Understanding Scalability of Multi-GPU Systems

Yuan Feng and Hyeran Jeon
Computer Science and Engineering Department
University of California Merced
{yfeng,hjeon7}@ucmerced.edu

ABSTRACT

Multi-GPU systems are considered as one of the most promising scalable accelerator systems. There have been several studies that tackled communication and scheduling efficiencies with a small-scale multi-GPU system (mostly with four GPU modules). In this paper, we examine scalability by increasing the number of GPUs. Our observations show that multi-GPU systems are yet to be scalable, mainly due to Non-Uniform Memory Access (NUMA) effects; furthermore, the state-of-the-art aggressive page distribution is one of the main reasons that increase slow remote accesses.

KEYWORDS

Multi-GPU, Characterization

ACM Reference Format:
Yuan Feng and Hyeran Jeon. 2023. Understanding Scalability of Multi-GPU Systems. In *15th Workshop on General Purpose Processing Using GPU (GPGPU '23), February 25, 2023, Montreal, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3589236.3589237

1 INTRODUCTION

With the advancement of interconnect and packaging technologies, various types of multi-GPU systems have debuted. Multi-GPU systems provide larger memory capacity through disaggregated but virtually unified memories equipped with individual GPUs, and higher parallel computing power by allowing individual jobs to be scheduled across GPUs. The most important performance bottleneck of multi-GPU systems is NUMA and Remote Direct Memory Access (RDMA) latency. To reduce remote accesses, several studies explored various hardware approaches such as cooperative thread array (CTA) scheduling, first-touch page allocation, RDMA cache for remote requests, coalesced RDMA requests, and lookup filters for TLB [1, 2]. Some studies leveraged software approaches, such as CTA colocation and remote address translation reduction through compile-time index analysis. Most of these studies evaluated their ideas on systems with a fixed number of GPUs. Though the NUMA effect can be evaluated on a small-scale system, it is unclear if the same results can be consistently observed in larger systems.

In this study, we examine the scalability of multi-GPU systems. We evaluate the performance of various workloads by increasing the number of GPUs. The GPUs are configured to use the first touch page placement and locality-aware CTA allocation [1]. We observe

GPGPU '23, February 25, 2023, Montreal, Canada

ACM ISBN 979-8-4007-0776-6/23/02.
https://doi.org/10.1145/3589236.3589237

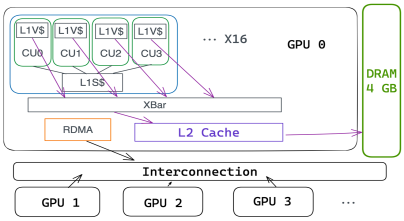


Figure 1: Evaluated Multi-GPU System

that only few workloads show scalable performance with more GPUs. The vast majority of workloads either derive zero to even negative speedup. The bottleneck mainly comes from application-agnostic page placement. In some applications, the first-touch page placement leads to imbalance page distribution such that the GPU monopolizing most of the hot pages should provide data for the other GPUs. In some other applications, even when the pages are evenly distributed, a few excessively shared pages lead to high remote accesses, which cannot be reduced through page-level data placement because excessively accessed shared data are within a small number of pages that cannot be distributed across GPUs. Therefore, we believe there needs a smarter data distribution algorithm that accommodates various application characteristics. The detailed characterizations are described in the following section.

2 METHODOLOGY

We used a cycle-level simulator, MGPUSim [3], to model a multi-GPU system. We configured GPUs based on AMD GCN3 architecture. The baseline architecture is illustrated in Figure 1. We examined scalability while integrating 2, 4, 6, 8, and 10 GPUs through an NVLink-like connection with 768 GBps bidirectional bandwidth. Each GPU is equipped with 16 Shader Arrays (SAs), and each SA consists of four Compute Units (CUs). L1 vector caches are private to each CU and L1 scalar caches are shared by all CUs in an SA. We selected a wide range of applications from different benchmark suites, including AES, FIR, KM, PR from Hetero-Mark; MM2, ATAX from Polybench; BS, MT, SC from AMD APP SDK; GUPS from HPCC benchmark, and S2D, SpMV from SHOC benchmark. We classified the applications based on memory intensity: Low (misses per kilo instructions (MPKI) < 100), Medium ($100 \leq$ MPKI < 1000), and High ($1000 \leq$ MPKI) as shown in Figure 2.

3 OBSERVATIONS AND ANALYSIS

Performance Scalability: Interestingly, some applications showed worse performance when using more GPUs. In Figure 2, PR, MM2, and SpMV show increasing execution time with more GPUs. We observed that one of the reasons for these counter-intuitive results is related to remote memory accesses. Figure 3 shows the number of incoming and outgoing RDMA requests per GPU when running

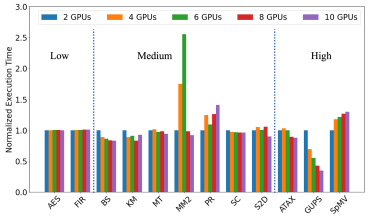


Figure 2: Execution Time Normalized by a 2 GPU System

PR on 4 to 10 GPUs. In all cases, there is one (or two) GPU(s) that receives most of the RDMA requests from the others (GPU 1 when using 4, 6, and 8 GPUs and GPU 3 when using 10 GPUs). The other GPUs barely receive any RDMA requests. This means that PR uses excessively accessed shared pages that cannot be distributed across GPUs even with the state-of-the-art first-touch page distribution algorithm [1]. Considering the execution concurrency among the GPUs, the frequently accessed pages should've been distributed to multiple GPUs. However, in PR, only one or two GPUs have all the shared pages, which means that the shared data are likely to be within a small number of pages that cannot be partitioned and mapped to multiple GPUs under page-based memory mapping. Such an imbalance RDMA traffic hinders parallelism because a certain GPU should handle a significant amount of the system-wide memory accesses. Also, the other GPUs encounter longer memory access latency due to the remote memory accesses.

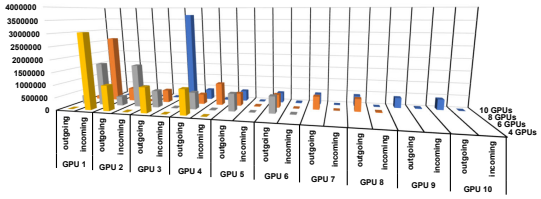


Figure 3: Per-GPU RDMA Traffic of PR

Page Distribution: One might think that the imbalanced RDMA traffic is sourced by unfair page distribution. The first-touch page distribution migrates a page from the CPU memory to a GPU's memory that accesses the page for the first time. Once the page is allocated in one of the GPU memory, it is not migrated to anywhere else. Further accesses to the page from remote GPUs are handled through RDMA. Therefore, there can be imbalanced page allocations. However, in most of the applications, the pages were almost evenly distributed as can be seen in the row marked with "Others" in Figure 4. Each color in the pie charts represents the fraction of all pages allocated in each GPU. Even in PR, pages were distributed evenly across GPUs. This confirms our understanding about PR that there is a small number of pages that are excessively shared across GPUs while the vast majority of pages are accessed less frequently and privately. Some applications such as GUPS, MT, S2D, and MM2 showed uneven page distribution as can be seen in Figure 4. In these applications, the number of incoming RDMA calls is almost proportional to the number of pages maintained in each GPU. This means that these applications make certain GPU to be a bottleneck as a data provider due to the first-touch algorithm's greedy page fetching strategy. When S2D is running on 8 GPUs, GPU 6 maintains almost half of the total pages, and hence the RDMA requests,

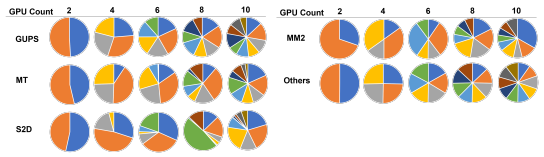
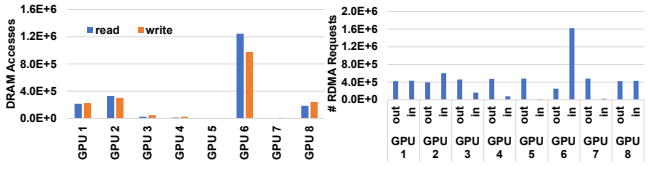


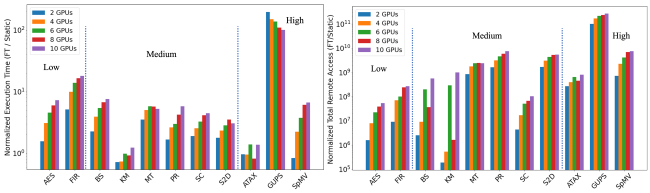
Figure 4: Page Distribution Across GPUs

as well as DRAM accesses of GPU 6, are several orders of magnitude higher than the other GPUs as can be seen in Figure 5. This is different from PR case, where pages were evenly distributed but encountered uneven RDMA calls due to a handful of excessively accessed shared pages. Some studies proposed an RDMA cache to reduce remote accesses [1]. But, as in Figure 5(a), writes are also increased, which means that RDMA caches will exhibit high cache misses due to frequent invalidations. These imbalanced RDMA requests caused by shared page monopoly made a locality-agnostic static page distribution (pages are evenly distributed across GPUs) outperform the first-touch page distribution, as plotted in Figure 6. From these observations, we believe a smarter page distribution algorithm is needed that accommodates different memory access patterns of various workloads.



(a) DRAM Accesses (b) RDMA Requests

Figure 5: DRAM and RDMA Requests of S2D on 8 GPUs



(a) Execution Time (b) RDMA Requests

Figure 6: Stats with First-Touch over Static Page Distribution

4 CONCLUSION

The emerging Multi-GPU shows new performance characteristics and optimization opportunities. This study shows unresolved limitations such as scalability and imbalance in RDMA accesses, DRAM accesses, page distribution across GPUs. Those problems need to be addressed by future research.

ACKNOWLEDGMENTS

This work was supported by NSF CCF-2114514.

REFERENCES

[1] A. Arunkumar et al. 2017. MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability. In *ISCA*.
[2] L. Belayneh et al. 2022. Locality-aware Optimizations for Improving Remote Memory Latency in Multi-GPU Systems. In *PACT*.
[3] Y. Sun et al. 2019. MGPUSim: Enabling Multi-GPU Performance Modeling and Optimization. In *ISCA*. 197–209.