AMD & SGLang

Highlights – Speed in Action

08.2025

AMD

together we advance_

A Snapshot

AMD & SGLang – one year journey of greatness

- AMD – a top org. contributor to the SGLang project
- SGLang – top LLM inference stack recommended by AMD
- SGLang – fostered AITER (AI Tensor Engine for ROCm) in AMD
- AMD & SGLang – **1st** MI300X production in deployment, live in Q1/25

AMD & SGLang == Upstream `sgl-project`

- AMD – no ROCm folk to maintain
- AMD – provides MI300X/MI325/MI350 to upstream CIs
- AMD & SGLang – fully integrated GitHub workflow & docker hub repo

How it began [Triton → AITER]

Initial triage – Aug 2024

- Driven by performance demand
- Motivated by advanced use cases – KV cache reuse & share, Structured Decoding, etc.

First AMD code merged – Sep 2024

- AMD MI30X enabled on ROCm 6.2
- All features of the time were addressed: FP8 (OCP interoperability), Triton MoE, CUDA Graph, Torch Compile
- Comparable performance vs. competitive solution: open source vLLM or SGLang on Nvidia platform

SGLang @ AMD AI Day, San Francisco – Oct 10, 2024

- Executive keynote featured by AMD
- Luminary Developer Speech by Dr. Lianmin Zheng

INT4-FP8 MoE introduced for very large models – Dec 2024

- PoC was done to support MoE model of 2 trillion parameters on single MI300X node ($8 \times \text{MI300X}$)
- Initial kernels and models served since Jan, 25

DeepSeek V3/R1/etc.

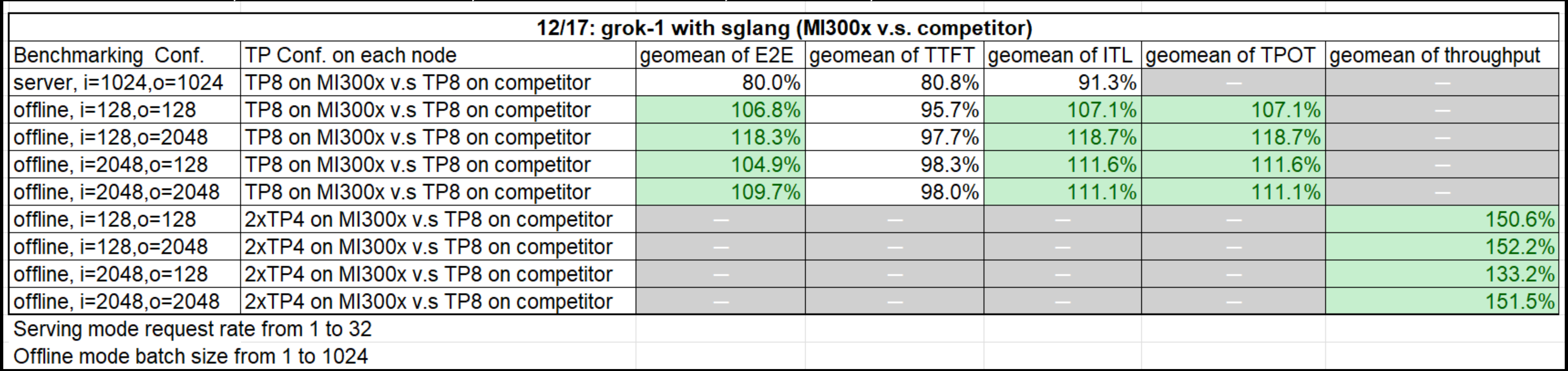
- Day 0 support, fully Functional with Accuracies – Dec 2024
- Performance boosted fast ever since

AITER (AI Tensor Engine for ROCm)

- Launched for SGL performance demand – Jan 2024
- Fused MoE kernels first used in SGLang – Feb 2024
- Fused MLA kernels, Attention kernels, etc. were added over time

Q4/24 Highlights — [Triton, AMD & SGL & xAI]

- Competitive performance vs. competing solution: Open Source SGLang or vLLM on competing platforms
- Achievements on 12/17/2024: enablement GROK1 with SGLang on MI300x machines
  - **Server Mode, IL1024-OL1024:** E2E latency is around 80% of competitor number.
  - **Offline Mode, 1xTP8 v.s. 1xTP8 :** E2E latency is around 105~118% of competitor.
  - **Offline Mode, 2xTP4 v.s. 1xTP8 :** Throughput is around 133~152% of competitor.
- Deployment
  - Taking 30% of production traffics on 800*MI300x GPUs. The p50 is 10% faster than competitor.
- Optimizations
  - MoE triton, CustomAllReduce, elementwise kernel, RMSNorm, …

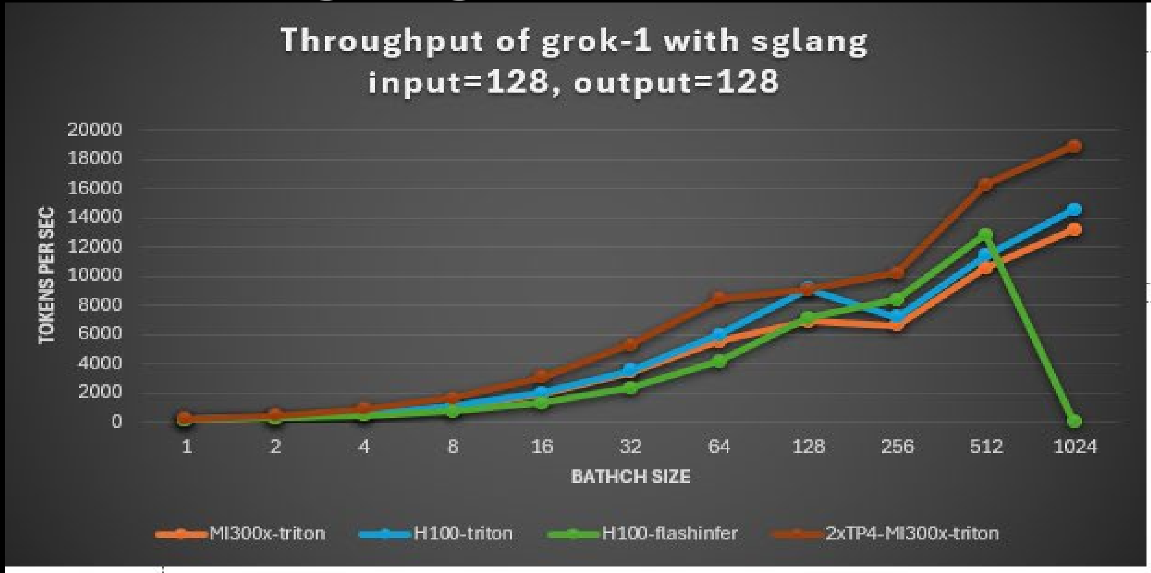


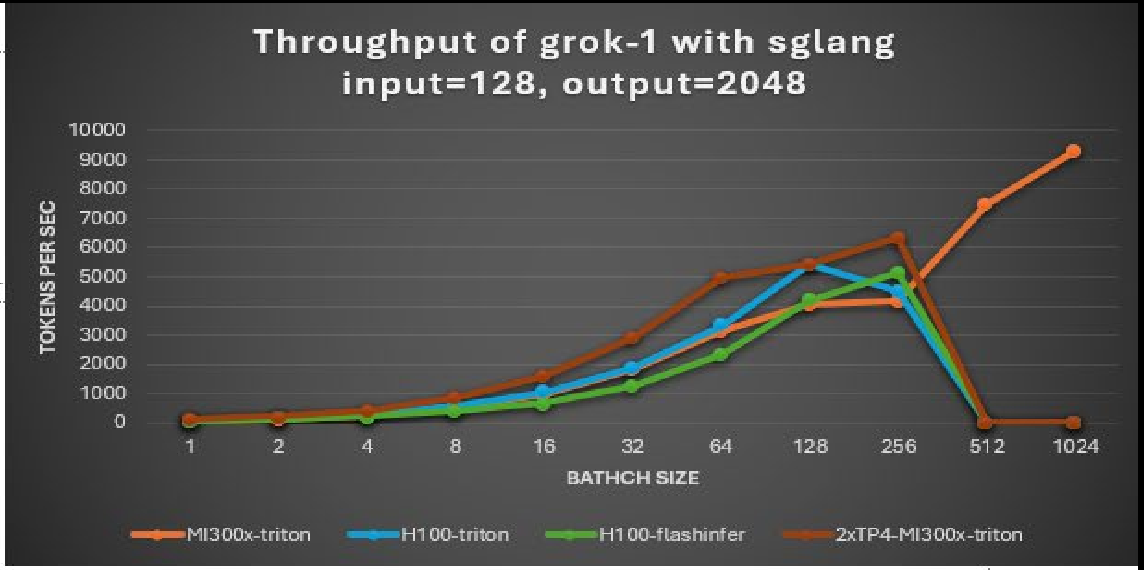
12/17: grok-1 with sglang (MI300x v.s. competitor)

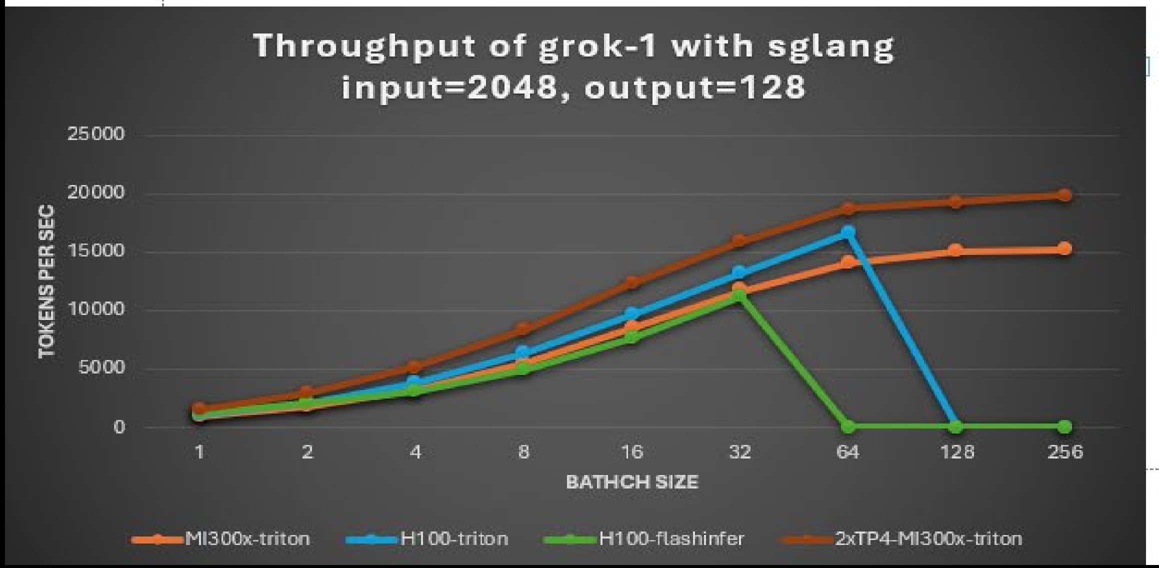
Benchmarking Conf.	TP Conf. on each node	geomean of E2E	geomean of TTFT	geomean of ITL	geomean of TPOT	geomean of throughput
server, i=1024,o=1024	TP8 on MI300x v.s TP8 on competitor	80.0%	80.8%	91.3%	—	—
offline, i=128,o=128	TP8 on MI300x v.s TP8 on competitor	106.8%	95.7%	107.1%	107.1%	—
offline, i=128,o=2048	TP8 on MI300x v.s TP8 on competitor	118.3%	97.7%	118.7%	118.7%	—
offline, i=2048,o=128	TP8 on MI300x v.s TP8 on competitor	104.9%	98.3%	111.6%	111.6%	—
offline, i=2048,o=2048	TP8 on MI300x v.s TP8 on competitor	109.7%	98.0%	111.1%	111.1%	—
offline, i=128,o=128	2xTP4 on MI300x v.s TP8 on competitor	—	—	—	—	150.6%
offline, i=128,o=2048	2xTP4 on MI300x v.s TP8 on competitor	—	—	—	—	152.2%
offline, i=2048,o=128	2xTP4 on MI300x v.s TP8 on competitor	—	—	—	—	133.2%
offline, i=2048,o=2048	2xTP4 on MI300x v.s TP8 on competitor	—	—	—	—	151.5%

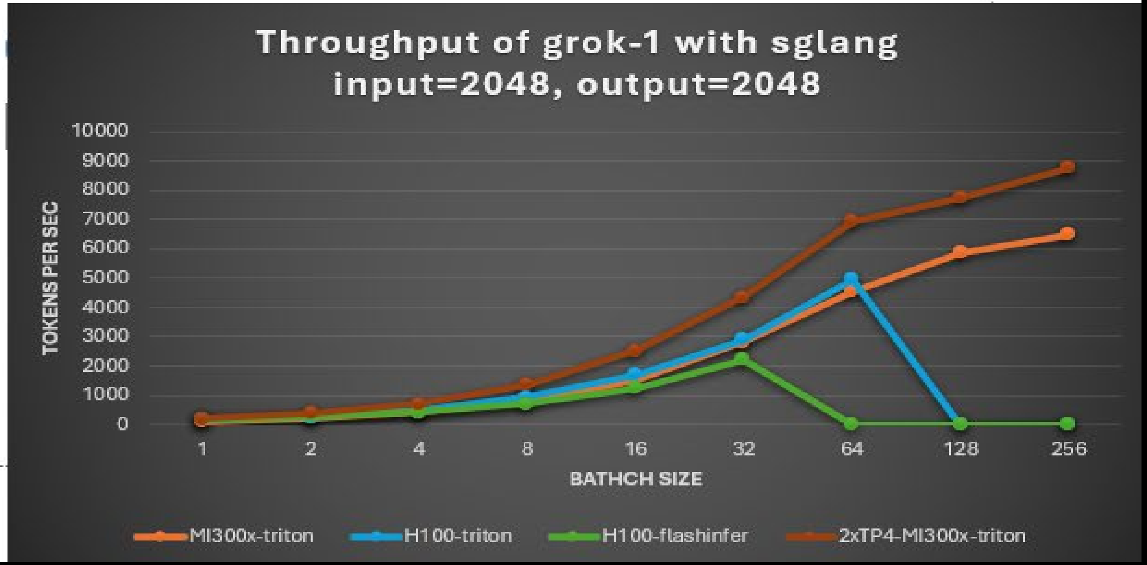
Serving mode request rate from 1 to 32
Offline mode batch size from 1 to 1024

Q4/24 Highlights — Continue

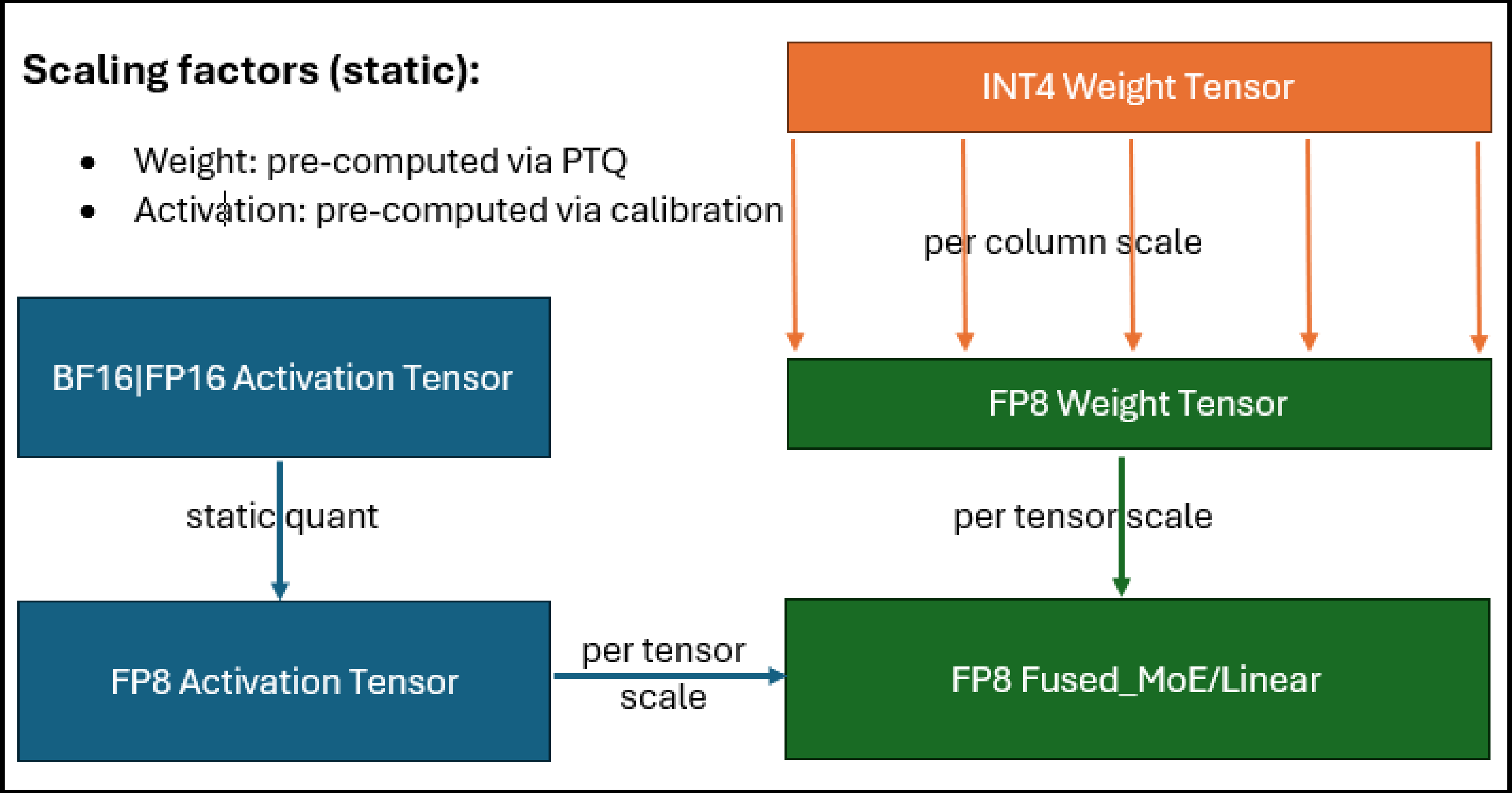








Q1/25 Highlights — [Grok-x; INT4 weight, FP8 Compute; 1st Production Deployment]



- **Serves model sizeof $8 \times Grok1$ on single MI300X (previous page)**
- **Work with Quark (AMD quantizer) quantized int4/fp8 models with dual scaling factors** (orange & green)
- **Also work with weight quantization at weight loading (PR in review)**
- **$INT_4 \rightarrow FP_8$ upcast is perf critical, kernel (fusion) implementation**
- **Integrate the AITER CK/ASM fused kernels for performance**

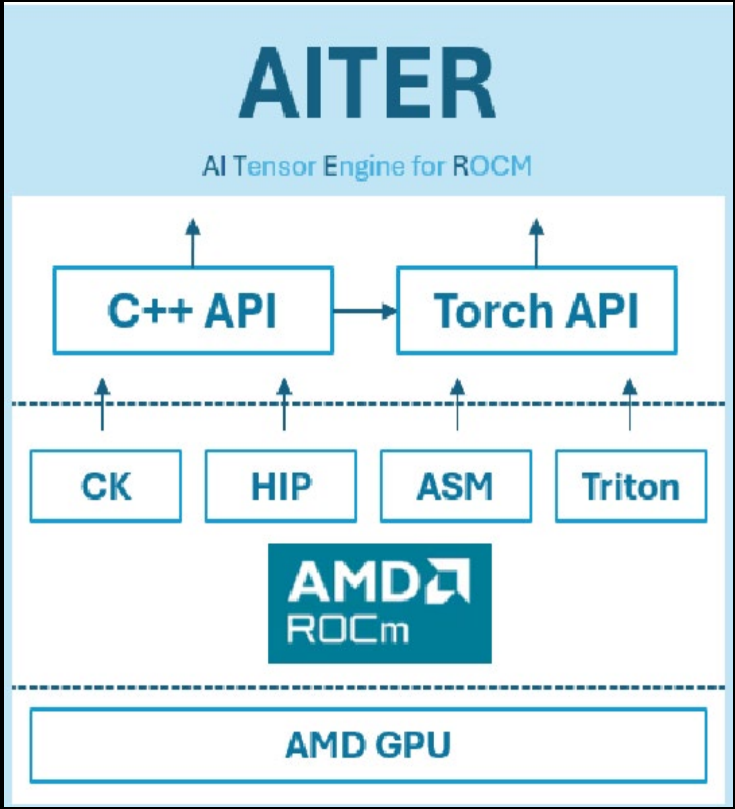
Q2/25 Highlights — [AITER Attention Backend; Llama-4]

AITER scope in the context of inference:

- A lightweight and customizable/tunable/deployable AI tensor engine.
- LLM block-level solutions to plug into customer's framework. (fused-moe, quant-KVCache, etc…)
- Heuristic/user-driver-tuning for self-hosted kernels written in CK/ASM/HIP/Triton
- Fused dynamic quantization kernels and offline utilities.
- A collection of inference AI operators
- C++/Pytorch API and package

AITER Attention Backend introduced optimized attentions from AMD

- Group AITER optimizations for attentions
- Supports Prefill, Extend, Decode attentions
- Supports MHA/GQA/MQA/MLA algo. of the most common sizes (extensible)
- Includes (both `MI30X` and `MI35X`):
  - `flash_attn_varlen_func`
  - `paged_attention_ragged`
  - `mla_prefill_fwd`
  - `mla_decode_fwd`
  - `mha_batch_prefill_func [highly optimized read paged KV]`
- Speculative decoding, Draft and MTP support `[tree sampling WIP]`
- Newer updates to `batch_prefill is WIP`



Q2/25 Highlights — [MI35X launch, MXFP4]

Initial MXFP4 enablement on MI350/MI355 hardware

- Apply new hardware instructions for MFMA operation
- Provide triton MoE and linear MXFP4 solution
- Work with Quark quantized DSv3
- Make Grok1 BF16 model to be MXFP4 quantized, dynamically at loading

Optimize MXFP4 model performance on MI350/MI355 hardware

- Provide the CK-MoE solution to support dynamic-tiling to enhance Gemm computation intensity
- Fused the activation MXFP4 quant with the different operations (ex: activation, linear, Layer Norm)
- Add more linear layers with MXFP4 quantization without affecting correctness

Q3/25 Highlights — [New formats; More models: Kimi-K2, GPT-OSS]

`OCP-FP8` native supports built for `MI35X (GFX950)`

- Seamless supports of FP8 from MI30X to MI35X, with OCP-FP8 Interoperability (effortless)

`NVFP4` QDQ is supported via `Petit Kernels`

- Supports **`CDNA2/CDNA3`** GPUs (AMD **`MI2XX/MI3XX`** series)
- https://github.com/sgl-project/sglang/pull/7302
  Supports Dense Models
- WIP for MoE Models

`MXFP4` is supported on `MI35X` via Triton & AITER CK Kernels

- Use MXFP _MFMA_ instructions from **`MI35X (GFX950)`** native ISA support
- Supported from both Triton (Dense, MoE) and AITER/CK (Dense, MoE) for greater flexibility
- Support Quark MXFP4 PTQ models, and dynamic quantized (MXFP4 weight, activation) MoE

`Kimi-K2`

- Day 0 support, functions with accuracy
- Early measure shows great advantage – MI300X vs. H200 for $\mathbf{max\_concurrency} > 4$
- AITER MLA optimization WIP (to support **`number of heads 8 per TP`**)

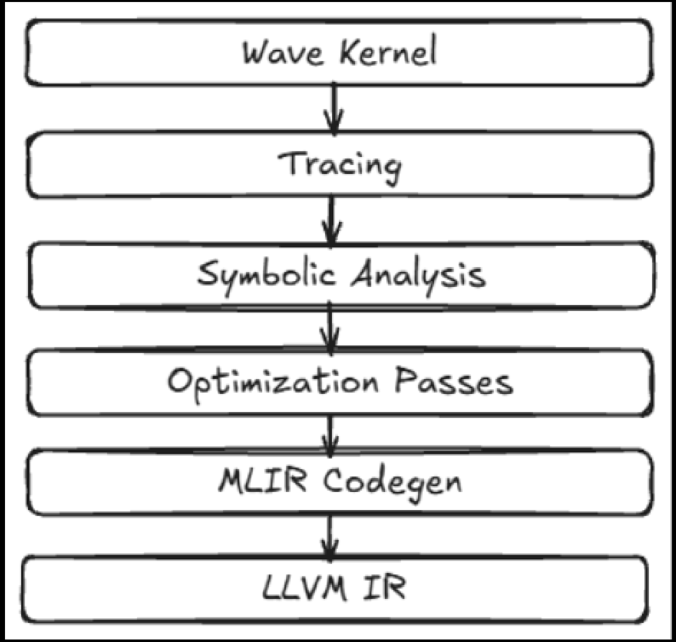
`GPT-OSS`

- Day 0 support, fully functional with accuracies
- BF16 model runs on MI30X/MI35X
- OpenAI/MXFP4 model runs on MI35X
- AITER CK-MoE optimization WIP (to support **`bias`**)

Q3/25 Highlights — [Wave backends]

Wave Attention Backend introduced from AMD

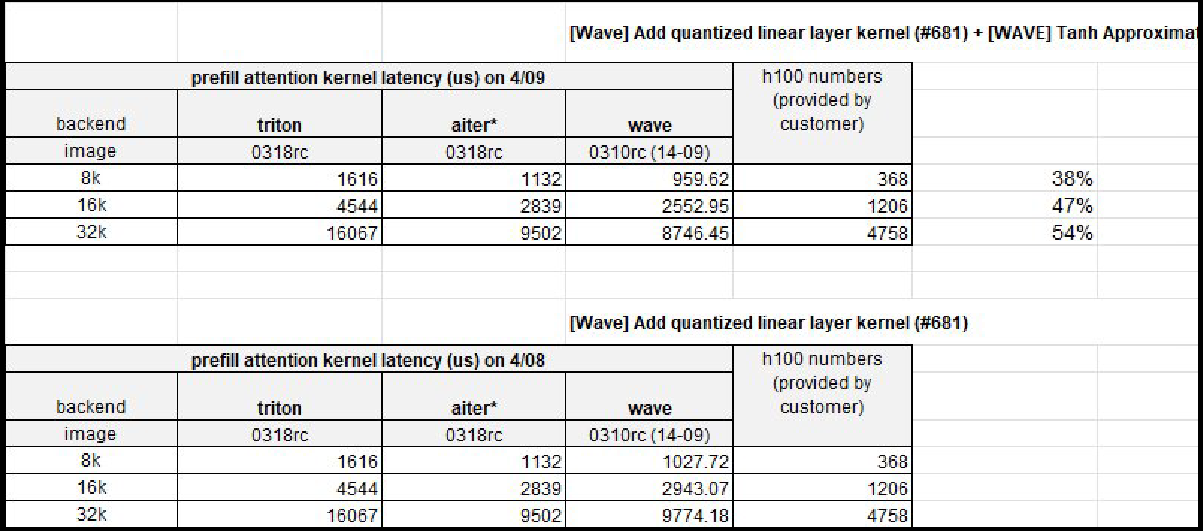
- `Wave` is a high-performance pythonic DSL
- `SWMD - Wave` level programming
- First-class support for PyTorch tensors
- SGLang attentions: Prefill, Extend, Paged Decode
- Constraint based Distribution + Tiling
- Harness `MLIR` based optimizations



Wave level control – Can specify how work is distributed to each wave within a block with minimal increase in complexity.

Easy Tiling/Reordering Optimization – Constraint/tiling is separate from program/compute. It allows for experimenting with different tiling adjustment, loop reordering or workgroup reordering without touching the actual kernel/program.

Replace tanh with tanh_approximation on wave kernel backend, this brings **7~15%** improvement. (IL=8k, from 1027.72 -> 959.62)



[Wave] Add quantized linear layer kernel (#681) + [WAVE] Tanh Approximat

prefill attention kernel latency (us) on 4/09				h100 numbers (provided by customer)	
backend	triton	aiter*	wave		
image	0318rc	0318rc	0310rc (14-09)		
8k	1616	1132	959.62	368	38%
16k	4544	2839	2552.95	1206	47%
32k	16067	9502	8746.45	4758	54%

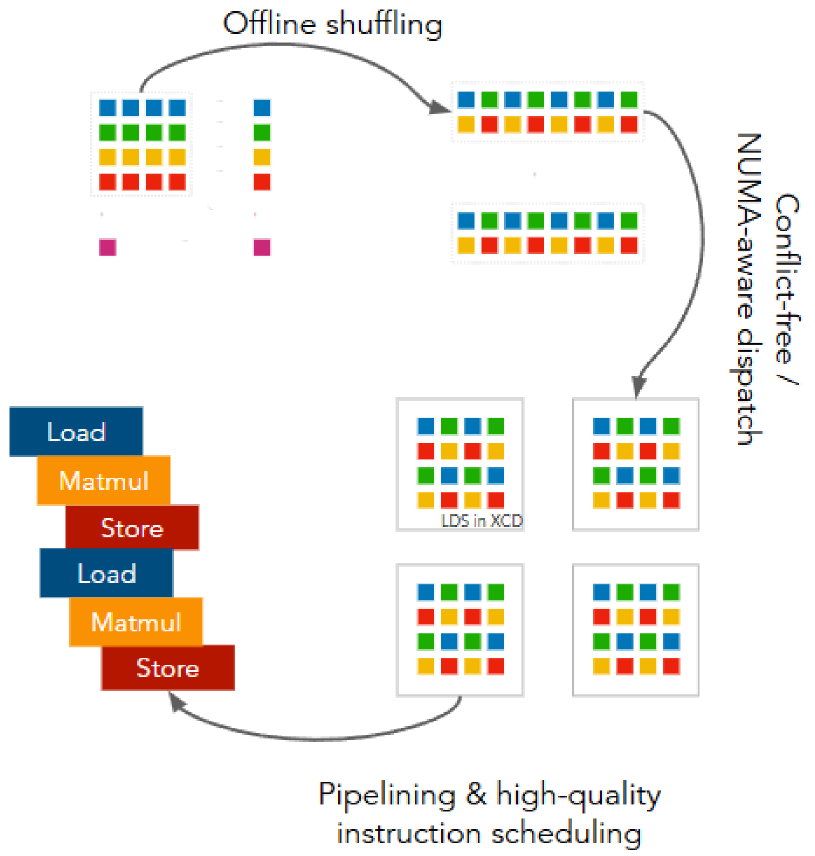
[Wave] Add quantized linear layer kernel (#681)

prefill attention kernel latency (us) on 4/08				h100 numbers (provided by customer)
backend	triton	aiter*	wave	
image	0318rc	0318rc	0310rc (14-09)	
8k	1616	1132	1027.72	368
16k	4544	2839	2943.07	1206
32k	16067	9502	9774.18	4758

AMD
together we advance_

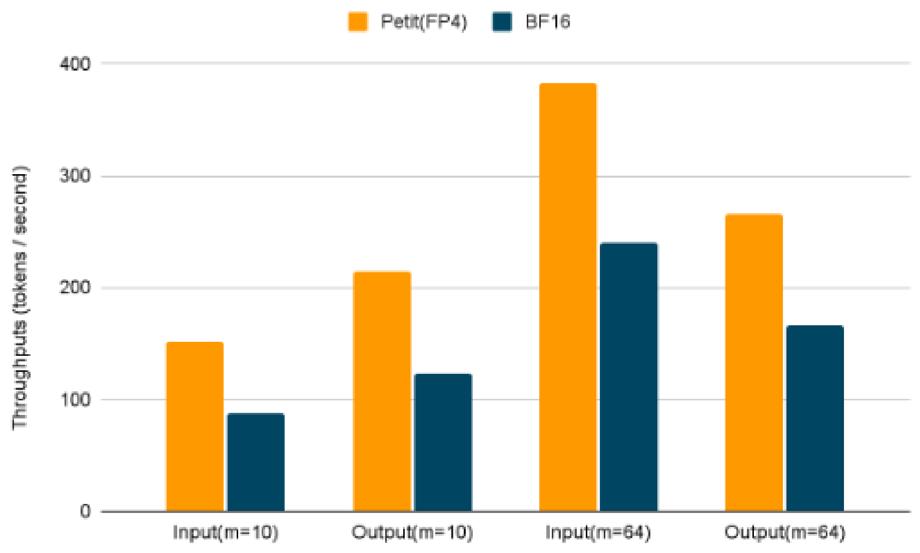
Q3/25 Highlights — [Petit Kernel]

- Petit: NVFP4 for MI250/MI300
- Tailored, E2E optimizations
  - Efficient dequantizations via offline bit-shuffling
  - Premuted LDS layout, XCC-aware workload partition
  - Software pipelining & low-level instruction scheduling


Offline shuffling
Conflict-free / NUMA-aware dispatch
LDS in XCD
Load
Matmul
Store
Load
Matmul
Store
Pipelining & high-quality instruction scheduling

Q3/25 Highlights — [Petit Kernel]

- Community-built from causalflow.ai (3-BSD license)
- Available on SGLang 0.4.10
  - 1.6x faster than BF16 models (Llama 3.3 70B)
  - 3.7x faster than HipBlasLt during decoding
- Roadmap
  - INT4/MXFP4 support
  - MoE support for DeepSeek & GPT-OSS

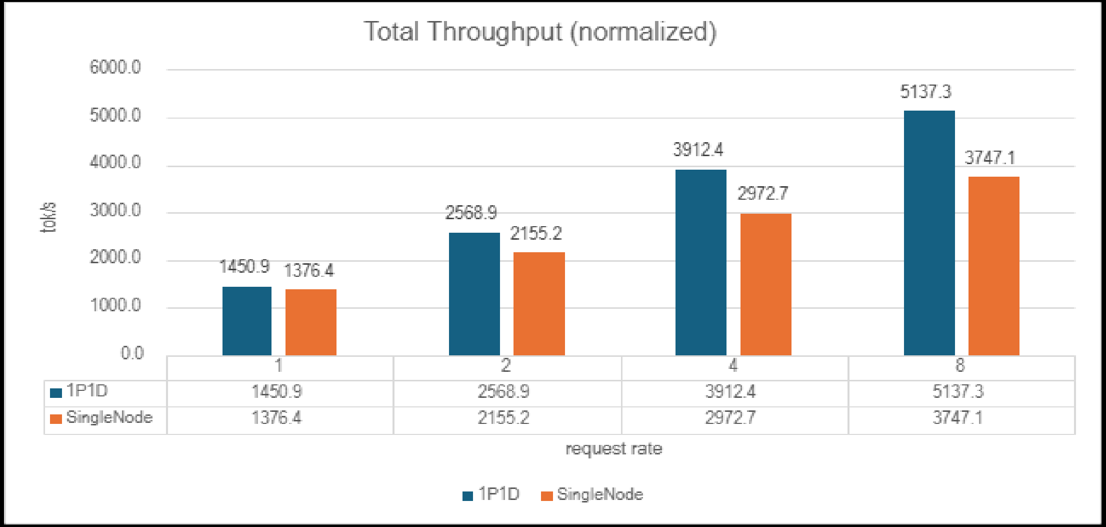


Offline generation speed, 1xMI300x

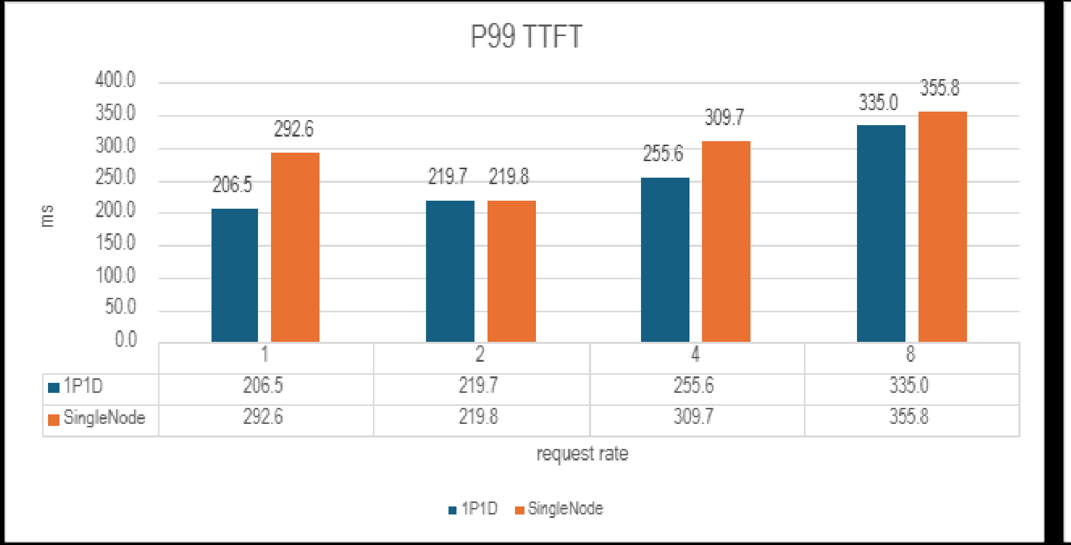
Distributed — [1P1D vs. 2×SingleNode normalized]

DeepSeek-R1

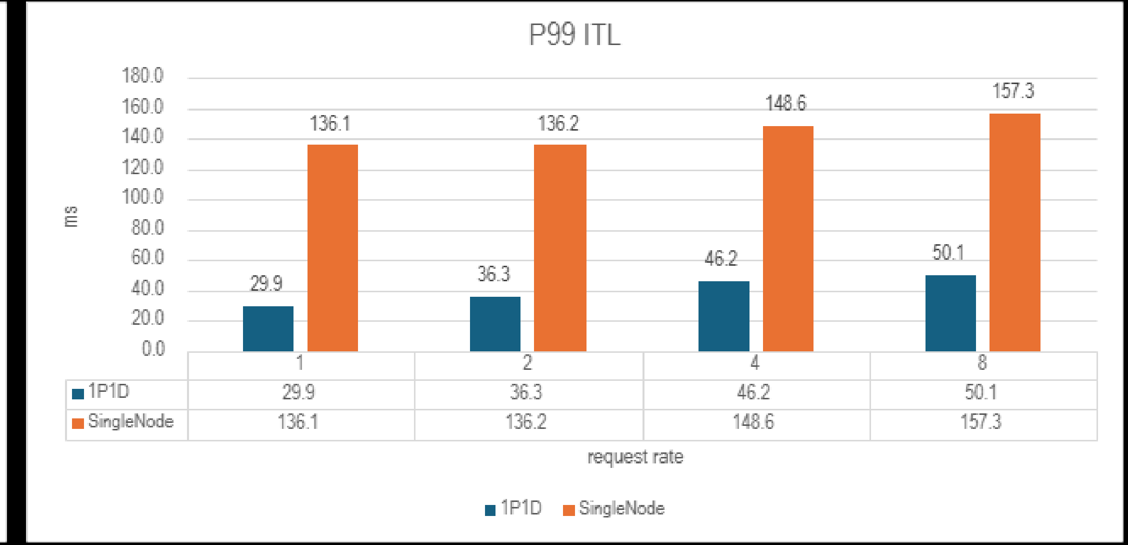
```
o  ISL [input len] = 1000
o  OSL [output len] = 600
o  Num_requests = 128
o  SGL v0.4.10.post2
o  Mooncake b63322c
o  Batch Transfer
o  Page_size = 1
o  AITER ENABLED
o  Thor2 BCM57608
o  MI300X
```



request rate	1	2	4	8
🟦 1P1D	1450.9	2568.9	3912.4	5137.3
🟧 SingleNode	1376.4	2155.2	2972.7	3747.1



request rate	1	2	4	8
🟦 1P1D	206.5	219.7	255.6	335.0
🟧 SingleNode	292.6	219.8	309.7	355.8



request rate	1	2	4	8
🟦 1P1D	29.9	36.3	46.2	50.1
🟧 SingleNode	136.1	136.2	148.6	157.3

AMD
together we advance_

Roadmap – key area: Distributed, MoRI, Rack Scale, OME

AITER — AI Tensor Engine for ROCM (AI/LLM **Compute** Blocks)

- More operators + more sizes support for more models
- Newer V3 Inference CK Attention Kernel . customizable
- https://github.com/ROCm/aiter

MoRI — Modular RDMA Interface (LLM Native **Communication** Engine)

- MoRI = DeepEP + shmem + CCL + NIXL + ⋯
- Integrating with SGLang – Q3 2025
- https://github.com/ROCm/mori

OME — Oracle Open Model Engine (Kubernetes Operator for LLM **Serving and Management**)

- Added Host and Topology Awareness
- Simplified Serving Configuration, Speed to deployment
- Aided Dynamic/Auto Scaling (LB) with optimal traffic engineering
- https://github.com/sgl-project/ome
- https://docs.sglang.ai/ome/

Wave & more AMD from Communities

- Petit Kernels ...
- UCCL/UCSHMEM…

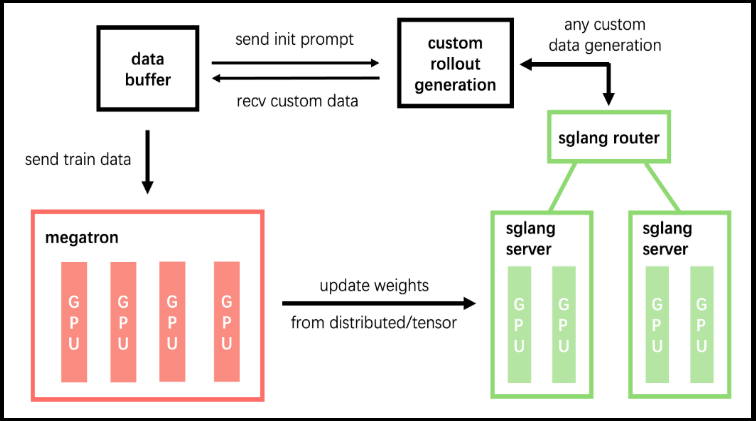
Community

Multiple ways to contribute

- Triton – most generic
- sgl-kernel – HIP, Cuda Hipified (CK, etc. upcoming)
- AITER – AMD specific & optimized
- Wave backend
- 3rd parties: Petit-kernel, etc.

slime: SGLang Native Post-Training Framework for RL Scaling

- Special hanks to the AMD GenAI - Foundation Model Team for Day-1 support.
- https://github.com/THUDM/slime/blob/main/docs/en/amd_tutorial.md
- https://lmsys.org/blog/2025-07-09-slime/
- https://github.com/THUDM/slime



OME: Open Model Engine – a Kubernetes operator that treats models as first-class resources

- **Multi-node serving:** Deploy massive models like DeepSeek V3 (685B) across multiple nodes with a simple configuration
- **Prefill-decode disaggregation:** Separate compute-intensive prefill fr. memory-bound decode, with independent scaling
- **Flexible architectures:** Both prefill and decode can run in single-node or multi-node configurations based on your needs
- **Serverless deployment:** Scale-to-zero for cost efficiency when models aren't in use
- **Business-driven scaling:** Complex autoscaling based on KV cache, tokens/second, latency targets, or custom metric
- **OCI + SGLang** backed

AMD Resources to community

- **AMD** Developer Cloud:
  https://amd.digitalocean.com/partnerships/amd/campaigns/8037ef70-7d32-11f0-a421-0a58ac14471d
- **AMD** University Program (AUP):
  https://www.amd.com/en/corporate/university-program/ai-hpc-cluster.html