

Benchmarking and Dissecting the Nvidia Hopper GPU Architecture

Weile Luo¹, Ruibo Fan¹, Zeyu Li¹, Dayou Du¹, Qiang Wang^{2,†}, Xiaowen Chu^{1,3,†}

Abstract—Graphics processing units (GPUs) are continually evolving to cater to the computational demands of contemporary general-purpose workloads, particularly those driven by artificial intelligence (AI) utilizing deep learning techniques. A substantial body of studies have been dedicated to dissecting the microarchitectural metrics characterizing diverse GPU generations, which helps researchers understand the hardware details and leverage them to optimize the GPU programs. However, the latest Hopper GPUs present a set of novel attributes, including new tensor cores supporting FP8, DPX, and distributed shared memory. Their details still remain mysterious in terms of performance and operational characteristics. In this research, we propose an extensive benchmarking study focused on the Hopper GPU. The objective is to unveil its microarchitectural intricacies through an examination of the new instruction-set architecture (ISA) of Nvidia GPUs and the utilization of new CUDA APIs. Our approach involves two main aspects. Firstly, we conduct conventional latency and throughput comparison benchmarks across the three most recent GPU architectures, namely Hopper, Ada, and Ampere. Secondly, we delve into a comprehensive discussion and benchmarking of the latest Hopper features, encompassing the Hopper DPX dynamic programming (DP) instruction set, distributed shared memory, and the availability of FP8 tensor cores. The microbenchmarking results we present offer a deeper understanding of the novel GPU AI function units and programming features introduced by the Hopper architecture. This newfound understanding is expected to greatly facilitate software optimization and modeling efforts for GPU architectures. To the best of our knowledge, this study makes the first attempt to demystify the tensor core performance and programming instruction sets unique to Hopper GPUs.

Index Terms—Instruction Latency, Tensor Core, PTX, Hopper, DPX, Asynchronous Execution, Distributed Shared Memory

I. INTRODUCTION

Graphics Processing Units (GPUs) have experienced a significant leap in their capacity to accelerate a wide array of applications, spanning from neural networks to scientific computing. This growth has been particularly propelled by the emergence of large language models (LLMs), where models like GPT-3, boasting over 150 billion parameters, stand as prime examples [1]. Modern GPU architectures, such as Ampere, Ada, and Hopper, embody cutting-edge features like tensor cores and high-bandwidth memory, meticulously crafted to elevate artificial intelligence applications. These

GPUs have now firmly established themselves as the bedrock of computing infrastructure in high-performance clusters.

Nvidia consistently introduces new GPU architectures every two years, incorporating advanced features. However, detailed micro-architecture information about these features is often limited, making precise quantification challenging. In-depth studies are increasingly essential to understand the impact of these advancements on application performance. The tensor core (TC) unit was initially introduced with the Volta architecture, focusing on accelerating deep neural networks with FP16 and FP32 precision operations. Subsequent Ampere architectures expanded TC capabilities to include sparsity and a broader range of data precisions such as INT8, INT4, FP64, BF16, and TF32. The Hopper architecture extended this further, introducing support for FP8 precision, significantly enhancing LLM training and inference acceleration. While a recent study [2] discussed TC programmability on Hopper, assembly code analysis and microbenchmarks were still conducted on Ampere and Turing, highlighting the need for further research specifically on Hopper tensor cores.

In addition to the new tensor core, as shown in Fig. 1, Hopper introduces innovative features: Dynamic Programming X (DPX) instructions, distributed shared memory (DSM), and an enhanced asynchronous execution mechanism (Tensor Memory Accelerator) for diverse scenarios. DPX instructions accelerate a wide range of dynamic programming algorithms, often involving numerous minimum/maximum operations for comparing previously computed solutions. DSM enables direct SM-to-SM communications, including loads, stores, and atomics across multiple SM shared memory blocks. Hopper supports asynchronous copies between thread blocks within a cluster, enhancing efficiency. However, detailed implementation and performance specifics remain undisclosed in existing literature. Unveiling these technical details is crucial for programmers to optimize AI applications effectively and leverage the new features of modern GPUs.

In this study, we conduct a comprehensive benchmarking of the latest GPU architectures (Ampere, Ada, and Hopper), focusing on key features like tensor cores and asynchronous operations. To the best of our knowledge, our research presents a pioneering analysis of the new programming interfaces specific to the Hopper architecture, offering a unique horizontal performance comparison among these cutting-edge GPU architectures. Many of our findings are novel and being published for the first time, providing valuable insights. We highlight the contributions of our work as follows.

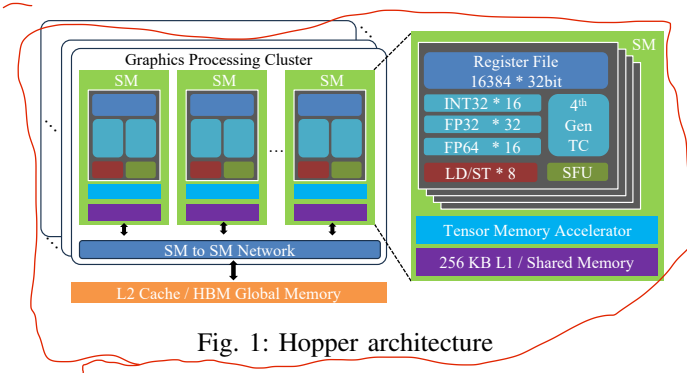
- We conduct detailed instruction-level testing and analysis

¹The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, {wluo976, rfan404, zli755, dda487}@connect.hkust-gz.edu.cn, xwchu@ust.hk

²School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China, qiang.wang@hit.edu.cn

³The Hong Kong University of Science and Technology, Hong Kong SAR, China.

[†]Corresponding authors.



on memory architecture and tensor cores across three GPU generations with different architectures. Our analysis highlights the unique advantages and potential of the Hopper architecture.

- We compare AI performance across recent GPU generations, examining latency and throughput of tensor cores at the instruction level, transformer engines at the library level, and real LLM generation at the application level. This comprehensive analysis provides valuable insights for AI system and application design and optimization, aiding informed decisions by researchers, developers, and manufacturers for next-generation AI solutions.
- Our research represents the inaugural exploration of Hopper architecture's distinctive features, encompassing DPX, asynchronous memory operations, and distributed shared memory. These innovations hold significant potential for enhancing GPU programming methodologies. Our study, by evaluating the performance of these features, contributes to performance modeling and algorithm design in dynamic programming and scientific computing. This contribution may fully unlock the GPU performance potential, driving advancements in the field.

II. RELATED WORK

Analyzing GPU microarchitectures and instruction-level performance is crucial for modeling GPU performance and power [3]–[10], creating GPU simulators [11]–[13], and optimizing GPU applications [12], [14], [15]. Early research [16]–[20] extensively dissected undisclosed GPU microarchitecture characteristics, particularly in older architectures like Fermi, Kepler and Maxwell. Jia et al. [19], [20] further evaluated the Volta and Turing GPU architectures, adding some special measurements for the new tensor cores.

Fused Matrix Multiplication Accumulation (MMA), a critical operation in AI, is predominantly accelerated by tensor cores (TCs) in Nvidia GPUs since the Volta architecture. To harness TCs' full potential, extensive research has dissected TCs across Volta, Turing, and Ampere architectures, focusing on compute throughput and register mapping. Early studies [21], [22] examined the first-generation tensor cores on Volta GPUs, emphasizing legacy *wmma* APIs and benchmarks using vendor CUBLAS and CUTLASS libraries. Subsequent work by Jia et al. [19], [20] expanded this analysis to the Turing architecture, providing preliminary assembly code analysis of

wmma APIs for TCs. However, comprehensive instruction-level microbenchmarks to fully exploit TC performance and numeric behaviors were lacking. Yan et al. and Md Aamir et al. [15], [23], [24] delved into assembly code (SASS level) benchmarking, demystifying Volta and Turing TCs. Their focus was on assembly-level optimizations for matrix multiplication performance. Additionally, they optimized half-precision matrix multiplication on TCs. A study by Fasi et al. [25] scrutinized the numeric behaviors of TCs, exploring rounding modes and subnormal behaviors of TF32, BF16, and FP16 data types. However, it's noted that *wmma* APIs have limitations in operand shapes and cannot fully leverage the new sparse matrix multiplication features on Ampere and Hopper GPUs.

On the contrary, the new *mma* programming interface was proposed since the Turing architecture and has evolved to support sparse matrix multiplication (*mma.sp*) on the architecture above Ampere. Sun et al. [2] conducted a comprehensive study of instruction-level microbenchmarks on the current *mma* APIs (*ldmatrix*, *mma* and *mma.sp*) instead of legacy *wmma* APIs, exploring the full performance of tensor cores, the computation numeric behaviors of low-precision floating points, and the new sparse matrix multiplication feature of Turing and Ampere GPUs. As for the newest Hopper, the SASS codes of *wgmma* and *mma* can be even more diverse to support a wide range of computational precisions. The usage as well as their performance is still uncovered.

In addition to performance, energy efficiency is also an often discussed factor. In addition to the above literature that models GPU power, some work also benchmarks the application level. [26] empirically investigated the impact of GPU Dynamic Voltage and Frequency Scaling (DVFS) on energy consumption and performance during deep learning, testing various GPU architectures, DVFS settings, and DNN configurations. [27] performed an empirical comparison of performance and energy efficiency across different AI accelerators from multiple vendors when training DNNs.

Despite the popularity of tensor cores and AI, another trend is the support of DPX, asynchronous operation support and distributed shared memory. To this end, benchmarking and dissecting the performance details of the modern GPU architectures is necessary and emerging. Revealing them helps programmers investigate more optimization opportunities.

III. METHODOLOGY

A. Memory Unit Access Latency and Throughput

In this subsection, we focus on two memory performance metrics: latency and throughput. Our memory test in this subsection uses a method similar to P-chase microbenchmark, which was first introduced on [28] [29]. Below we will introduce how we test these two metrics.

1) *L1 Cache*: For the latency test, we first load the data from global memory to the L1 cache using the *ca* modifier. Then we use a thread to access this L1 cache to obtain the latency. For the throughput test, we also first load the memory into the L1 cache using the *ca* modifier. Since the L1 cache is