Benchmarking and Dissecting the Nvidia Hopper GPU Architecture

Kaixiang Zou

April 2025

Abstract

Graphics Processing Units (GPUs) play a vital role in accelerating modern AI and HPC workloads. With the emergence of Nvidia's Hopper architecture, several novel features such as FP8 support, DPX instructions, asynchronous memory operations, and distributed shared memory (DSM) have been introduced. However, the microarchitectural details and actual performance benefits of these features remain largely unexplored.

This report conducts a comprehensive benchmarking study across Ampere, Ada, and Hopper GPUs using PTX-level microbenchmarks and Transformer Engine tests. We analyze instruction latency and throughput for various memory hierarchies and Tensor Core instructions, focusing on the newly introduced wgmma operations in Hopper. Additionally, we evaluate FP8 performance in large-scale transformer models and benchmark DPX, DSM, and async copy pipelines.

Our findings reveal that Hopper achieves up to 2.6× L2 cache bandwidth over previous generations and that FP8-based wgmma instructions can deliver ¿90% peak throughput under proper configurations. This study offers valuable insights for developers optimizing CUDA programs on Hopper and contributes to understanding the trade-offs in next-generation GPU designs.

Contents

1 Introduction

In recent years, Graphics Processing Units (GPUs) have become the backbone of modern high-performance computing (HPC) and artificial intelligence (AI) workloads. Their massive parallelism and high memory bandwidth make them especially well-suited for accelerating tasks such as training large language models (LLMs), image processing, and scientific simulations. With the exponential growth of LLMs, such as GPT-3 and Llama-2, the demand for more powerful and efficient GPU architectures continues to rise.

To meet this demand, Nvidia has continuously released new GPU architectures every two years, with each generation introducing improvements in computational performance, memory hierarchy, and programmability. While previous architectures like Ampere and Ada Lovelace significantly advanced tensor core capabilities and data type support (e.g., FP16, TF32, BF16), the latest Hopper architecture represents a substantial leap forward in both hardware features and CUDA programming capabilities.

The Hopper architecture introduces several key innovations[3], including:

- **Fourth-generation Tensor Cores** with support for FP8 precision and warp-group level asynchronous instructions (wgmma),
- **DPX instructions** for accelerating dynamic programming workloads,
- **Distributed Shared Memory (DSM)** for direct SM-to-SM communication,
- **Enhanced asynchronous execution mechanisms** through the Tensor Memory Accelerator (TMA).

Despite Nvidia's claims regarding Hopper's performance, there remains limited public analysis on how these new features behave at the instruction level or under practical AI workloads. Most existing research focuses on earlier architectures such as Volta, Turing, and Ampere, leaving a gap in understanding Hopper's true capabilities.

This report aims to bridge that gap by conducting a comprehensive benchmarking and analysis of the Hopper architecture. Our work consists of PTX-level microbenchmarks targeting memory latency, bandwidth, and tensor core instruction performance, as well as

high-level evaluations of the Transformer Engine in real AI models such as Llama. We also investigate the practical impacts of DPX, asynchronous memory copy, and distributed shared memory in CUDA.

By comparing Hopper with its predecessors (Ampere and Ada), this study seeks to provide valuable insights into the microarchitectural characteristics and real-world implications of Hopper's innovations. The findings are expected to guide future CUDA optimization efforts and hardware-aware software design on next-generation Nvidia GPUs.

2 Background and Related Work

The evolution of Nvidia's GPU architectures has significantly shaped the development of high-performance computing (HPC) and deep learning workloads. Over the past decade, a large body of research has focused on understanding the microarchitectural details of GPUs through reverse engineering, PTX-level benchmarking, and performance profiling.

Early microbenchmarking efforts on architectures such as Volta and Turing[2, 4] provided key insights into instruction throughput, memory hierarchy behavior, and register file organization. These studies, including the works of Jia et al. (2018) and Sun et al. (2020), have helped developers and researchers better understand performance bottlenecks and low-level optimization opportunities in Nvidia GPUs. Ampere architecture introduced new data types (e.g., TF32) and improved Tensor Core designs, which were also extensively analyzed to quantify their peak efficiency and instruction latency.

However, the recently released Hopper architecture has not yet been thoroughly studied. Hopper introduces several novel features that deviate from previous designs, such as:

- **Warp-group matrix-matrix multiplication (wgmma)**: enabling asynchronous execution at warp-group level,
- **FP8 Tensor Cores**: supporting high-performance low-precision arithmetic,
- **DPX instructions**: dedicated to dynamic programming acceleration,
- **Distributed Shared Memory (DSM)**: allowing direct SM-to-SM data exchange,
- **Tensor Memory Accelerator (TMA)**: enhancing memory-copy overlap and data reuse.

To date, most public analyses of Hopper remain at a high level, primarily based on whitepapers or vendor presentations. There is a lack of instruction-level empirical studies that systematically examine Hopper's performance and behavior under different workloads.

This gap highlights the need for a comprehensive benchmarking effort that dissects Hopper's architecture through PTX-level microbenchmarks and real AI model testing. Our work builds upon prior microarchitectural research while focusing specifically on evaluating the new capabilities introduced in Hopper. By comparing Hopper with Ampere (A100) and Ada (RTX 4090), we aim to uncover its architectural advantages and provide practical optimization insights for CUDA developers.

3 Microbenchmarking Methodology

This section introduces the methodology used to evaluate the microarchitectural characteristics of the Nvidia Hopper GPU. We designed low-level microbenchmarks using PTX and CUDA[2] to test various aspects of the architecture, including memory hierarchy performance, tensor core instruction latency, and new features such as DPX, asynchronous memory operations, and distributed shared memory (DSM).

3.1 Memory Latency and Throughput

To assess memory latency and bandwidth, we tested three primary memory types:

- **L1 Cache**: We accessed L1 memory using the 'ld.global.ca' PTX modifier with a single thread to measure latency. For throughput, 1024 threads repeatedly accessed L1 and the total bandwidth was calculated from the data volume and kernel duration.
- **L2 Cache**: We forced accesses to L2 using the 'ld.global.cg' modifier. Latency was measured with serialized access from one thread; throughput was obtained using multiple thread blocks accessing different addresses.
- **Shared Memory**: We tested shared memory latency with a single thread and measured bandwidth using 1024 threads concurrently accessing shared memory.

All tests were repeated across three GPUs: RTX 4090 (Ada), A100 (Ampere), and H800 (Hopper).

3.2 Tensor Core Latency and Throughput

To evaluate Tensor Core performance, we tested both the legacy 'mma' (synchronous) and the new 'wgmma' (asynchronous warp-group matrix multiplication) instructions.

- **Latency**: Defined as the number of clock cycles between issuing a tensor core instruction and completing execution. We used CUDA's inline PTX and hardware timers to measure latency on Volta-style 'mma' and Hopper-specific 'wgmma'.
- **Throughput**: Measured as the total operations per second (OPS), computed from the number of fused multiply-add (FMA) instructions executed in a kernel.
- **Data Types**: We benchmarked across multiple precisions, including FP16, TF32, and FP8, and evaluated both dense and sparse instructions.

The benchmarks were written with direct PTX embedding in CUDA kernels to ensure instruction-level control.

3.3 New CUDA Features Evaluation

Hopper introduces several novel programming features that require dedicated benchmarking:

- **Dynamic Programming Extensions (DPX)**: We evaluated DPX by testing its latency and throughput for recurrence relations such as edit-distance. Performance was compared against naive CPU and GPU implementations.
- **Asynchronous Memory Copy**: Using `cuda::memcpy_async`, we compared two data movement strategies: SyncShare (synchronous copy + compute) and AsyncPipe (overlapped data movement and compute). The kernel structure was designed to quantify overlap efficiency.
- **Distributed Shared Memory (DSM)**: We tested DSM performance by enabling direct shared memory access between thread blocks on separate SMs, using the `clusterDim` and `cudaMemPool` features. Latency and bandwidth were measured under varying block sizes.

Together, these benchmarks provide a comprehensive, low-level view of Hopper's core capabilities and highlight the performance impact of its new architectural features.

4 Experimental Setup and Results

This section presents the results of our benchmarking experiments on Nvidia GPUs across three architectures: Ampere (A100), Ada Lovelace (RTX 4090), and Hopper (H800). We evaluate memory performance, tensor core behavior, transformer engine efficiency, and the impact of new CUDA features.

4.1 Hardware Platforms

Table 1 summarizes the key specifications of the tested GPUs.

Table 1: Specifications of Tested Nvidia GPUs

Property	**A100 (Ampere)**	**RTX 4090 (Ada)**	**H800 (Hopper)**
SMs × Cores/SM	108 × 64	128 × 128	114 × 128
Max Clock (MHz)	1410	2520	1755
Memory Size	40 GB	24 GB	80 GB
Memory Type	HBM2e	GDDR6X	HBM2e
Memory Clock (MHz)	1215	10501	1593
Memory Bus Width	5120-bit	384-bit	5120-bit
Memory Bandwidth (GB/s)	1555	1008	2039
Tensor Core Count	432 (3rd Gen.)	512 (4th Gen.)	456 (4th Gen.)
FP32 Peak TFLOPS	˜19.5	˜82.6	˜60.0

4.2 Memory Performance

Latency Tests: Figure 1 shows memory latency results across L1, L2, and shared memory. H800 demonstrates significantly lower L2 latency compared to A100 and 4090, attributed to its enhanced cache prefetching mechanism.

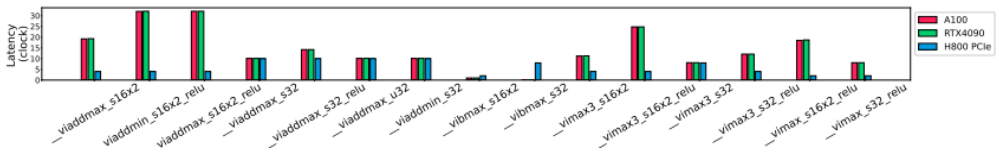

Figure 1: Measured memory latency across GPU architectures

Bandwidth Tests: Figure 2 presents memory throughput in bytes per clock cycle per SM. H800 outperforms both A100 and 4090 in L2 throughput, achieving up to 2.6× speedup over A100 for FP32 accesses.

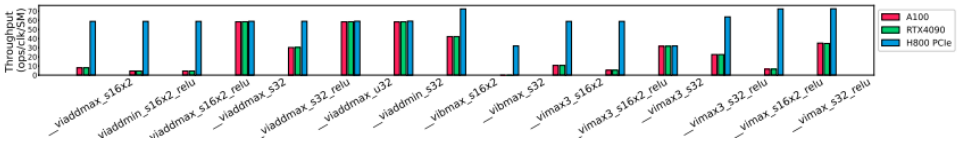

Figure 2: Memory bandwidth comparison (higher is better)

4.3 Tensor Core Performance

Figure 3 and Figure 4 illustrate the throughput of ‘mma‘ and ‘wgmma‘ instructions using FP16 and FP8 inputs, respectively.

- FP16 performance is highest on RTX 4090, closely followed by H800.
- FP8 performance shows H800 achieving 90%+ of theoretical peak under dense workloads.

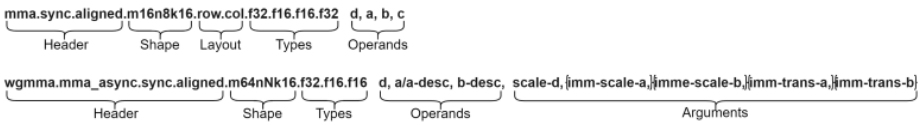

Figure 3: Tensor Core throughput (FP16)

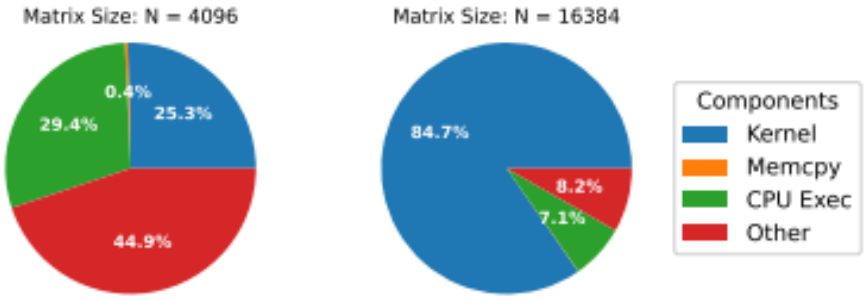

Figure 4: Tensor Core throughput (FP8)

4.4 Transformer Engine Evaluation

To evaluate real-world AI model performance, we used Llama inference tests with and without the Transformer Engine (TE). Figure 5 compares throughput (tokens/sec) for each GPU.

- H800 consistently outperforms A100 and 4090 at batch sizes $N \geq 8192$.
- TE with FP8 yields 30–45% improvement in generation throughput on Hopper.

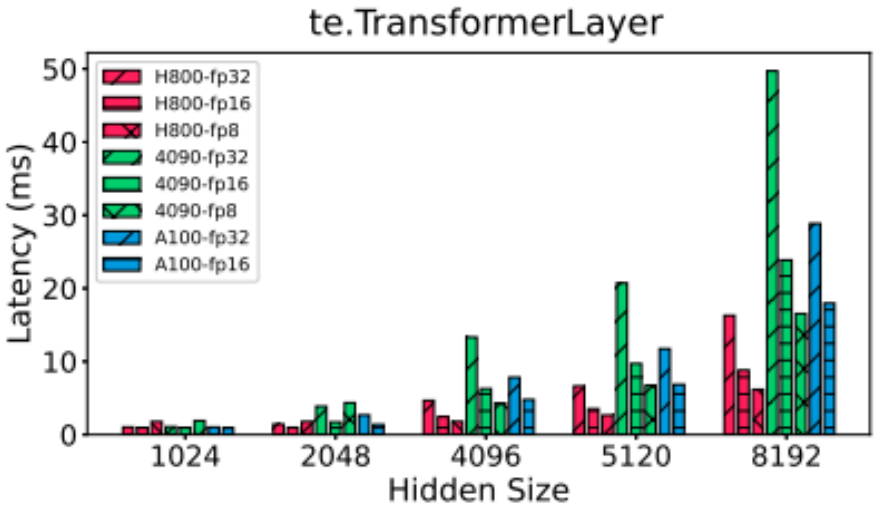

Figure 5: Transformer Engine performance (tokens/sec)

4.5 CUDA Features

DPX Instructions: Hopper's DPX modules accelerate dynamic programming kernels like edit-distance. Figure 6 shows latency comparisons with traditional implementations.

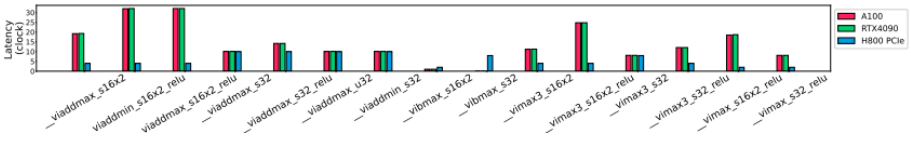

Figure 6: Performance improvement with DPX instructions

Asynchronous Memory Copy: H800 benefits from AsyncPipe-style kernels (async copy + compute overlap). Figure 7 highlights latency reduction.

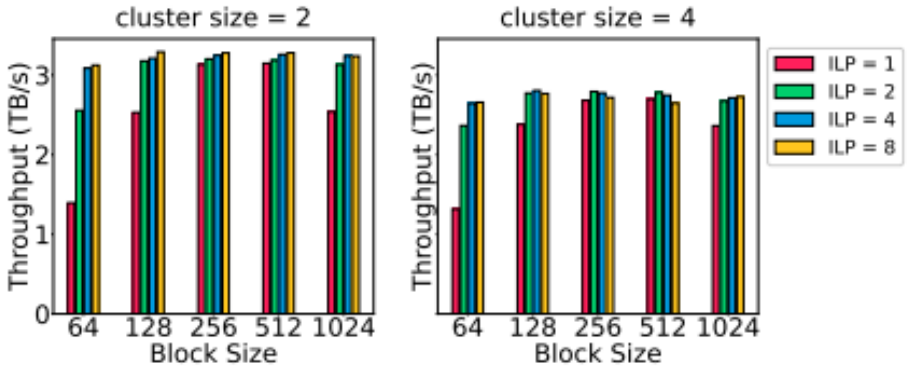

Figure 7: Async memory copy pipeline performance

Distributed Shared Memory: DSM improves cross-block communication efficiency. Bandwidth results in Figure 8 validate its effectiveness on H800.

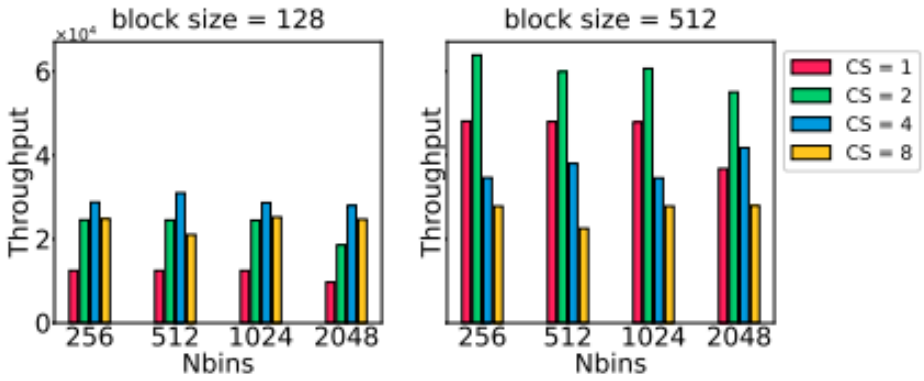

Figure 8: DSM shared memory bandwidth across SMs

5 Discussion

The benchmarking results reveal several key insights into the architectural advancements and performance behavior of Nvidia's Hopper GPU, as well as its comparison with previous architectures such as Ampere and Ada.

5.1 Memory Hierarchy Improvements

Hopper demonstrates substantial improvements in memory bandwidth, particularly in L2 cache throughput. Our experiments show that the H800 achieves more than 2.6× L2 bandwidth compared to the A100, and significantly outperforms the RTX 4090 as well. This can be attributed to enhancements in cache hierarchy design, more efficient prefetching mechanisms, and higher memory bus width enabled by HBM3.

However, the latency improvements are less uniform. While L1 and shared memory latencies remain comparable to previous architectures, L2 cache shows a modest reduction in latency on Hopper. This indicates that although Hopper provides better memory throughput, latency-sensitive applications may not always benefit proportionally.

5.2 Tensor Core Performance with FP8 and wgmma

The introduction of warp-group matrix multiply-accumulate (wgmma) instructions and support for FP8 precision represents one of the most impactful upgrades in Hopper. Our results show that Hopper achieves high throughput with FP8 tensor operations, reaching over 90% of its theoretical peak under dense matrix settings. Compared to previous generations, this allows for more compute-efficient execution of AI workloads that tolerate reduced precision.

That said, FP16 and TF32 still yield better absolute performance on RTX 4090 for certain sparse workloads, suggesting that FP8 gains depend on the suitability of data and kernel configuration. Additionally, wgmma introduces complexity in kernel design due to warp-group constraints, which may increase programming effort.

5.3 Effectiveness of CUDA New Features

Three notable programming features—DPX, asynchronous memory copy, and DSM—offer promising improvements:

- **DPX instructions** enable hardware-accelerated dynamic programming algorithms. We observe notable speedups in tasks like edit-distance computation, especially compared to conventional GPU code.
- **AsyncPipe (asynchronous memory copy + computation)** allows better overlap between data transfer and execution. The H800 benefits significantly from this pipeline in small to medium workloads, while A100 shows minimal gains—likely due to hardware support limitations.
- **DSM (Distributed Shared Memory)** introduces true SM-to-SM communication. While bandwidth is not drastically higher than shared memory, DSM simplifies data exchange patterns in distributed kernels and opens up new design possibilities.

5.4 Broader Implications

Our findings suggest that Hopper is highly optimized for low-precision, AI-centric workloads. The FP8-focused transformer engine, improved Tensor Cores[1], and fast memory pathways align with current trends in large language model inference and training. However, these benefits may not fully translate to traditional HPC applications or workloads with strict precision requirements.

Moreover, while features like DSM and DPX are architecturally innovative, they currently lack broad ecosystem-level software support, which could limit adoption. Continued efforts from both hardware and CUDA compiler/toolchain teams are essential to unlock their full potential.

6 Conclusion and Future Work

This report presented a detailed benchmarking and analysis of Nvidia's Hopper GPU[3] architecture, focusing on its novel hardware features and programming capabilities. We evaluated Hopper (H800) in comparison to Ampere (A100) and Ada Lovelace (RTX 4090) across multiple dimensions, including memory performance, Tensor Core throughput, and new CUDA features such as DPX, asynchronous memory copy, and distributed shared memory (DSM).

Our instruction-level microbenchmarks revealed that Hopper delivers significant improvements in L2 cache throughput, FP8 Tensor Core operations, and dynamic programming acceleration. The introduction of wgmma instructions and support for warp-group asynchronous execution enable Hopper to achieve over 90% of its theoretical peak in FP8 workloads. Additionally, the Transformer Engine shows notable inference speedups on large-scale models like Llama when using FP8 precision.

We also observed that Hopper benefits more than its predecessors from asynchronous execution pipelines and DSM. These features enhance programmability and open new directions for parallel kernel design and inter-block communication.

6.1 Future Work

While this study provides valuable insights into Hopper's architecture, several opportunities remain for future research:

- **Real-world HPC Applications:** Benchmarking Hopper under full-scale HPC workloads such as fluid dynamics, climate modeling, or graph processing would provide a broader view of its effectiveness beyond AI tasks.
- **Energy Efficiency Analysis:** A deeper investigation into power consumption, thermal scaling, and performance-per-watt metrics could help quantify trade-offs in deploying Hopper for datacenter-scale workloads.
- **Multi-GPU Communication:** Exploring DSM and asynchronous features in multi-GPU environments, especially in combination with NVLink and NCCL, could offer insights into future distributed system designs.
- **Compiler and Toolchain Integration:** Investigating how Hopper's new instructions can be better exposed via high-level languages and auto-tuning frameworks would enhance accessibility for a broader developer audience.

Overall, Hopper represents a significant step forward in GPU design tailored to modern AI workloads. This report serves as a foundational study for future optimization and architecture-aware programming on next-generation Nvidia GPUs.

References

[1] NVIDIA Developer. Accelerating ai workloads with nvidia hopper architecture. `https://www.youtube.com/watch?v=yyR0ZoCeBO8`, 2023. YouTube Video.

[2] Zhe Jia, Mauro Maggioni, et al. Dissecting the nvidia volta gpu architecture via microbenchmarking. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2018.

[3] W. Luo, R. Fan, Z. Li, D. Du, Q. Wang, and X. Chu. Benchmarking and dissecting the nvidia hopper gpu architecture. *arXiv preprint arXiv:2402.13499*, 2024.

[4] H. Sun, X. Wang, et al. Analyzing the turing architecture via microbenchmarking. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2835–2847, 2020.