Architecting Throughput Processors with New Flash

Jie Zhang and Myoungsoo Jung
Computer Architecture and Memory Systems Laboratory,
Korea Advanced Institute of Science and Technology (KAIST)
http://camelab.org

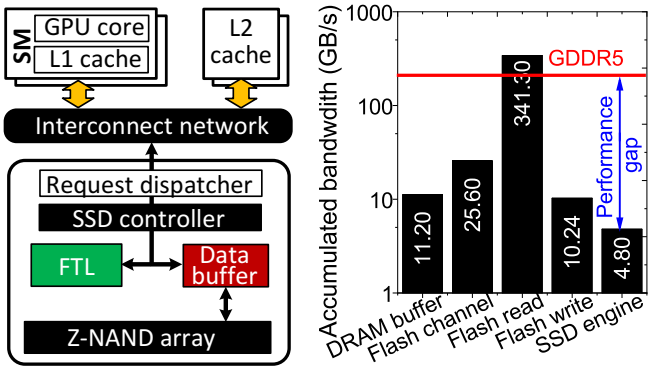
I. Introduction

Over the past few years, graphics processing units (GPUs) become prevailing to accelerate the large-scale data-intensive applications such as graph analysis and bigdata [1], because of the high computing power brought by their massive cores. However, as a peripheral device, GPUs have the limited on-board space to deploy an enough number of memory packages [2], while the capacity of a single memory package is difficult to expand due to the DRAM scaling issues such as DRAM cell disturbances and retention time violations [3]. To meet the requirement of such large memory capacity, a GPU vendor utilizes the underlying NVMe SSD as a swap disk of the GPU memory and leverages the memory management unit (MMU) in GPUs to realize memory virtualization [4]. For example, if a data block requested by a GPU core misses in the GPU memory, GPU's MMU raises the exception of a page fault. As both GPU and NVMe SSD are peripheral devices, the GPU informs the host to service the page fault, which unfortunately introduces severe data movement overhead. Specifically, the host first needs to load the target page from the NVMe SSDs to the host-side main memory and then moves the same data from the memory to the GPU memory. The data copy across different computing domains, the limited performance of NVMe SSD and the bandwidth constraints of various hardware interfaces (i.e., PCIe) significantly increase the latency of servicing page faults, which in turn degrades the overall performance of many applications at the user-level.

We propose ZnG, a new GPU-SSD integrated architecture, which can maximize the memory capacity in a GPU and address performance penalties imposed by an SSD. Specifically, ZnG replaces all GPU internal DRAMs with an ultra-low-latency SSD to maximize the GPU memory capacity. ZnG further removes performance bottleneck of the SSD by replacing its flash channels with a high-throughput flash network and integrating SSD firmware in the GPU's MMU to reap the benefits of hardware accelerations. Although flash arrays within the SSD can deliver high accumulated bandwidth, only a small fraction of such bandwidth can be utilized by GPU's memory requests due to mismatches of their access granularity. To address this, ZnG employs a large L2 cache and flash registers to buffer the memory requests. Our evaluation results indicate that ZnG can achieve 7.5× higher performance than prior work.

II. Related Work and Challenges

Baseline GPU-SSD architecture. To reduce overhead of moving data between GPU and SSD, a prior study [5], referred to as *HybridGPU*, proposes to directly replace a GPU's on-board DRAM packages with Z-NAND flash packages as shown in Figure 1a. Z-NAND, as a new type of NAND flash, achieves 64 times higher capacity than DRAM, while reducing the access latency of conventional flash media from hundreds of micro-seconds to a few micro-seconds [6]. However, Z-NAND faces several challenges to service the GPU memory requests directly: 1) the minimum access granularity of Z-NAND is a page, which is not compatible with the memory requests; 2) Z-NAND programming (writes) requires the assistance of SSD firmware to manage address mapping as it forbids in-place updates; and 3) its access latency is still much longer than DRAM. To address these challenges, HybridGPU employs a customized SSD controller to execute SSD firmware and has a small DRAM as read/write buffer to hide the relatively long Z-NAND latency.



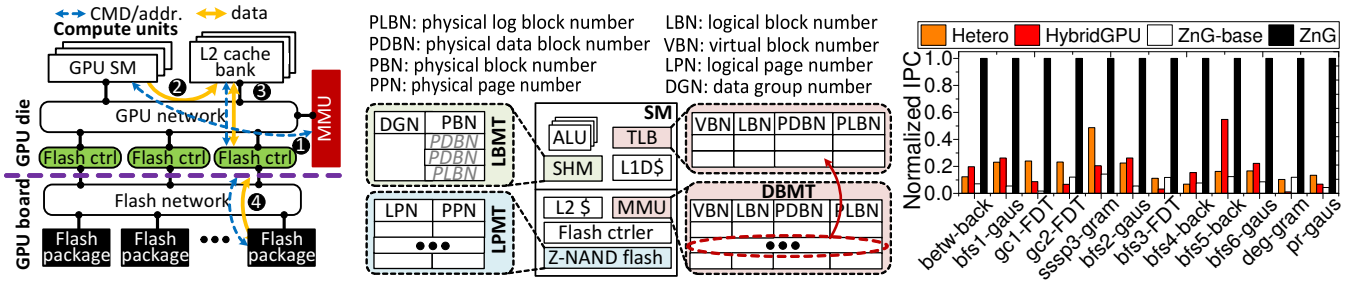
(a) HybridGPU design. (b) Bandwidth.

Fig. 1: HybridGPU architecture and the performance analysis.

Challenges. While HybridGPU can eliminate the data movement overhead by placing Z-NAND close to GPU, there is a huge performance disparity between this approach and the traditional GPU memory subsystem. To figure out the main reason behind the performance disparity, we analyze the bandwidths of traditional GPU memory subsystem and each component in HybridGPU. The results are shown in Figure 1b. The maximum bandwidth of HybridGPU's internal DRAM buffer is 96% lower than that of the traditional GPU memory subsystem. This is because the state-of-the-art GPUs employ six memory controllers to communicate with a dozen of DRAM packages via a 384-bit data bus, while the DRAM buffer is a single package connected to a 32-bit data bus. It is possible to increase the number of DRAM packages in HybridGPU by reducing the number of its Z-NAND packages. However, this solution is undesirable as it can reduce the total memory capacity in GPU. In addition, I/O bandwidth of the flash channels and data processing bandwidth of a SSD controller are much lower than those of the traditional GPU memory, which can also become performance bottleneck. This is because the bus structure of the flash channels constrains itself from scaling up with a higher frequency while the single SSD controller has a limited computing power to translate the addresses of the memory requests from all the GPU cores.

III. High Level View

Putting Z-NAND flash close to GPU. Figure 2a shows an architectural overview of the proposed ZnG. Compared to HybridGPU, ZnG removes the components of the request dispatcher, the SSD controller and the data buffer (which are placed between the GPU L2 cache and the Z-NAND flash). Instead, the underlying flash network is directly attached to the GPU interconnect network through the flash controllers. While the flash controllers manage the I/O transactions of the underlying Z-NAND, ZnG integrates a request dispatcher in each flash controller to interact with the GPU interconnect network in sending/receiving the request packets. There exist two root causes that ZnG does not directly attach Z-NAND packages to the GPU interconnect network. First, Z-NAND packages leverage Open NAND Flash Interface (ONFI) for the I/O communication whose frequency and hardware (electrical lane) configurations are different from those of the GPU interconnect network. Second, since a bandwidth capacity of the GPU interconnect network much exceeds total bandwidth brought by all the underlying Z-NAND packages, directly attaching



(a) High-level view. (b) Mapping tables. (c) Performance.

Fig. 2: Over view of ZnG design, mapping table design and performance of GPU architectures.

Z-NAND packages to the GPU interconnect network can significantly underutilize the network resources. Thus, we employ a mesh structure as the flash network, which can meet the bandwidth requirement of Z-NAND packages by increasing the frequency and link widths, rather than leveraging the existing GPU interconnect network.

Zero-overhead FTL. As the SSD controller is removed from ZnG, we offload FTL to other hardware components. GPU's MMU can be a good candidate component to implement the FTL as all memory requests leverage the MMU to translate their virtual addresses to memory logical addresses. We can achieve a zero-overhead FTL if the MMU directly translates the virtual address of each memory request to flash physical address. However, MMU does not have a sufficient space to accommodate all the mapping information of FTL. An alternative solution is to revise the internal row decoder of each Z-NAND plane to remap a request's address to a wordline of Z-NAND flash array, which is inspired by [7]. While this approach can eliminate the FTL overhead, reading a page requires searching the row decoders of all Z-NAND planes, which introduces huge Z-NAND access overhead. To address these challenges, ZnG collaborates such two approaches, which is shown in Figure 2b. Our key observation is that a wide spectrum of the data analysis workloads is read-intensive, which generates only a few write requests to Z-NAND. Thus, we split FTL's mapping table into a read-only block mapping table (cf. DBMT in Figure 2b) and a log page mapping table (cf. LPMT). To reduce the mapping table size, the block mapping table records the mapping information of a flash block rather than a page. This design in turn reduces the mapping table size to 80 KB, which can be placed in the MMU. While read requests can leverage the read-only block mapping table to find out its flash physical address, this mapping table cannot remap incoming write requests to new Z-NAND pages. To address this, we implement a log page mapping table in the flash row decoder. The MMU can calculate the flash block address of the write requests based on the block mapping table. We then forward the write requests to the target flash block. The flash row decoder remaps the write requests to a new page location in the flash block. Once all the spaces of Z-NAND are used up, we allocate a helper thread (inspired by [8]) with a block mapping table (cf. LBMT in Figure 2b) to reclaim flash block(s) via garbage collection.

Read and write optimizations. Simply removing the internal DRAM buffer imposes huge performance degradation in a GPU-SSD system owing to the disparity of GPU memory access sizes (128B) and Z-NAND page size (2KB). To address the degradation, ZnG assigns GPU L2 cache and Z-NAND internal registers as read and write buffers, respectively, to accommodate the read and write requests. Specifically, we increase the L2 cache capacity by replacing SRAM with non-volatile memory, in particular STT-MRAM to accommodate as many read requests as possible. While STT-MRAM can increase the L2 cache capacity by 4 times, its long write latency makes it infeasible to accommodate the write requests [9]. We then increase the number of registers in Z-NAND to shelter the write requests.

IV. Evaluation and Conclusion

Experiment. We use SimpleSSD [10] and MacSim [11] to model an SSD and GPU, similar to a 800GB ZSSD and NVIDIA GTX580, respectively. We implement four different GPU-SSD platforms: (1) `Hetero`: GPU and SSD are discrete devices attached to the host; (2) `hybridGPU` [5]; (3) `ZnG-base`: the baseline architecture of ZnG, which integrates the implementation of Section III without the read and write optimizations; (4) `ZnG`: putting the read and write optimizations into `ZnG-base`.

Performance. Figure 2c shows IPC values of different GPU-SSD platforms under various workload executions, and the values are normalized to `ZnG`'s IPCs. Although the GPU in `Hetero` can enjoy high bandwidth of its internal GDDR5 DRAM, data initially resides in the external SSD, which takes a long delay to move from the SSD to the GPU. As shown in the figure, `HybridGPU` outperforms `Hetero` by 31%, on average, under the workloads, `betw-back`, `bfs1-gaus`, `bfs2-gaus` and `bfs6-gaus`. As directly serving read/write requests with Z-NAND can drastically waste network bandwidth and flash bandwidth, `ZnG-base` cannot catch up the performance of `HybridGPU`. By effectively buffering data in L2 cache and flash registers, `ZnG` can fully utilize its accumulated bandwidth, which is much higher than `ZnG-base`.

In conclusion, we propose ZnG, a new GPU-SSD architecture, which can maxmimize the memory capacity in a GPU while addressing performance penalties imposed by the GPU on-board GPU. `ZnG` can achieve 7.5× higher bandwidth than `HybridGPU`, on average.

V. Original Publication

J. Zhang and M. Jung. 2020. Architecting GPU Multi-Processors with New Flash for Scalable Data Analysis. ACM/IEEE ISCA. *https://arxiv.org/pdf/2006.08975.pdf*

Acknowledgment

This research is supported by NRF 2021R1A2C4001773, DOE DE-AC02-05CH 11231, KAIST Start-Up Grant (G01190015), and Hynix grant (G01200477), and MemRay grant (G01190170).

References

[1] H. Jiang *et al.*, "Scaling up mapreduce-based big data processing on multi-gpu systems," *Cluster Computing*, 2015.
[2] K. team, "Teardown of evga geforce rtx 2080 ti xc ultra," *https://xdevs.com/guide/evga_2080tixc/*, 2016.
[3] Y. Kim, *Architectural techniques to enhance DRAM scaling*. PhD thesis, figshare, 2015.
[4] AMD, "The world's first gpu to break the terabyte memory barrier," *https://www.amd.com/en/products/professional-graphics/radeon-pro-ssg*, 2017.
[5] J. Zhang *et al.*, "Flashgpu: Placing new flash next to gpu cores," in *DAC*, ACM, 2019.
[6] S. Koh *et al.*, "Exploring system challenges of ultra-low latency solid state drives," in *HotStorage 18*, 2018.
[7] L. Yavits *et al.*, "Resistive address decoder," *https://www.researchgate.net/publication/313814525_Resistive_Address_Decoder*, 2017.
[8] N. Vijaykumar *et al.*, "A case for core-assisted bottleneck acceleration in gpus: enabling flexible data compression with assist warps," *ACM SIGARCH Computer Architecture News*, 2016.
[9] J. Zhang *et al.*, "Fuse: Fusing stt-mram into gpus to alleviate off-chip memory access overheads," in *HPCA*, IEEE, 2019.
[10] M. Jung *et al.*, "Simplessd: modeling solid state drives for holistic system simulation," *CAL*, 2017.
[11] H. Kim *et al.*, "Macsim: A cpu-gpu heterogeneous simulation framework user guide," *Georgia Institute of Technology*, 2012.