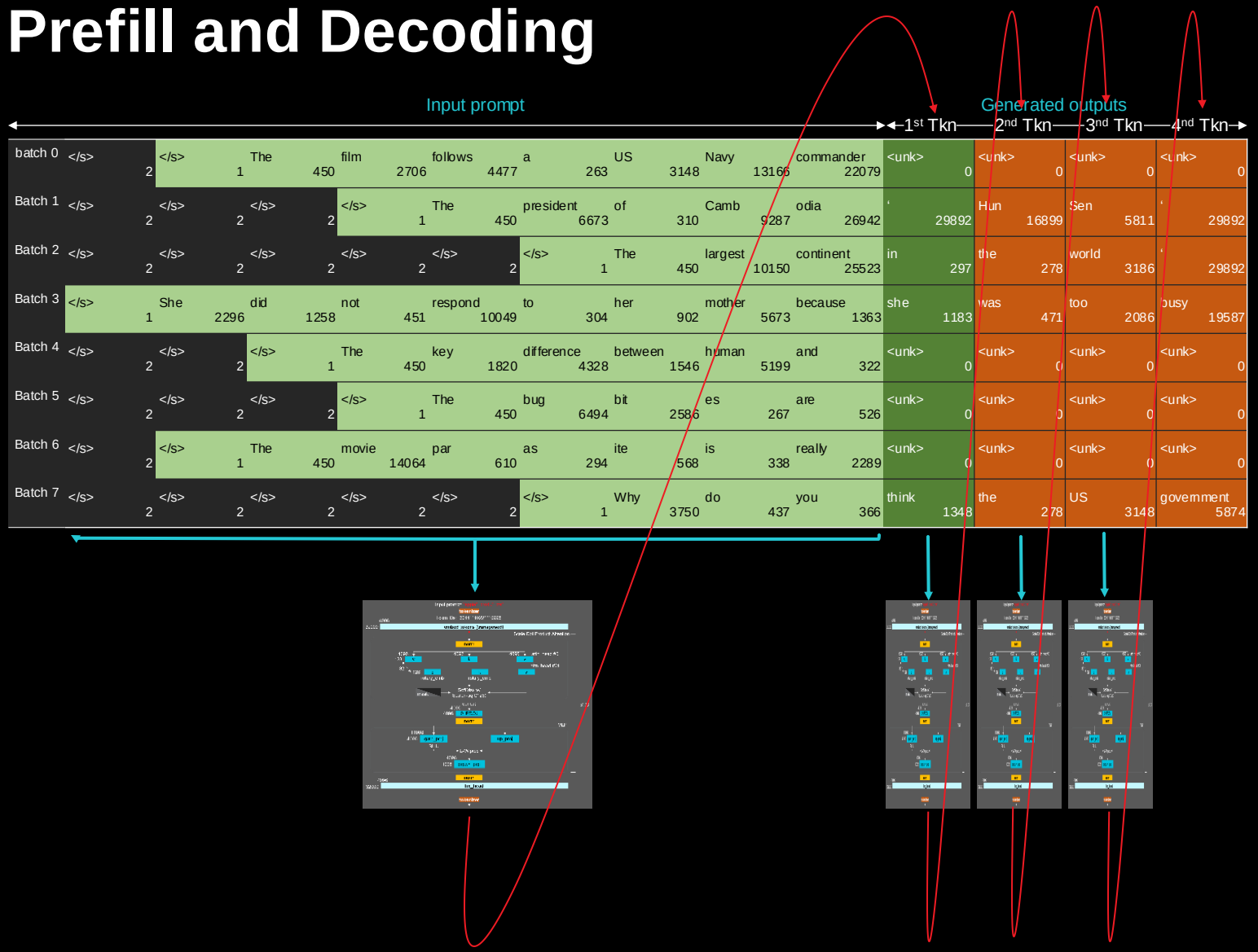
Agenda

1. AMD Software and hardware for LLMs
2. LLM inference challenges and optimization directions
3. Commercial LLM serving deployment with Hugging Face
4. (Demo) Running Llama 3.1-405B
5. Useful AMD ROCm™ software resources & call for contributions
6. QA

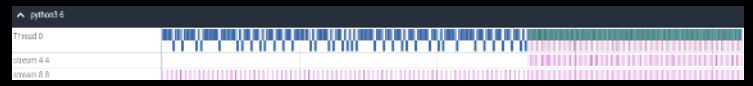
Typical LLM Model Architecture: Llama 7B



Prefill and Decoding



QWEN 1.5 110B, TP4, BS16, IN2048, OUT128



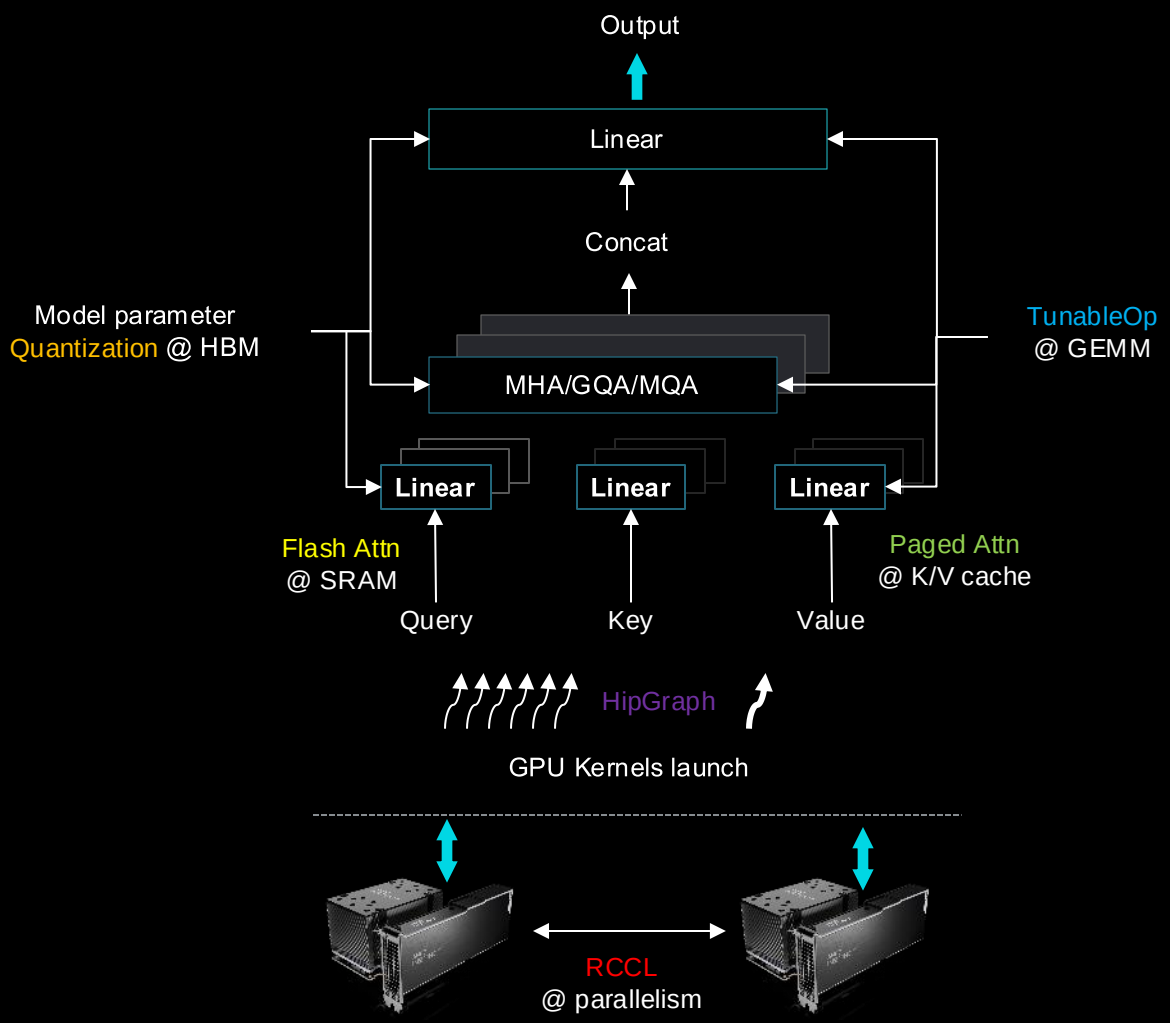
PREFILL
1 token x 80 layers

Compute-bound
(Large GeMM)

DECODING
128 tokens
x 80 layers/token

Memory-bound
(Tall&Skinny
GeMM)

Typical LLM Inference Optimizations



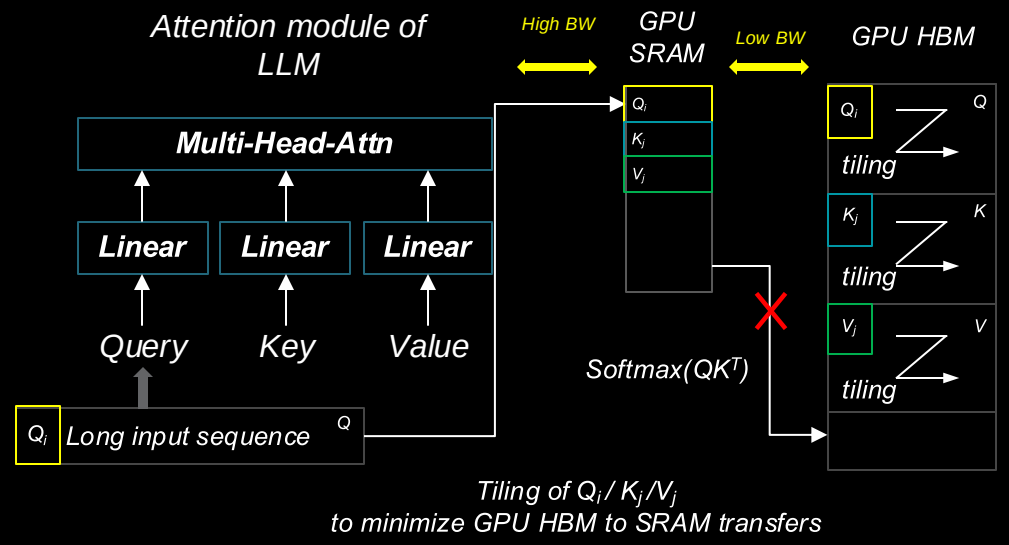
Flash Attention, Xformers	• Tiling of input sequence in GPU SRAM to reduce HBM data movement
Paged Attention	• Partitioned KV cache into fixed size blocks to reduce memory usage
GEMM Optimization – PyTorch TunableOp	• Automatic selection of the best performing GEMM kernels
Graph Optimization – HipGraph	• Launch multiple kernels through a single CPU operation
Collective communication – RCCL	• Collective Ops across multiple devices to support Tensor/ Pipeline parallel
Quantization – GPTQ, Bitsandbytes	• Weight-only compression to reduce HBM memory footprint

Prefill Optimization – Flash Attention

Algorithm Description

- Tiling of Attention Q/K/V to smaller blocks to minimize GPU HBM to SRAM transfers
- Fused computation of partial $Q_i K_j V_j$ to prevent frequent partial *Softmax(QK^T)* writeback to GPU HBM
- Takeaway: **Lower prefill latency**. Most effective for long and large batch sequences

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



Basic Usage

1 Install AMD ROCm™ flash attention

```
git clone https://github.com/ROCm/flash-attention
```

2 Utilize FAv2 with Hugging Face transformers Lib

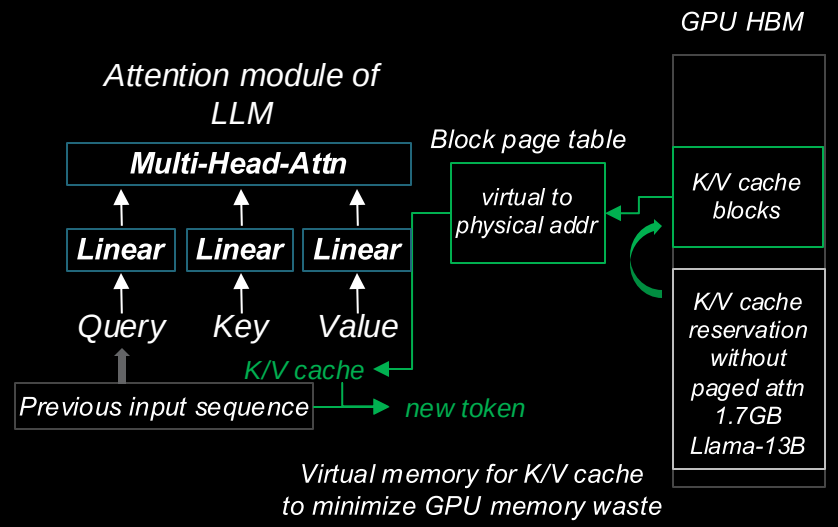
```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16, attn_implementation="flash_attention_2")
```

3 Leverage vLLM or Hugging Face TGI LLM serving framework

Decode Optimization – Paged Attention

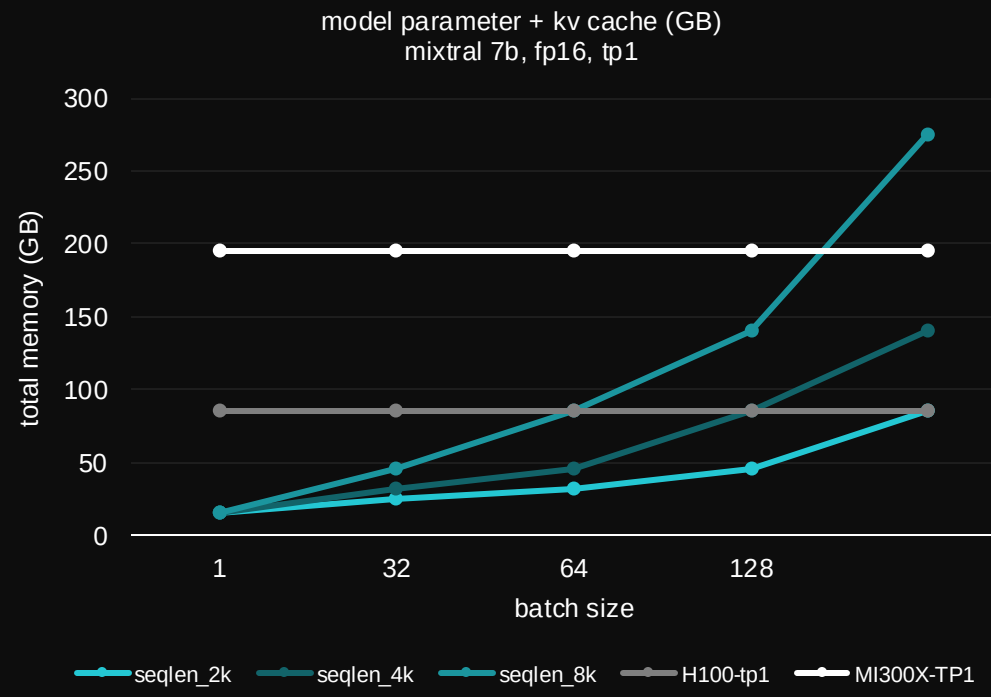
Algorithm Description

- Virtual memory of K/V cache into fixed size blocks to prevent memory waste for dynamic requests
- Parallel accessing (batch & beam search) to K/V cache blocks enables memory sharing
- Takeaway: **Higher output decoding throughput.** Near optimal GPU memory usage



Basic Usage

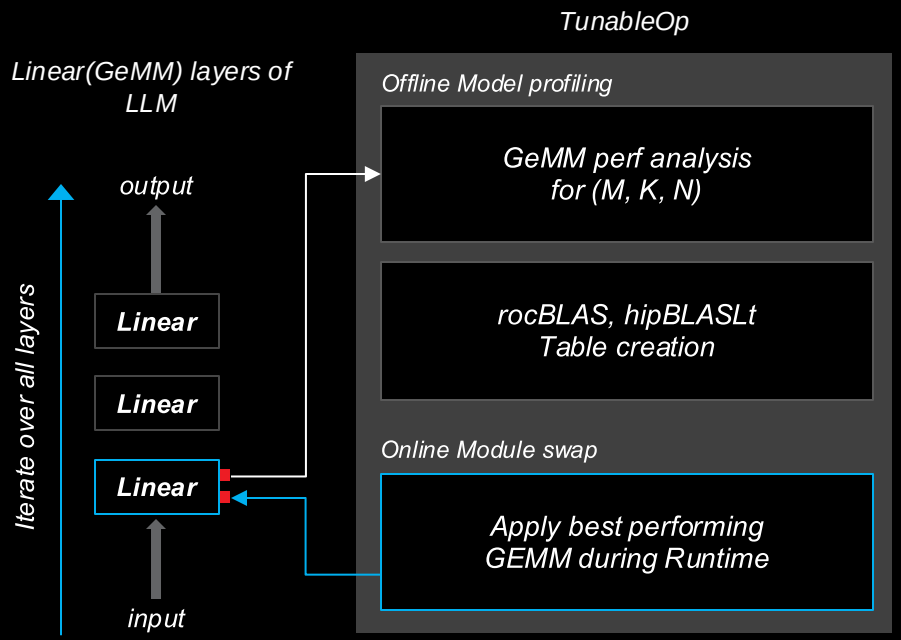
1 Leverage vLLM or Huggingface TGI LLM serving framework



Prefill / Decode Optimization – PyTorch TunableOp

Approach:

- Pick the best performing BLAS algorithms in the GEMM (M, K, N) search space
- Exclusively available in the AMD ROCm™ software
- Takeaway: **Additional GeMM performance gain**



Basic Usage

1 Available on the latest rocm-pytorch stable/nightly

Just turn on env var.

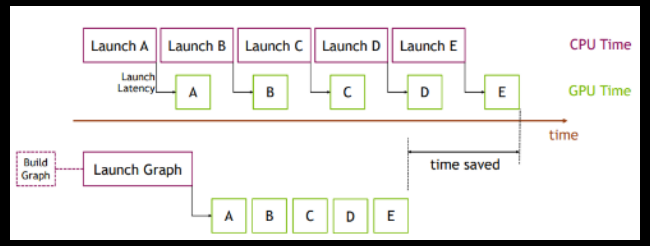
```
export PYTORCH_TUNALBEOP_ENABLED=1
```

https://rocm.blogs.amd.com/artificial-intelligence/pytorch-tunableop/README.html

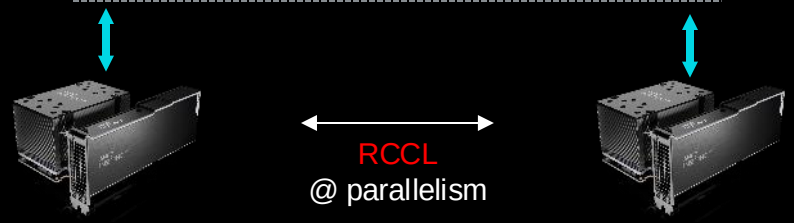
Decode Optimization – HIP Graph

Approach:

- Aggregate multiple kernels in a single graph and launch together
- Refer to this: https://pytorch.org/blog/accelerating-pytorch-with-cuda-graphs/
- Takeaway: **better output decoding performance** due to less asynchronous kernel launch from each GPU

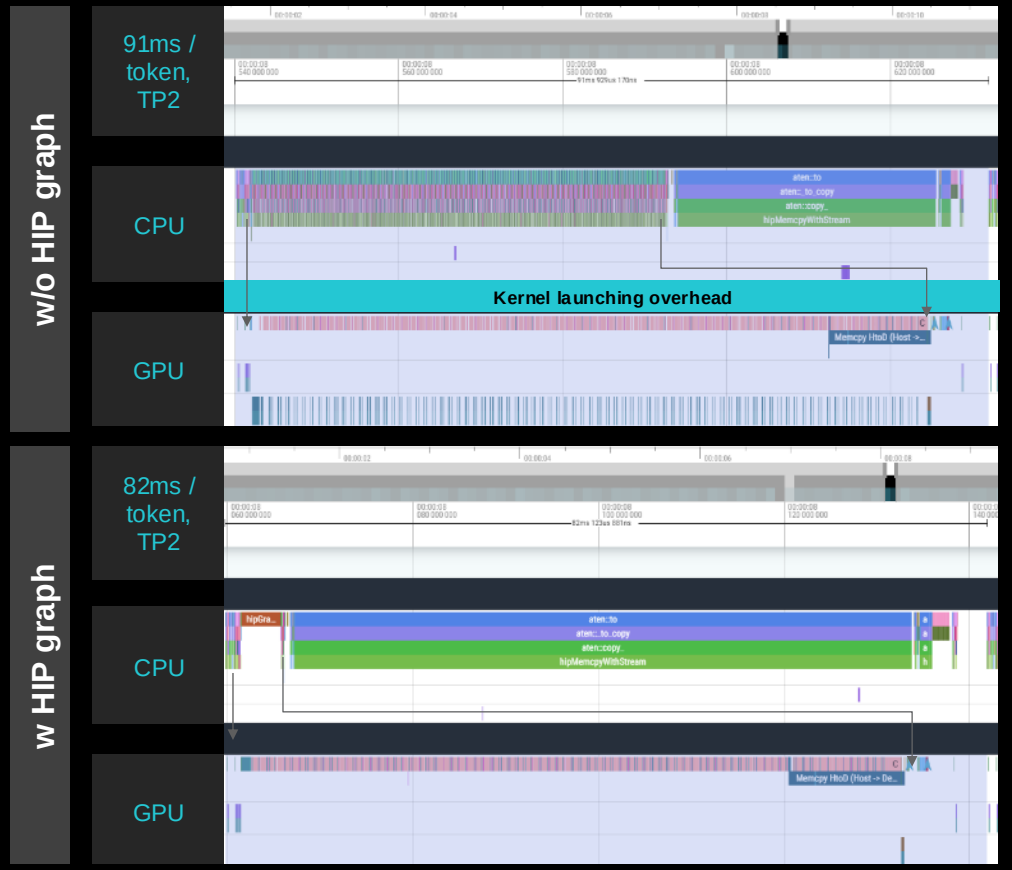


GPU Kernels launch



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

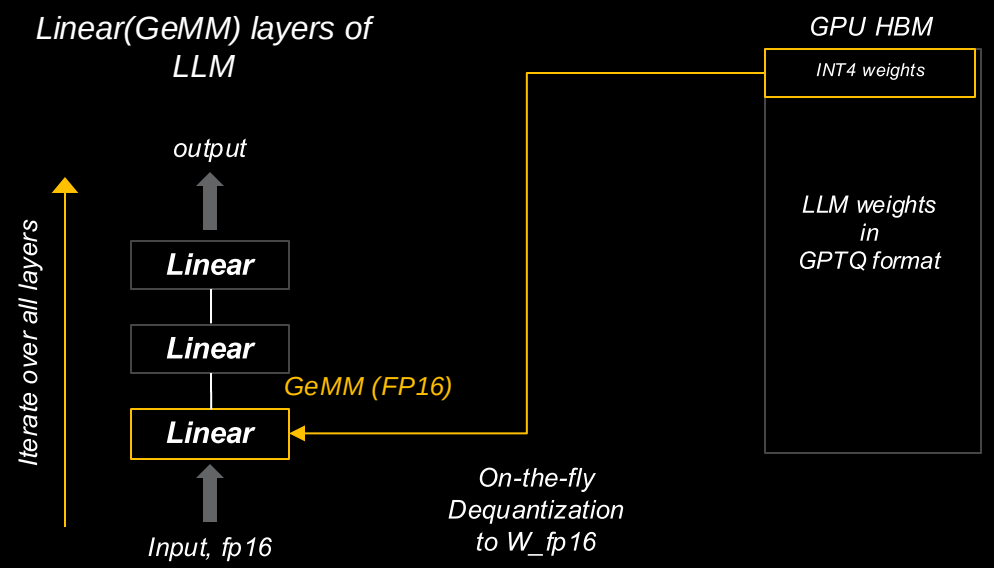
Behavior



Quantization - GPTQ / AWQ

Weights-only quantization to compress large LLMs to INT2, 3 or 4 bits (W4A16) in GPU HBM

Takeaway: **4x lower GPU memory consumption** (INT4), Minor perplexity degradation



Basic Usage

1. AutoGPTQ (install) — https://github.com/AutoGPTQ/AutoGPTQ

2. AutoAWQ & Kernel (install) — https://github.com/casper-hansen/AutoAWQ
https://github.com/casper-hansen/AutoAWQ_kernels
Dependency BNB: https://github.com/ROCm/bitsandbytes

3. Huggingface Transformers lib

```
from transformers import AutoModelForCausalLM, AutoTokenizer, GPTQConfig, AwqConfig
quant_config = GPTQConfig(bits=4, use_exllama=True, version=2)
quant_config = AwqConfig(version="exllama")
model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16, attn_implementation="eager", device_map=device, quantization_config=quant_config)
```

4. Leverage vLLM or Huggingface TGI LLM serving framework

Agenda

1. AMD Software and hardware for LLMs
2. LLM inference challenges and optimization directions
3. **Commercial LLM serving deployment with Hugging Face**
4. (Demo) Running Llama 3.1-405B
5. Useful AMD ROCm™ software resources & call for contributions
6. QA

Commercial LLM Serving Deployment with Hugging Face

Out-of-the-box optimized LLM inference on the MI300X accelerator

Text Generation Inference (TGI)

DOCs	https://huggingface.co/docs/text-generation-inference/en/index
Git	https://github.com/huggingface/text-generation-inference

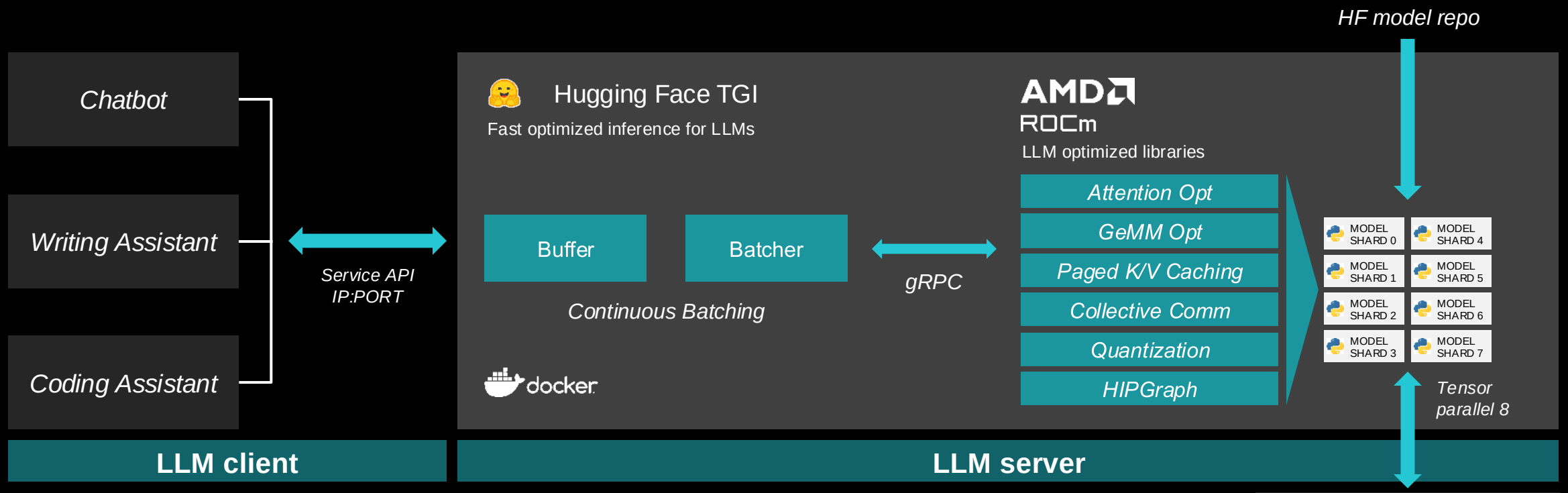
vLLM

DOCs	https://docs.vllm.ai/en/latest/index.html
Upstream Git	https://github.com/vllm-project/vllm
ROCm/vllm Git	https://github.com/ROCm/vllm

AMD Infinity Hub	https://www.amd.com/en/developer/resources/infinity-hub.html#f-amd_hub_category=AI%20%26%20ML%20Frameworks
ROCm/vllm docker	*docker pull rocm/vllm:rocm6.2_mi300_ubuntu22.04_py3.9_vllm_7c5fd50*
ROCm/vllm benchmark	https://github.com/ROCm/MAD/blob/develop/benchmark/vllm/README.md

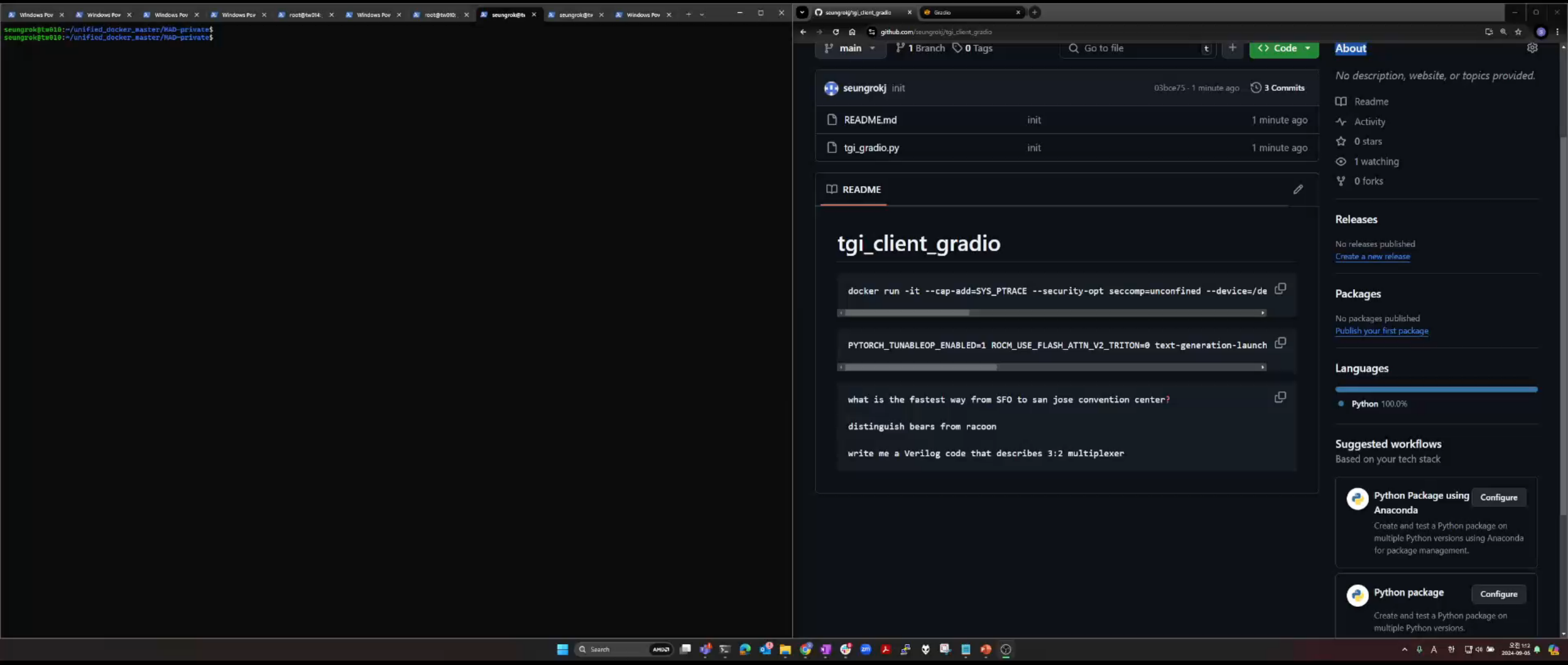
Hugging Face Text Generation Inference (TGI)

Overall architecture



Agenda

1. AMD Software and hardware for LLMs
2. LLM inference challenges and optimization directions
3. Commercial LLM serving deployment with Hugging Face
4. (Demo) Running Llama 3.1-405B
5. Useful AMD ROCm™ software resources & call for contributions
6. QA

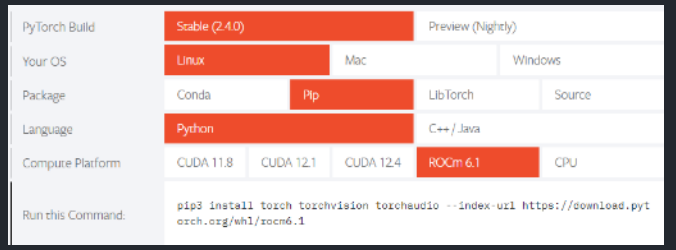

seungrok@tw010:~/unified_docker_master/MAD-private$
seungrok@tw010:~/unified_docker_master/MAD-private$
seungrokj/tgi_client_gradio
Gradio
github.com/seungrokj/tgi_client_gradio
main
1 Branch
0 Tags
Go to file
Code
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
seungrokj init
03bce75 · 1 minute ago
3 Commits
README.md
init
1 minute ago
tgi_gradio.py
init
1 minute ago
README
tgi_client_gradio
docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined --device=/de
PYTORCH_TUNABLEOP_ENABLED=1 ROCM_USE_FLASH_ATTN_V2_TRITON=0 text-generation-launch
what is the fastest way from SFO to san jose convention center?
distinguish bears from racoon
write me a Verilog code that describes 3:2 multiplexer
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Python 100.0%
Suggested workflows
Based on your tech stack
Python Package using Anaconda
Configure
Create and test a Python package on multiple Python versions using Anaconda for package management.
Python package
Configure
Create and test a Python package on multiple Python versions.

Agenda

1. AMD Software and hardware for LLMs
2. LLM inference challenges and optimization directions
3. Commercial LLM serving deployment with Hugging Face
4. (Demo) Running Llama 3.1-405B
5. **Useful AMD ROCm™ software resources & call for contributions**
6. QA

AMD ROCm™ Software Resources

AMD ROCm™ Software Documentation (6.2 is the latest)	• https://rocm.docs.amd.com/en/latest/
Ubuntu Dockers (ubuntu20.04, ubuntu22.04)	• https://hub.docker.com/r/rocm/dev-ubuntu-20.04/tags • https://hub.docker.com/r/rocm/dev-ubuntu-22.04/tags
Ubuntu pyTorch Dockers (ubuntu20.04, ubuntu22.04, CentOS 7 + torch2.1.2)	• https://hub.docker.com/r/rocm/pytorch/tags
pyTorch	• https://pytorch.org • 2.4.0 stable / 2.5.0 nightly



Join the Community and Contribute

ROCm™ Software Blog	• https://rocm.blogs.amd.com/blog.html
AMD Community AI Blog	• https://community.amd.com/t5/ai/bg-p/amd-ai
AMD and Hugging Face Collaboration	• https://huggingface.co/blog/huggingface-and-optimum-amd • https://huggingface.co/blog/huggingface-mi300-amd
AMD Specific LLM Acceleration Libraries	• [flash attention v2] https://github.com/ROCm/flash-attention • [open-ai triton] https://github.com/ROCm/triton • [vllm] https://github.com/ROCm/vllm • [xformers] https://github.com/ROCm/xformers • [hipBLASLt] https://github.com/ROCm/hipBLASLt • [rocBLAS] https://github.com/ROCm/rocBLAS • [composable kernel] https://github.com/ROCm/composable_kernel

Conclusion

AMD Instinct™ MI300 accelerators

Leading Memory and Competitive AI Performance to Power Generative AI

LLM Inference Challenges and Optimization Opportunities

Different techniques required to optimized performance

Running Llama 3.1-405B with Hugging Face text generation inference

Similar development flow compared to CUDA. Easy to develop LLMs.

Try Today

vLLM | AMD INSTINCT

Download vLLM toolkit

First publicly available, pre-optimized, QAed docker image for AMD Instinct MI300X accelerator.

Build your AI Models on the AMD Developer Cloud

Apply for access to AMD Instinct™ accelerators

Q & A


AMD

Disclaimer and Attribution