GPUdrive: Reconsidering Storage Accesses for GPU Acceleration

Mustafa Shihab, Karl Taht, and Myoungsoo Jung

Computer Architecture and Memory Systems Laboratory

Department of Electrical Engineering

The University of Texas at Dallas

Takeaways

- ***Challenge:*** File-driven data movement between the CPU and the GPU can degrade **performance** and **energy-efficiency** of GPU-accelerated data processing.
- ***Underlying Issues:***
  - **Performance disparity** in terms of device-level latencies: A storage I/O access is orders of magnitudes slower than a memory access
  - **Imposed overheads** from memory-management, data-copy, and user/kernel-mode switching
- ***Goals:***
  - Resolve performance disparity by constructing a high-bandwidth storage system
  - Optimize storage and GPU system software stacks to reduce data-transfer overheads
- ***Our Approach:*** ***GPUdrive -*** a low cost and low power all-flash array designed specifically for stream-based, I/O-rich workloads inherent in GPUs
- ***Results:*** Our prototype ***GPUdrive*** can eliminate **60% - 90%** performance disparity, while consuming **49%** less dynamic power than the baseline, on average.

Overview

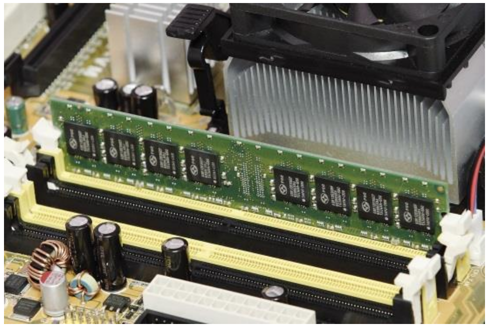
- Motivations
- GPUdrive
- Evaluations
- Related Prior Works
- Conclusion

GPU, Big Data and Storage Access

GPU-accelerated computing in big data analytics

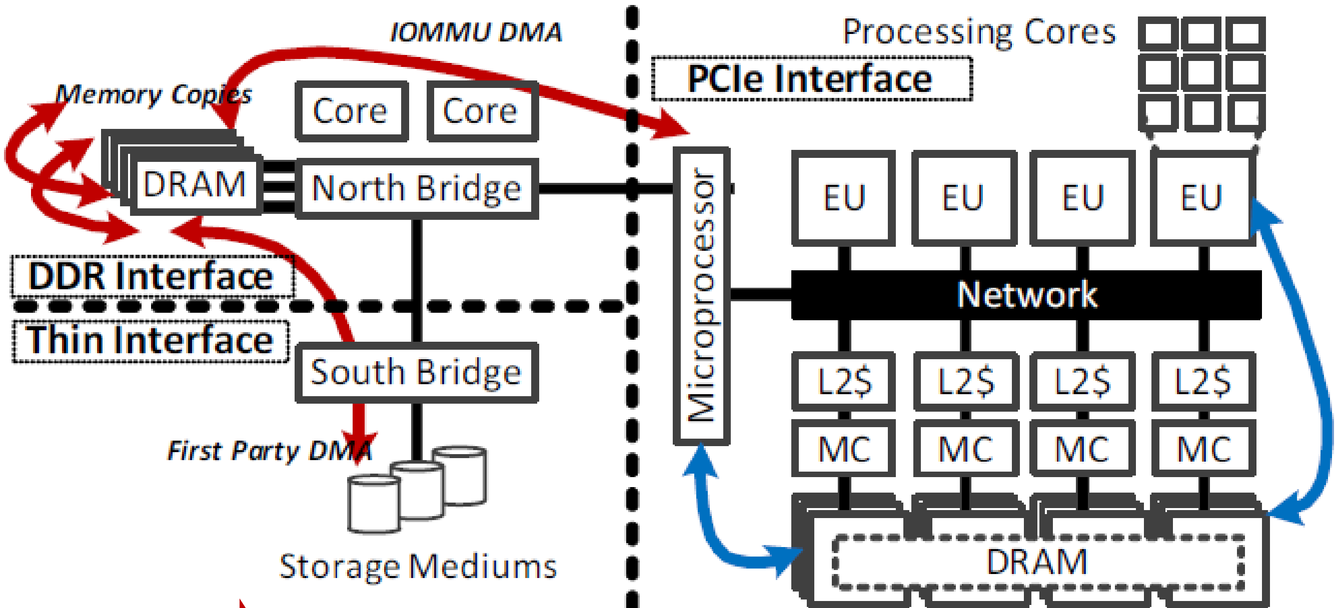
16x to 72x speed up over CPU only approach

But Big Data is too big for Memory!



So GPU has to regularly access storage devices

Storage Access in GPU Computing

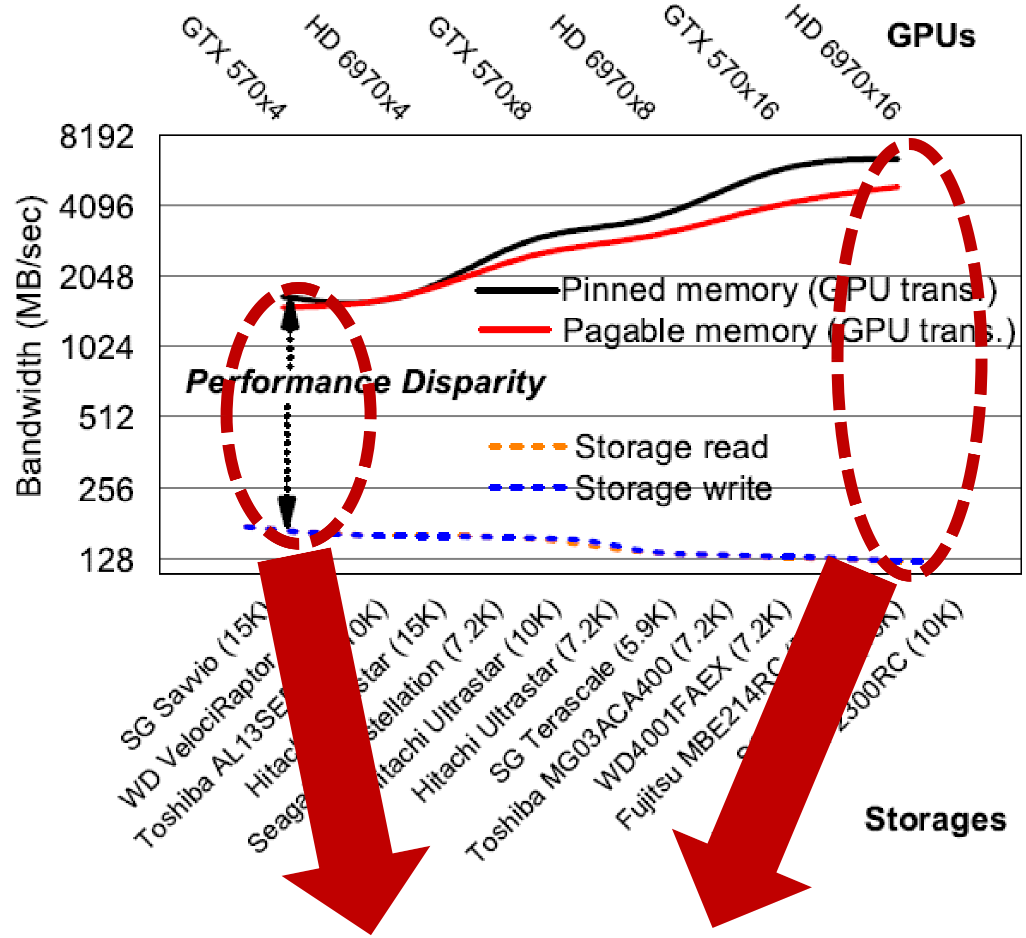


GPU-kernel accessing data from Storage

GPU-memory, CPU-memory, thin interfaces, and the storage device itself

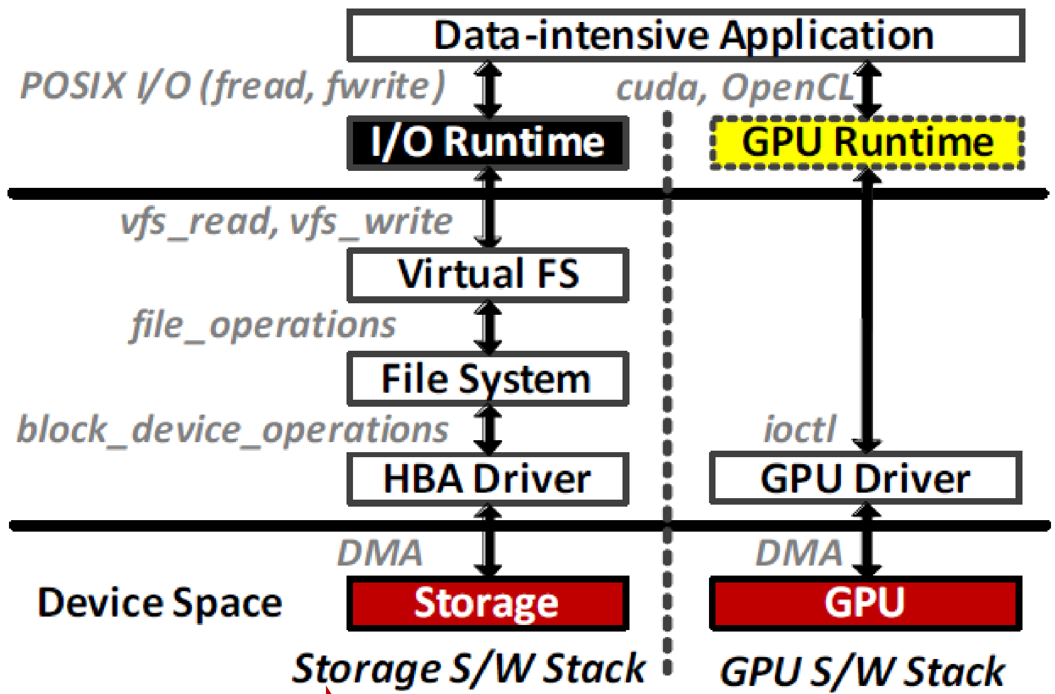
Data Transfer Situation

Numerous ill-tuned hops through the layers makes storage data transfers cumbersome and slow.



Data-transfer-rates degrade by 2000% - 8000% when the GPU applications access the storage devices.

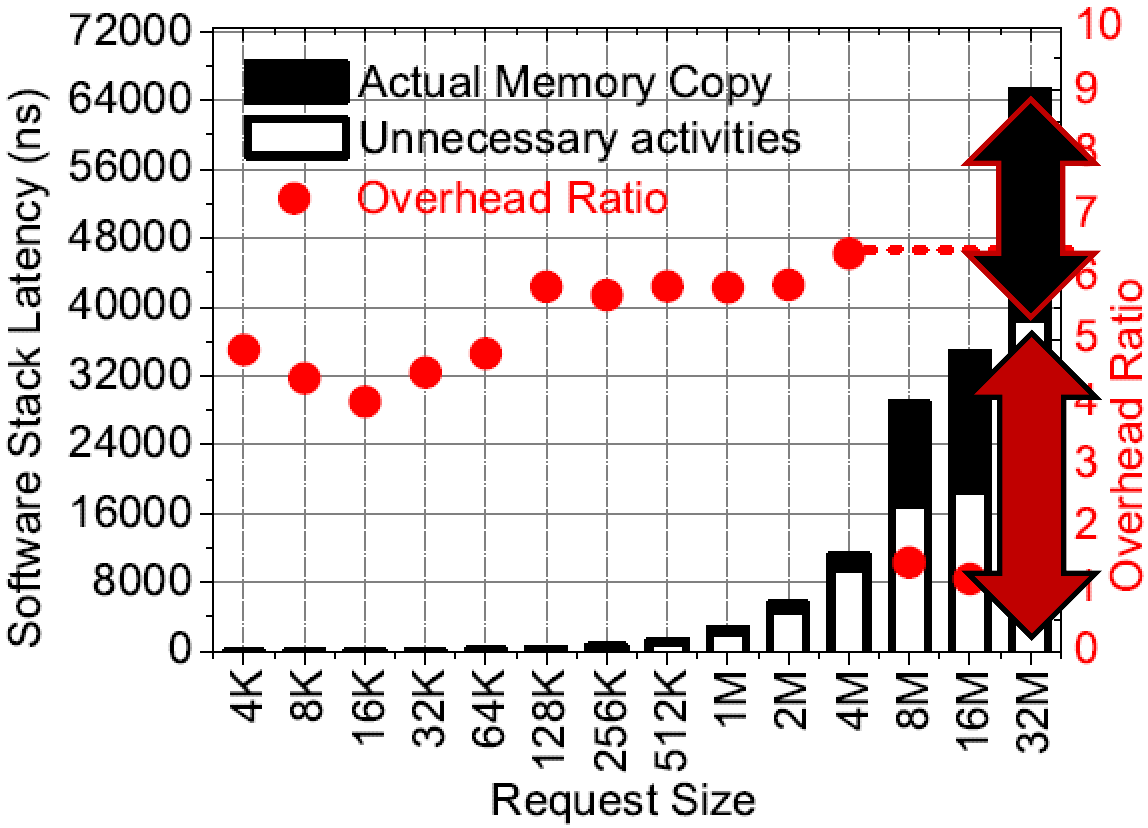
System Software Stacks



Mutually detached Storage and GPU - managed by different software stacks

Significant unnecessary overheads from data copies, memory management, and user/kernel-mode switching

Imposed Overheads

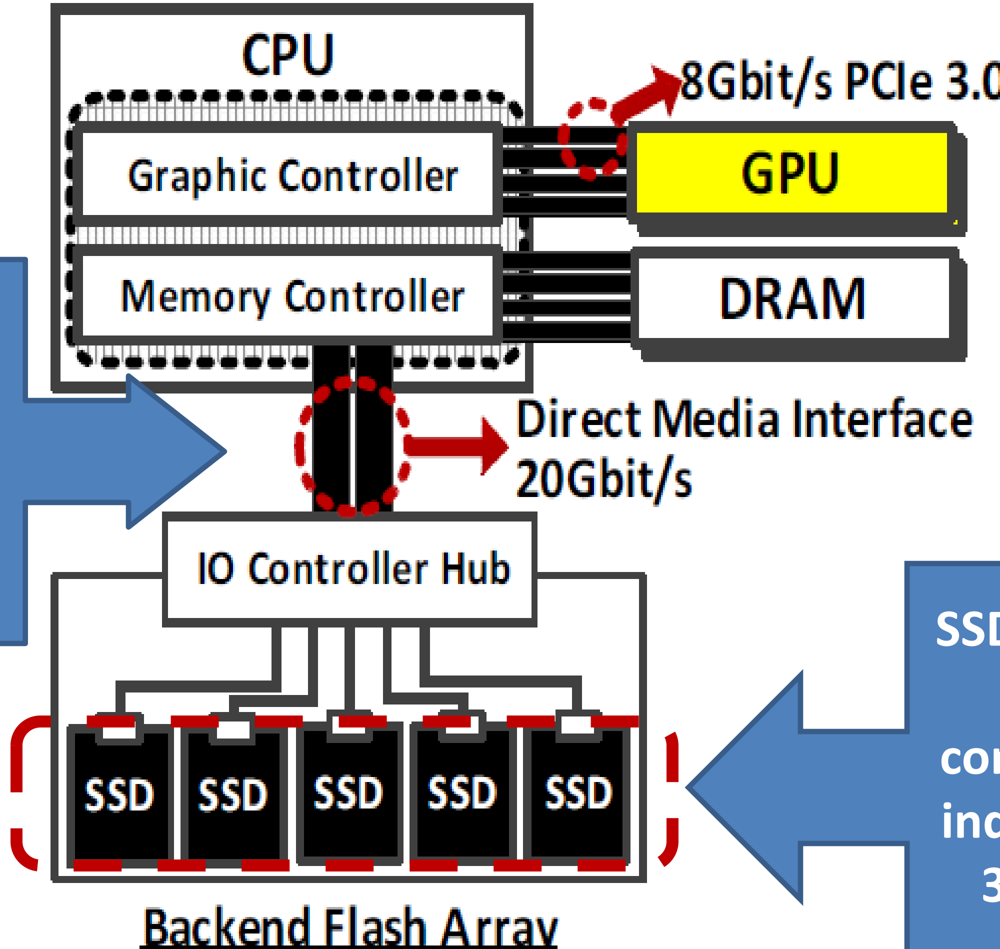


Execution times for unnecessary data copies exceeds latency related to actual data movement by 16% - 537%

Overview

- Motivations
- GPUdrive
- Evaluations
- Related Prior Works
- Conclusion

GPUdrive



SSDs bi-directionally communicate with memory controller hub over Direct Media Interface (DMI)

SSDs connected to the I/O controller with individual SATA 3.0 physical channel

Overview

- Motivations
- GPUdrive
- Evaluations
- Related Prior Works
- Conclusion

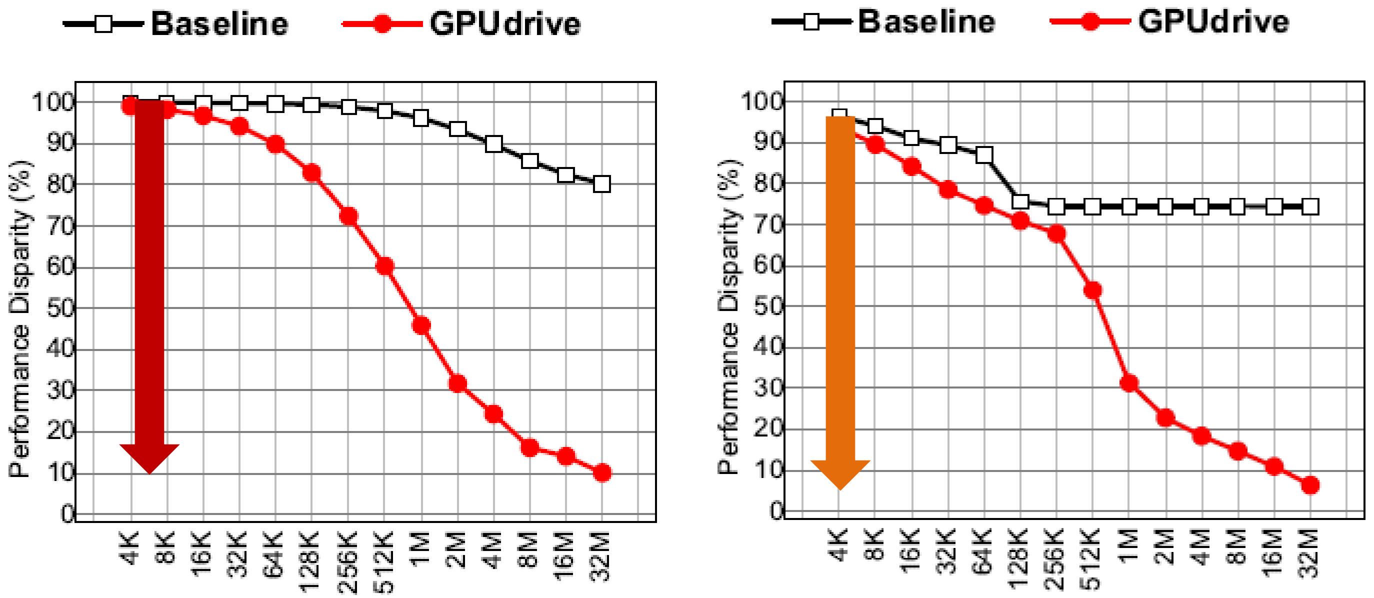
Experimental Setup

Host Evaluation Platform	Intel Core i7 with 16GB DDR3 Memory
GPU	NVIDIA GTX 480 (480 CUDA cores) with 1.2GB DDR3/GDDR5 memory
Host – GPU interface	PCI Express 2.0 x16
Baseline System	Enterprise-scale 7500 RPM HDDs
GPUdrive Prototype	SATA-based SSDs
Benchmark Applications	NVIDIA CUDA SDK and Intel IOmeter (with modified codes)
Benchmarks	bench-rdrd: random read bench-sqrd: sequential read bench-rdwr: random write bench-sqwr: sequential write

This is the *preliminary* evaluation

Upload Performance Analysis

- Performance disparity reduction

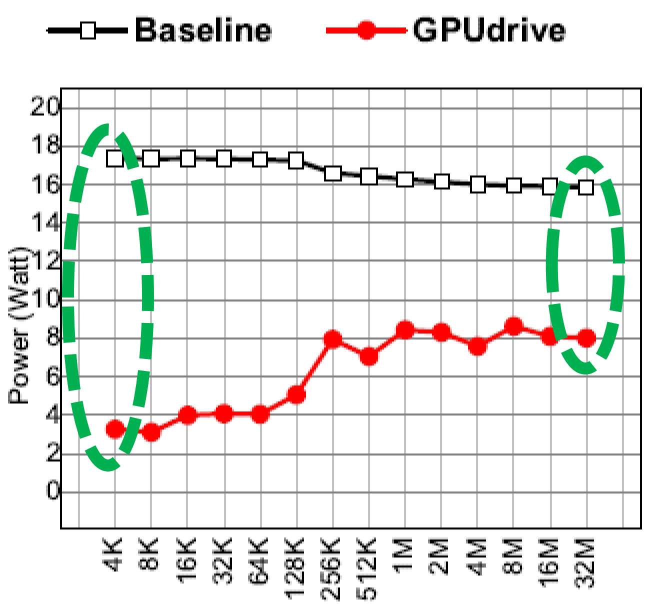


(a) bench-rdrd (b) bench-sqrd

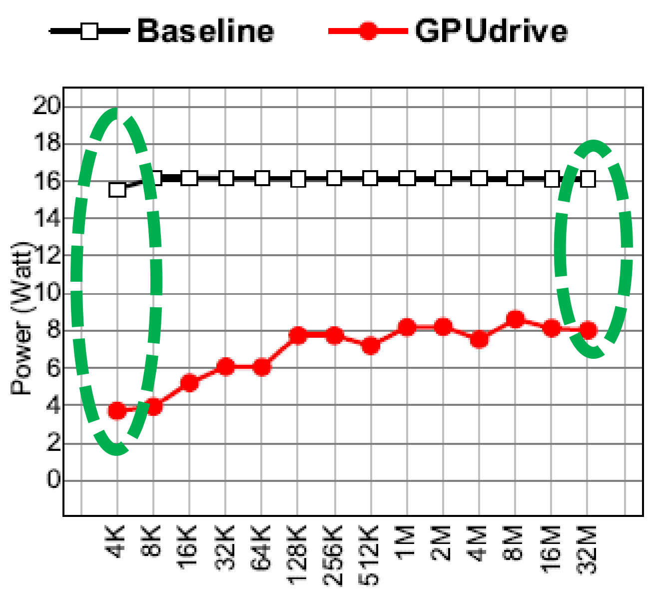
GPUdrive prototype reduces the performance disparity between the CPU and the GPU on bench-rdrd and bench-sqrd by 90% and 92%, respectively.

Upload Performance Analysis

- Dynamic Power Analysis



(a) bench-rdrd

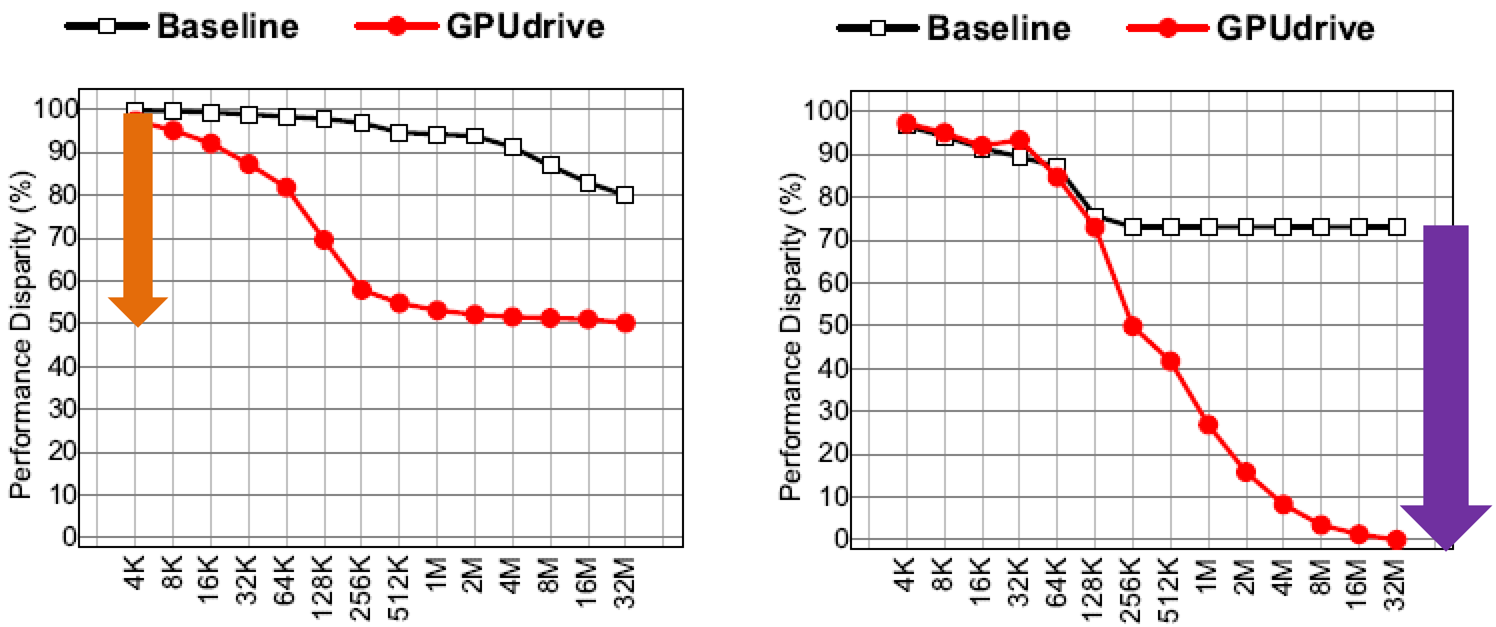


(b) bench-sqrd

GPUdrive prototype requires 77% - 52% less dynamic power than the baseline storage array

Download Performance Analysis

- Performance disparity reduction



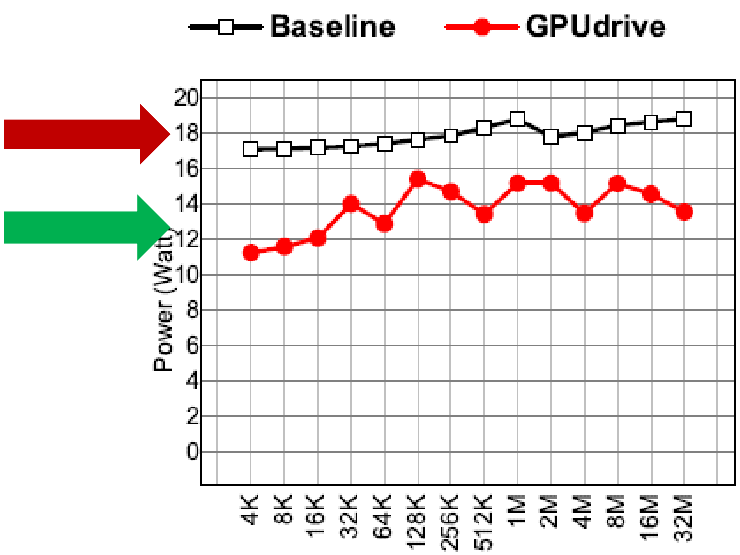
(a) bench-rdwr (b) bench-sqwr

bench-rdwr: reduction rates on downloads are limited

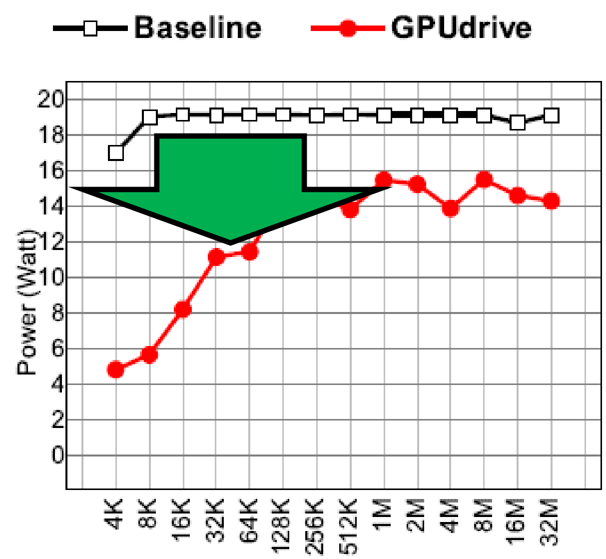
bench-sqwr: GPUdrive successfully removes the performance disparity in the case of large I/O requests (32MB)

Download Performance Analysis

❑ Dynamic Power Analysis



(a) bench-rdwr



(b) bench-sqwr

- ❑ bench-rdwr: baseline consumes **18 watts,** whereas GPUdrive consumes **13 watts**, irrespective of the request sizes.
- ❑ bench-sqwr: GPUdrive prototype require on average **30%** less dynamic power than the baseline

Overview

- Motivations
- GPUdrive
- Evaluations
- Related Prior Works
- Conclusion

Related Prior Works

- **Shinpei Kato et al.** presented a ***zero-copy I/O processing*** scheme in [7] to reduce computation cost and latency by mapping the I/O address space to the virtual address space and allowing data transfer to and from the compute device directly.

- **Daniel Lustig et al.** proposed a CPU-GPU synchronization technique in [8] that shortens the offload latency by employing ***fine-granularity data transfer**, **early kernel launch**, and a **proactive data return mechanism***.

- Also, in the industry, techniques such as **NVIDIA's *GPUDirect***, ***pinned memory***, and ***unified virtual addressing (UVA)*** *are used to* manage memory-level data transfers between the CPU and the GPU.

Overview

- Motivations
- GPUdrive
- Evaluations
- Related Prior Works
- Conclusion

Conclusion

- Data movement between the CPU and the GPU can degrade **performance** and **energy-efficiency** of GPU-accelerated data processing.

- *We propose **GPUdrive** -* a low-cost and low-power **all-flash array,** designed specifically for the workloads inherent in GPUs, with **optimized** storage and GPU system software stacks.

- Our prototype **GPUdrive** can eliminate **60% - 90%** performance disparity, while consuming **49%** less dynamic power than the baseline, on average.

- We are working on to extend the findings of these preliminary evaluations.

Thank you

Questions?

Reference

[1] Ranieri Baraglia et al. Sorting using bitonic network with cuda, 2009.
[2] Wenbin Fang et al. Mars: Accelerating mapreduce with graphics processors. TPDS, 2011.
[3] C. Gregg and K. Hazelwood. Where is the data? why you cannot debate cpu vs. gpu performance without the answer. In ISPASS, 2011.
[4] Intel. Iometer User's Guide. 2003.
[5] Myoungsoo Jung and Mahmut Kandemir. Revisiting widely held ssd expectations and rethinking system-level implications. In SIGMETRICS, 2013.
[6] S. Kato et al. Rgem: A responsive gpgpu execution model for runtime engines. In RTSS, 2011.
[7] Shinpei Kato et al. Zero-copy i/o processing for low-latency gpu computing. In ICCPS, 2013.
[8] Daniel Lustig et al. Reducing gpu offload latency via fine-grained cpu-gpu synchronization. In HPCA, 2013.
[9] Mellanox. Nvidia gpudirect technology accelerating gpu-based systems. http://www.mellanox.com/pdf/whitepapers/TB_GPU_Direct.pdf.
[10] NVDIA. Nvidia cuda library documentation. http://docs.nvidia.com/cuda/.
[11] NVIDIA. Gpu-accelerated applications. http://www.nvidia.com/content/tesla/pdf/gpuaccelerated-applications-for-hpc.pdf.
[12] Nadathur Satish et al. Designing efficient sorting algorithms for manycore gpus, 2009.
[13] Tim C. Schroeder. Peer-to-peer and unified virtual addressing. 2013.
[14] Jeff A. Stuart and John D. Owens. Multi-gpu mapreduce on gpu clusters. In IPDPS, 2011.
[15] RenWu et al. Gpu-accelerated large scale analytics, 2009.