What is the quantitative performance gap (in FLOPS/watt and latency) between AMD's ROCm software stack and Nvidia's CUDA stack for the top 5 foundational models (e.g., Llama-4, GPT-5, Gemma-3), and does this gap widen or narrow as model parameter counts exceed 100B?

May 30, 2026 | SnugLab Research | readme.snuglab.com

Executive Summary

The quantitative performance gap between AMD's ROCm and Nvidia's CUDA stack for foundational models narrows significantly in FLOPS/watt and latency as model parameter counts exceed 100 billion. This narrowing is primarily driven by the shift from compute-bound to memory-bound inference workloads, where AMD's superior High Bandwidth Memory (HBM) capacity and targeted software optimizations offset NVIDIA's lead in raw compute utilization and multi-node training [2, 3, 9]. For large models at high batch sizes, the performance gap shrinks to 5-10%, and AMD achieves latency parity or cost-efficiency advantages [2, 3].

Key Findings

Defining the Performance Gap

Accurately defining the quantitative performance gap requires normalization for architectural factors such as HBM bandwidth and capacity, batch size scaling, and quantization levels, rather than relying solely on raw FLOPS/watt and time-per-token metrics [2, 9]. While the AMD MI300X boasts a 32% theoretical FLOPS advantage over the NVIDIA H100, it achieves only about 45% utilization in microbenchmarks compared to NVIDIA's 93% due to software efficiency and clock drops under dense workloads [2]. However, for large foundational models, inference becomes heavily memory-bound, making HBM capacity a decisive factor [2, 9]. Quantization techniques further shift the bottleneck to memory capacity, where AMD holds an advantage, and can reduce energy consumption by 50-62%, altering FLOPS/watt efficiency [8, 9].

Performance for Foundational Models

The performance gap between ROCm and CUDA is workload-dependent, revealing a performance inversion rather than a consistent lead [2, 3, 4].

- **Models <100B Parameters:** For smaller models and low batch sizes (1-4), the NVIDIA H100 maintains a 20-30% throughput advantage [3]. For instance, in an 8-GPU configuration, the H100 delivers 31.9% lower latency than the MI300X [4].
- **Models >100B Parameters (Memory-Bound Inference):** As model parameter counts exceed 100 billion, inference workloads shift from compute-bound to memory-bound, with the decode phase dominating 77-91% of total inference time [1, 9]. In these scenarios, AMD's larger HBM capacity becomes critical. The AMD MI350X offers 288GB HBM3E compared to the NVIDIA B200's 192GB, allowing massive models like Llama 4 Maverick to fit on fewer GPUs (e.g., 3 MI350X GPUs versus 5 B200 GPUs at FP16) [2, 3]. This reduces interconnect complexity and VRAM pressure, which often bottlenecks performance before raw FLOPS for 70B-class models under concurrent load [9].
- **Quantified Gap Narrowing:** This memory advantage narrows the performance gap significantly. At higher batch sizes (64-128) typical for high-throughput serving, the throughput gap between the AMD MI355X and NVIDIA H100 shrinks to 5-10% for PyTorch and vLLM workloads [3]. For example, on the 671B-parameter DeepSeek R1 model, the MI355X achieves approximately 4,200 tokens/sec compared to the H100's approximately 4,500 tokens/sec [3]. Meta runs 100% of its live Llama 3.1 405B inference on AMD MI300X hardware, demonstrating production-level parity [2].
- **Latency:** The latency gap narrows for 100B+ models because memory-bound inference favors AMD's higher per-chip VRAM, reducing inter-GPU communication overhead [2, 3, 9]. The decode phase, which is largely insensitive to GPU frequency scaling, allows lower frequencies to achieve better energy efficiency and higher tokens per joule [1].

Multi-Node Training Performance

The performance gap widens during multi-node training, where NVIDIA's CUDA stack delivers compounding efficiency gains due to superior software maturity, interconnect technology, and higher effective GPU utilization [2, 4]. In multi-node training scenarios, the H100 outperforms the MI300X by 10-25%, with the gap expanding as scale increases [2]. Multi-GPU throughput benchmarks show NVIDIA's lead growing from 29.4% higher

throughput on 2 GPUs to 46% higher on 8 GPUs [4]. NVIDIA's mature ecosystem, including highly sophisticated runtime optimizations like automatic kernel fusion and memory access pattern optimization, contributes to this widening gap [4].

ROCm Software Optimizations

ROCm 7.0 compiler optimizations and open-source tooling contribute to the narrowing performance gap for large models. ROCm 7.0 delivered up to 3.5 times inference performance improvements over ROCm 6.0 [2]. Specific ROCm software components driving the 5-10% latency parity at high batch sizes include:

- **hipBLASLt and TensileLite Tuning:** These provide optimized General Matrix Multiply (GEMM) operations and generate tailored GEMM kernels, delivering significant speedups [13]. PyTorch TunableOp automatically selects the best-performing kernels from rocBLAS and hipBLASLt [2].
- **Flash Attention and AITER Primitives:** Flash Attention 2 reduces memory movements [2, 6], and vLLM leverages specialized AMD AITER primitives (e.g., `ROCM_AITER_FA`) for Multi-Head Attention, which use highly optimized kernels and hardware-optimized KV cache layouts to deliver 2.7-4.4x higher throughput than legacy backends [14].
- **Fleet Persistent Kernels:** Fleet is a persistent kernel runtime that launches a single kernel for the computation's lifetime, avoiding separate kernel launches per operator and delivering deterministic latency across batch sizes [12].

Efficiency and Total Cost of Ownership

ROCm's lower effective utilization (~45% versus CUDA's ~93%) creates a theoretical overhead of approximately 107% in training time and energy consumption [2]. This translates to a 10-25% slower training time for AMD hardware in real-world multi-node training workloads [2]. Consequently, while AMD GPUs may have a lower upfront hardware cost, the combination of lower software utilization and increased engineering overhead narrows the effective Total Cost of Ownership (TCO) advantage for training workloads to 15-25% [2]. However, for memory-bound inference, the efficiency gains from AMD's HBM capacity and targeted optimizations make it highly competitive in FLOPS/watt and latency [2, 3, 9].

Implications

The narrowing performance gap for large foundational model inference suggests that AMD's ROCm stack, particularly with its HBM capacity advantage, is becoming a viable and competitive alternative to NVIDIA's CUDA for deployment scenarios focused on high-throughput, memory-bound inference. This could lead to increased adoption of AMD hardware in cloud and enterprise environments for serving large language models, potentially diversifying the AI accelerator market. However, NVIDIA's continued lead in multi-node training indicates that it remains the dominant platform for large-scale model development and pre-training. Organizations must carefully evaluate their specific workload requirements-whether compute-heavy training or memory-heavy inference-to determine the most cost-effective and performant hardware and software stack.

Limitations and Caveats

Direct, comprehensive quantitative performance metrics (e.g., specific FLOPS/watt and P50/P99 latency benchmarks) for all top 5 foundational models (Llama-4, GPT-5, Gemma-3, Qwen 3, DeepSeek R1) on comparable AMD and NVIDIA hardware are not consistently available across all model sizes and batch sizes [3, 10, 11]. Specifically, detailed benchmarks for GPT-5 and Gemma-3 (especially 100B+ parameter variants, which do not exist for Gemma-3) are limited or non-existent in the provided research [5, 7, 11]. Furthermore, specific performance metrics for batch sizes of 256 or 512, and a statistical threshold for ROCm's throughput advantage, were not provided [3, 4]. The impact of ROCm's memory allocators on fragmentation and latency spikes, while acknowledged as less optimized, is not quantitatively detailed in comparison to CUDA's equivalent stacks [2].

Sources

[1] LLM Charecterization Ccgrid 2026 - shashikantilager.com - https://shashikantilager.com/assets/pdf/publications/LLM_charecterization_ccgrid_2026.pdf

[2] Amd Vs Nvidia Ai Gpu Market Share 2026 - siliconanalysts.com - https://siliconanalysts.com/analysis/amd-vs-nvidia-ai-gpu-market-share-2026

[3] [blog] Rocm Vs Cuda Gpu Cloud 2026 - spheron.network - https://www.spheron.network/blog/rocm-vs-cuda-gpu-cloud-2026/

[4] Cuda Vs Rocm - aimultiple.com - https://aimultiple.com/cuda-vs-rocm

[5] [blog] Best Ai Gpu Startups Nvidia Vs Amd - buildmvpfast.com - https://www.buildmvpfast.com/blog/best-ai-gpu-startups-nvidia-vs-amd

[6] [blog] Rocm Vs Cuda A Performance Showdown For Modern Ai Workloads - tensorwave.com - https://tensorwave.com/blog/rocm-vs-cuda-a-performance-showdown-for-modern-ai-workloads

[7] [peer-reviewed] Paper - semanticscholar.org - AUTHORS UNAVAILABLE - https://www.semanticscholar.org/paper/d242a0ce0869cc612c251d50a4ab188e59af113b

[8] [preprint] Html - arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/html/2512.16531v1

[9] [blog] Llm Inference Optimization Techniques Reduce Latency Cost - runpod.io - https://www.runpod.io/blog/llm-inference-optimization-techniques-reduce-latency-cost

[10] [preprint] Html - arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/html/2604.09048v1

[11] Llm Selection Guide - iternal.ai - https://iternal.ai/llm-selection-guide

[12] Mistral Large 3 Comparison - ai-crucible.com - https://ai-crucible.com/articles/mistral-large-3-comparison/

[13] [preprint] arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/pdf/2604.15379

[14] Model Acceleration Libraries - rocm.docs.amd.com - https://rocm.docs.amd.com/en/latest/how-to/rocm-for-ai/inference-optimization/model-acceleration-libraries.html