

AMD  ×  **Hugging Face**

Deploying Optimized LLMs on AMD GPUs

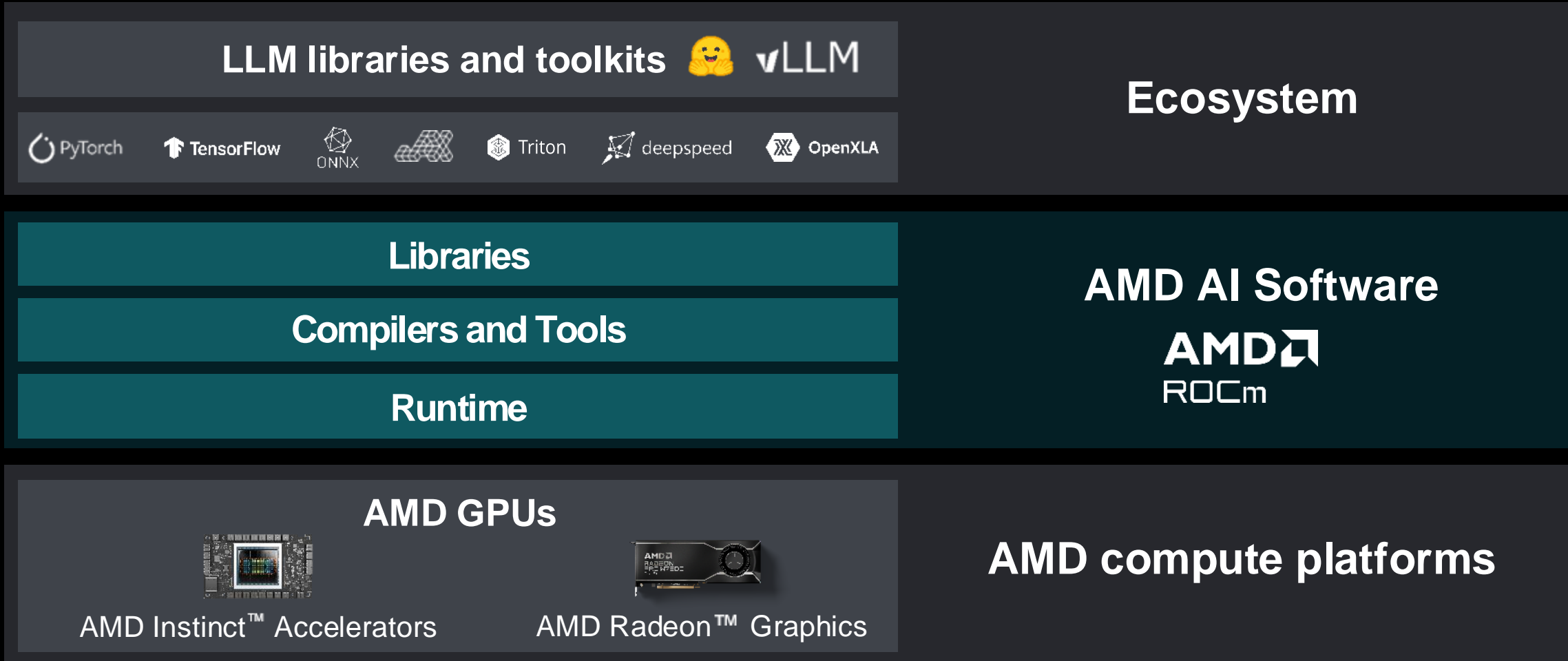
Seungrok Jung
AI Group at AMD

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



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

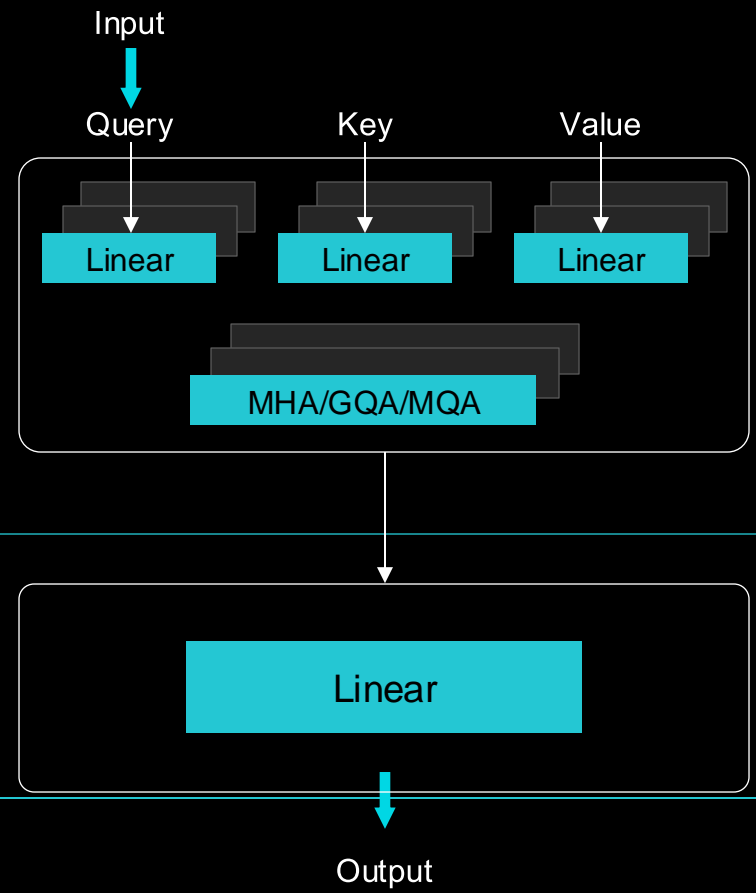
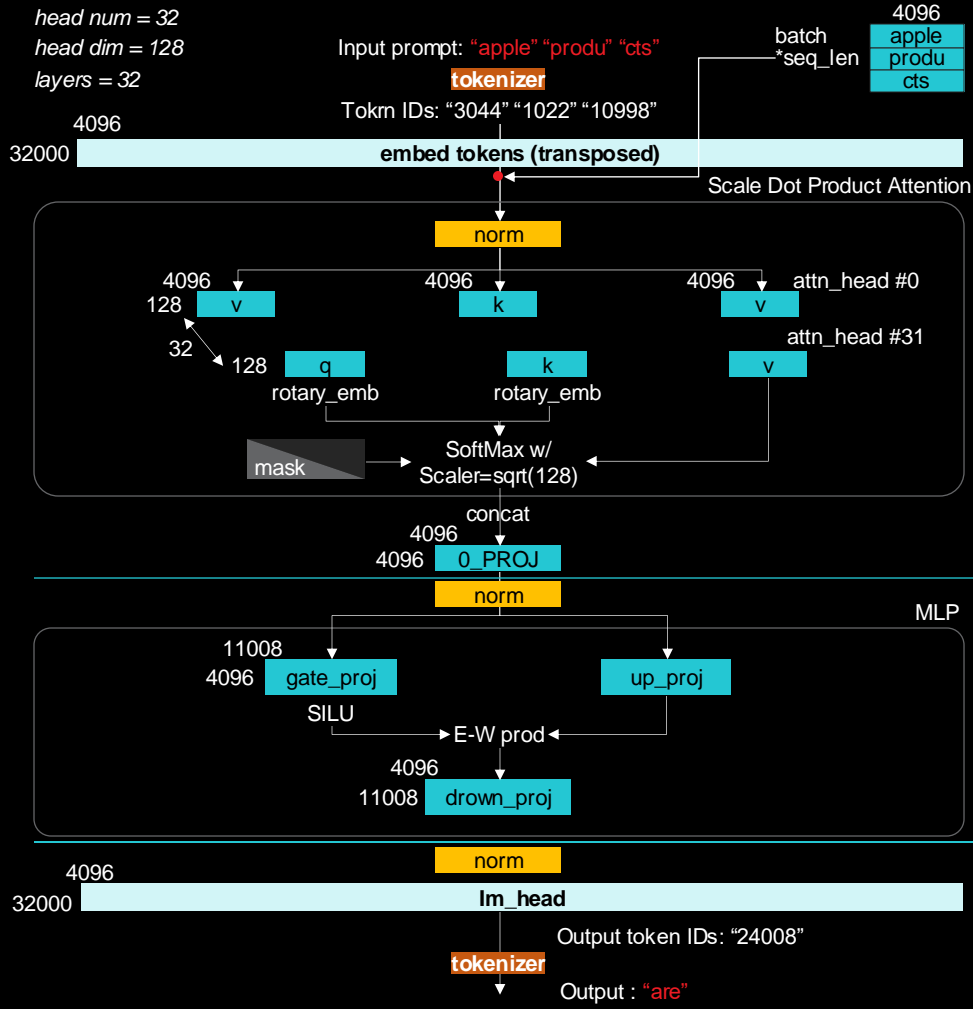
Agenda

-
- 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

Llama-2-7B

vocab size = 32000
 embedding size = 4096
 head num = 32
 head dim = 128
 layers = 32



SDPA (Attn)

Tensor Parallel
All_reduce_sum

MLP

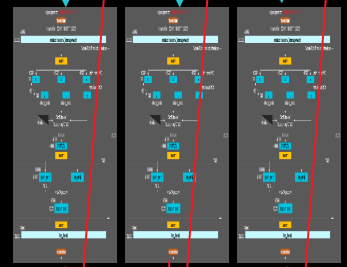
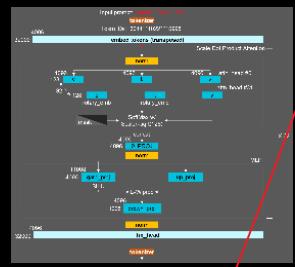
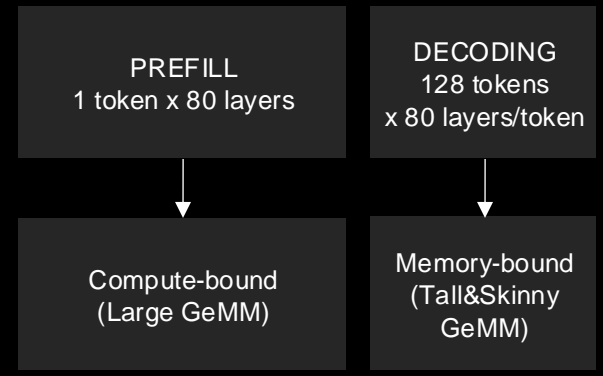
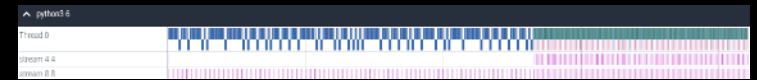
Tensor Parallel
All_reduce_sum

X 32

Prefill and Decoding

	Input prompt										Generated outputs															
											1 st Tkn	2 nd Tkn	3 rd Tkn	4 th Tkn												
batch 0	</s>	</s>	1	The	450	film	2706	follows	4477	a	263	US	3148	Navy	13166	commander	22079	<unk>	0	<unk>	0	<unk>	0	<unk>	0	
Batch 1	</s>	</s>	</s>	</s>	</s>	</s>	1	The	450	president	6673	of	310	Camb	9287	odia	26942	<unk>	29892	Hun	16899	Sen	5811	<unk>	29892	
Batch 2	</s>	</s>	</s>	</s>	</s>	</s>	2	The	450	largest	10150	continent	25523	in	297	the	278	world	3186	<unk>	29892	<unk>	29892	<unk>	29892	
Batch 3	</s>	She	2296	did	1258	not	451	respond	10049	to	304	her	902	mother	5673	because	1363	she	1183	was	471	too	2086	busy	19587	
Batch 4	</s>	</s>	</s>	1	The	450	key	1820	difference	4328	between	1546	human	5199	and	322	<unk>	0	<unk>	0	<unk>	0	<unk>	0	<unk>	0
Batch 5	</s>	</s>	</s>	</s>	1	The	450	bug	6494	bit	2586	es	267	are	526	<unk>	0	<unk>	0	<unk>	0	<unk>	0	<unk>	0	
Batch 6	</s>	1	The	450	movie	14064	par	610	as	294	ite	568	is	338	really	2289	<unk>	0	<unk>	0	<unk>	0	<unk>	0	<unk>	0
Batch 7	</s>	</s>	</s>	</s>	</s>	</s>	2	Why	3750	do	437	you	366	think	1348	the	278	US	3148	goverment	5874	<unk>	<unk>	<unk>	<unk>	

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

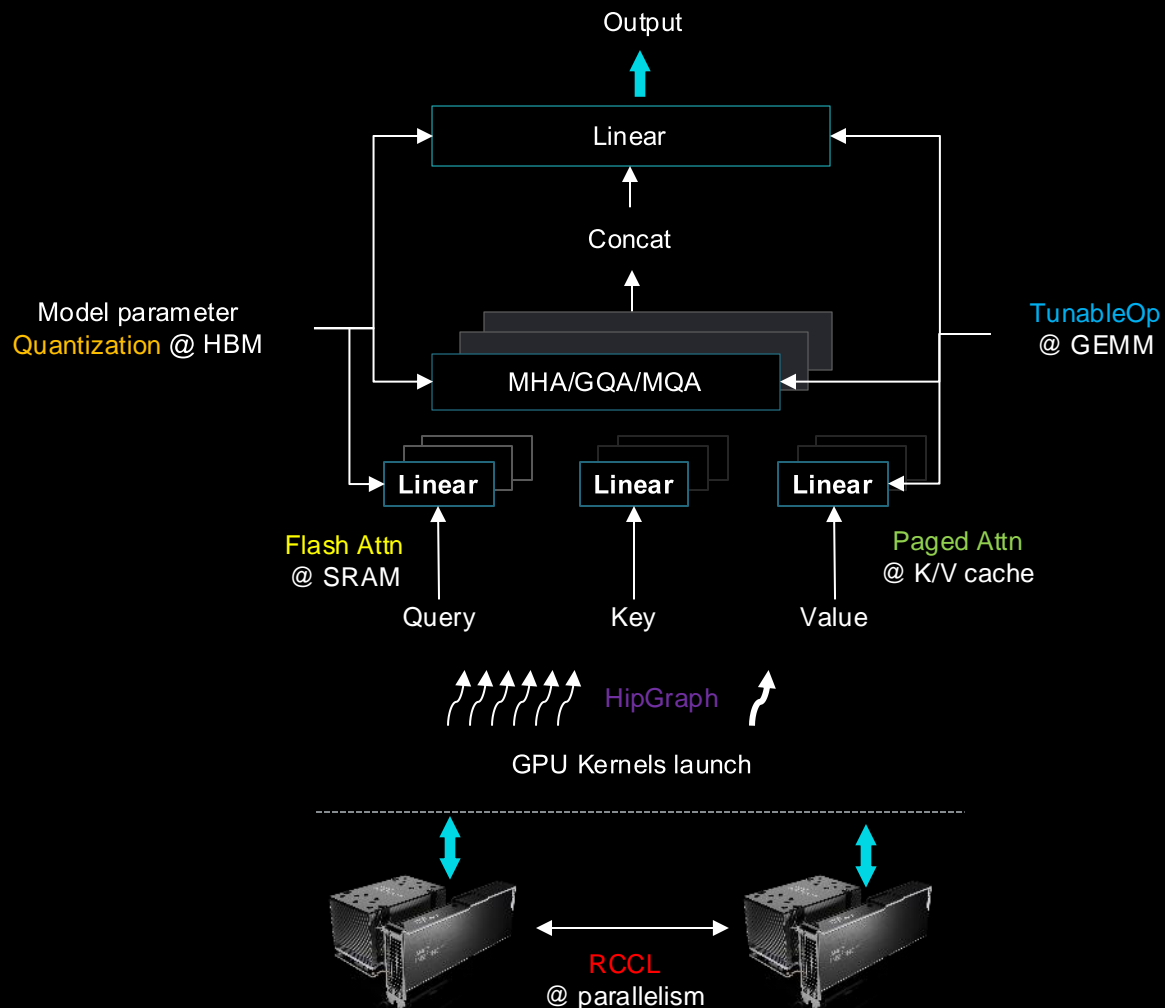


Prefill (1st Tkn): 200 ms

Decode (2nd ~ 4th Tkn): 33 ms/Tkn

Total latency = 200 ms + 33 ms x 3 = 299 ms

Typical LLM Inference Optimizations



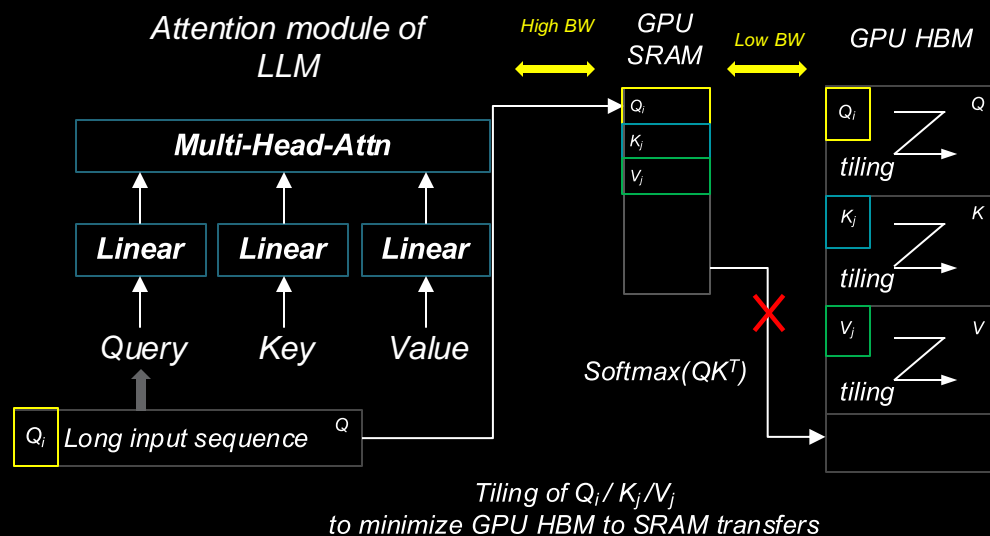
Flash Attention, Xformers	<ul style="list-style-type: none"> Tiling of input sequence in GPU SRAM to reduce HBM data movement
Paged Attention	<ul style="list-style-type: none"> Partitioned KV cache into fixed size blocks to reduce memory usage
GEMM Optimization – PyTorch TunableOp	<ul style="list-style-type: none"> Automatic selection of the best performing GEMM kernels
Graph Optimization – HipGraph	<ul style="list-style-type: none"> Launch multiple kernels through a single CPU operation
Collective communication – RCCL	<ul style="list-style-type: none"> Collective Ops across multiple devices to support Tensor/ Pipeline parallel
Quantization – GPTQ, Bitsandbytes	<ul style="list-style-type: none"> 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

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



Basic Usage

- 1 Install AMD ROCm™ flash attention

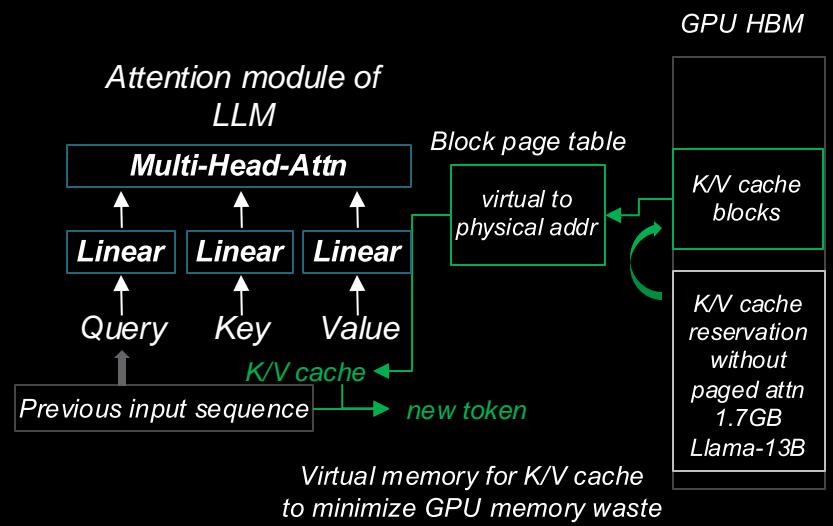

```
git clone https://github.com/ROCm/flash-attention
```
- 2 Utilize FAv2 with Hugging Face transformers Lib


```
import torch
from transformers import
AutoModelForCausalLM, AutoTokenizer
model =
AutoModelForCausalLM.from_pretrained(mod
el_name, torch_dtype=torch.float16,
attn_implementation="flash_attention_2")
```
- 3 Leverage vLLM or Hugging Face TGI LLM serving framework

Decode Optimization – Paged Attention

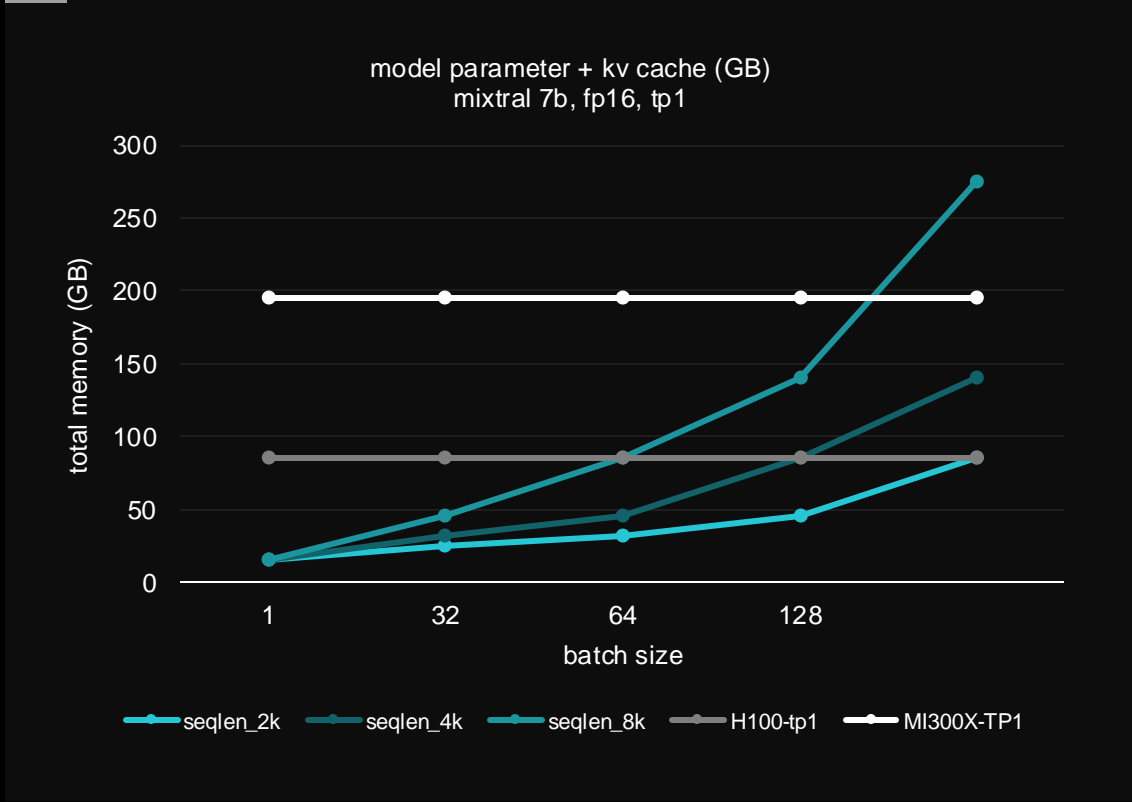
Algorithm Description

- 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

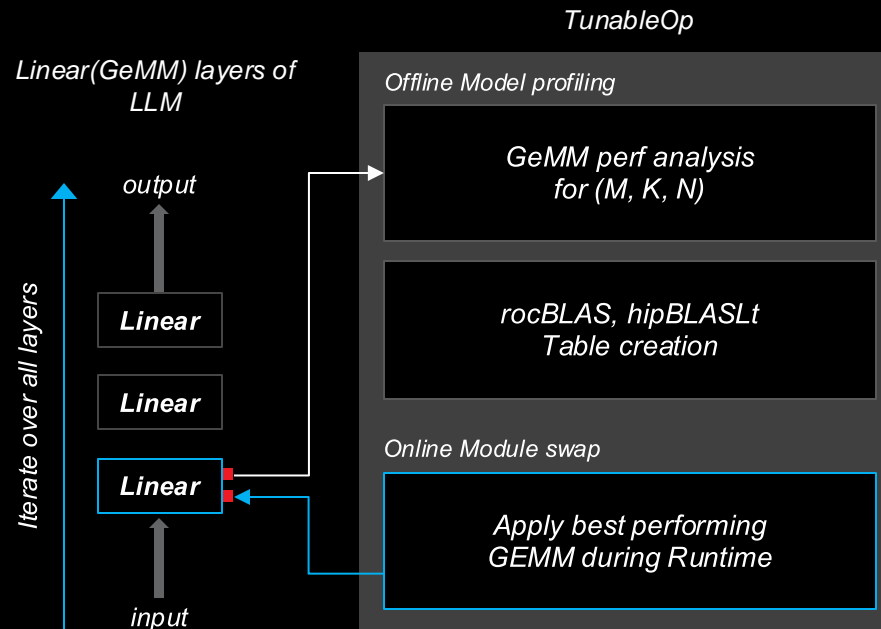
1

Available on the latest rocm-pytorch stable/nightly

Just turn on env var.

```
export PYTORCH_TUNABLEOP_ENABLED=1
```

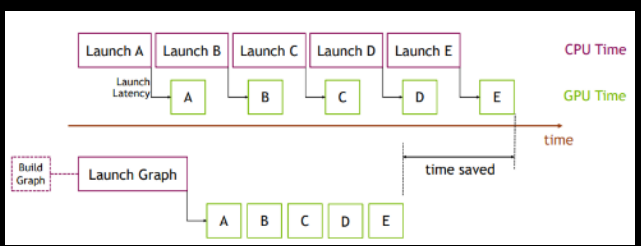
<https://rocm.blogs.amd.com/artificial-intelligence/pytorch-tunableop/README.html>



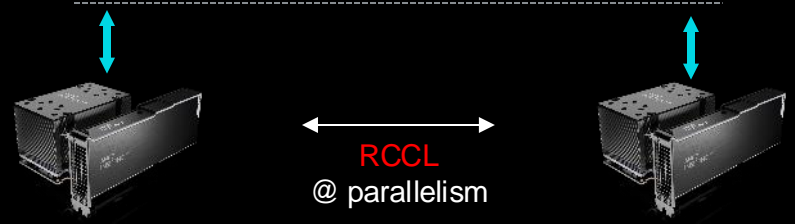
Decode Optimization – HIP Graph

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

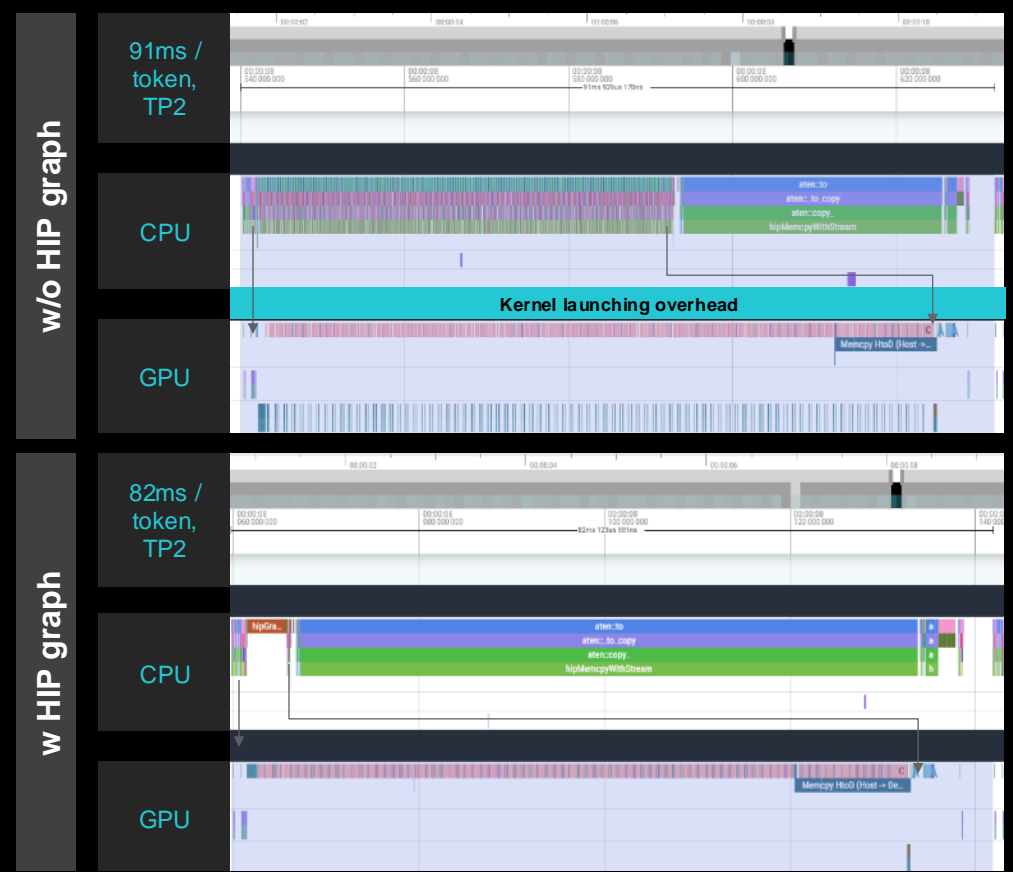


↑↑↑↑↑ HipGraph ↑
GPU Kernels launch



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

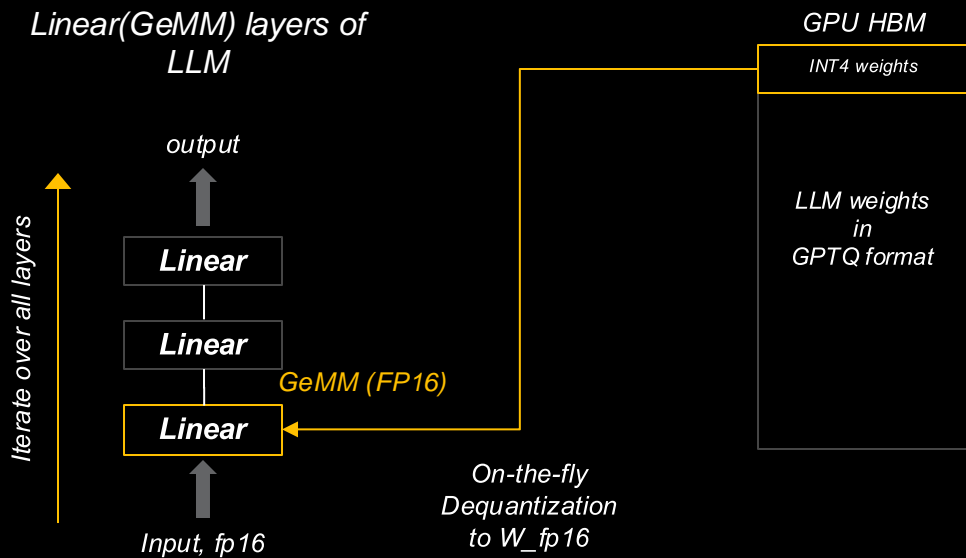
Behavior



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>

DOCs <https://docs.vllm.ai/en/latest/index.html>

AMD Infinity Hub https://www.amd.com/en/developer/resources/infinity-hub.html#-amd_hub_category=AI%20%26%20ML%20Frameworks

Git <https://github.com/huggingface/text-generation-inference>

Upstream Git <https://github.com/vllm-project/vllm>

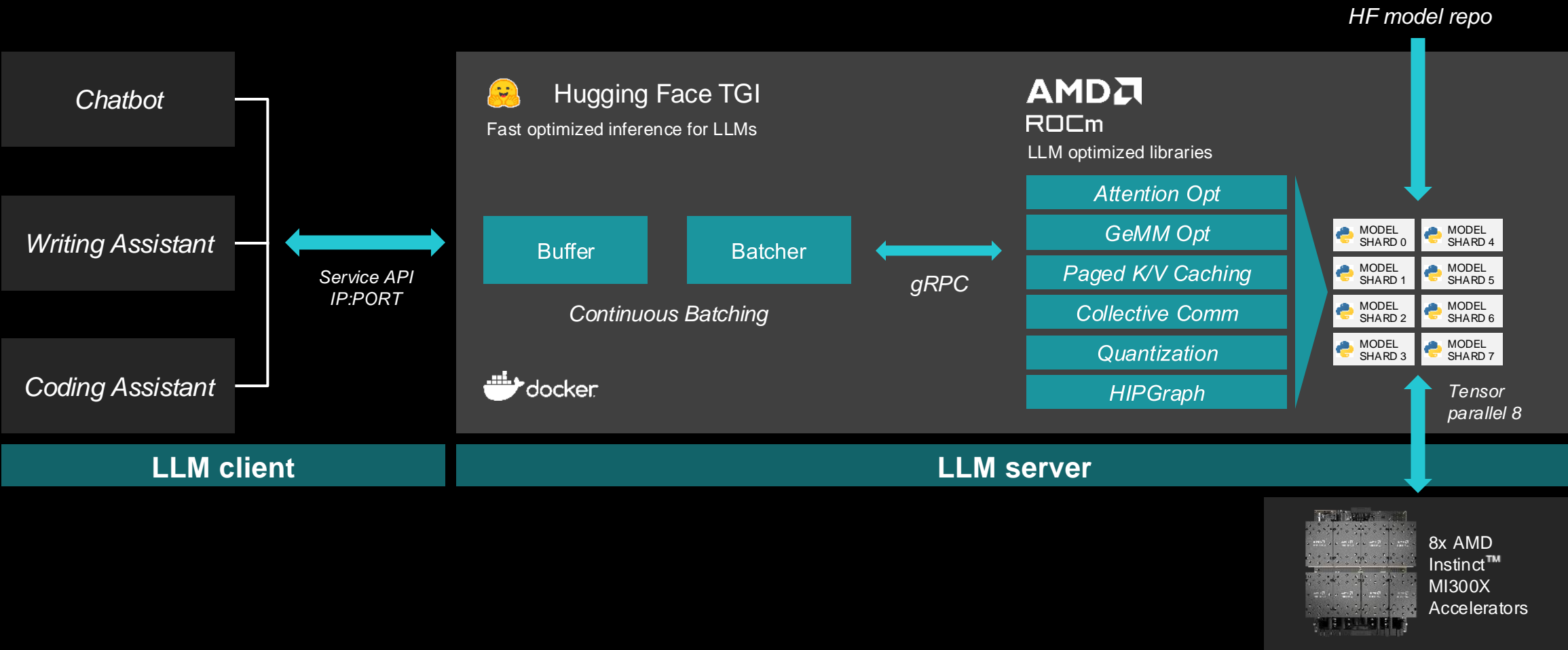
ROCm/vllm docker `docker pull rocm/vllm:rocm6.2_mi300_ubuntu22.04_py3.9_vllm_7c5fd50`

ROCm/vllm Git <https://github.com/ROCm/vllm>

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

```
Windows Pov x Windows Pov x Windows Pov x Windows Pov x root@tw014: x Windows Pov x root@tw010: x seungrok@ts x seungrok@ts x Windows Pov x + -
```

```
seungrok@tw010:~/unified_docker_master/HAD-private$  
seungrok@tw010:~/unified_docker_master/HAD-private$
```

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

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.

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=/dev
```

```
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
```

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)

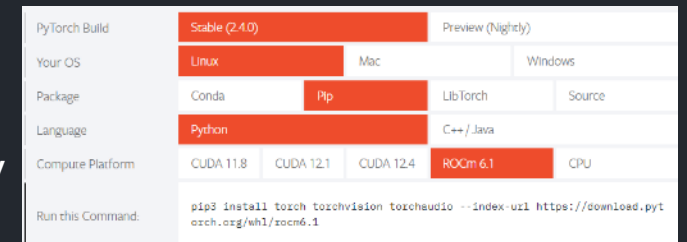
- <https://hub.docker.com/r/rocm/dev-ubuntu-20.04/tags>
- <https://hub.docker.com/r/rocm/dev-ubuntu-22.04/tags>

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



Download vLLM toolkit



First publicly available, pre-optimized, QAd 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

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18u.

© 2024 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, AMD 3D V-Cache, AMD CDNA, AMD Instinct, AMD RDNA, AMD ROCm, AMD XDNA, EPYC, Radeon, Ryzen, Versal, Threadripper, and combinations thereof are trademarks of Advanced Micro Devices, Inc. HP® and the HP logo are registered trademarks of Hewlett-Packard Development Company, L.P. Lenovo® is a trademark of Lenovo in the United States, other countries, or both. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. Microsoft and Windows are registered trademarks of Microsoft Corporation in the US and/or other countries. Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners. Certain AMD technologies may require third-party enablement or activation. Supported features may vary by operating system. Please confirm with the system manufacturer for specific features. No technology or product can be completely secure.