Scheduling-Aware Prefetching: Enabling the PCIe SSD to Extend the Global Memory of GPU Device

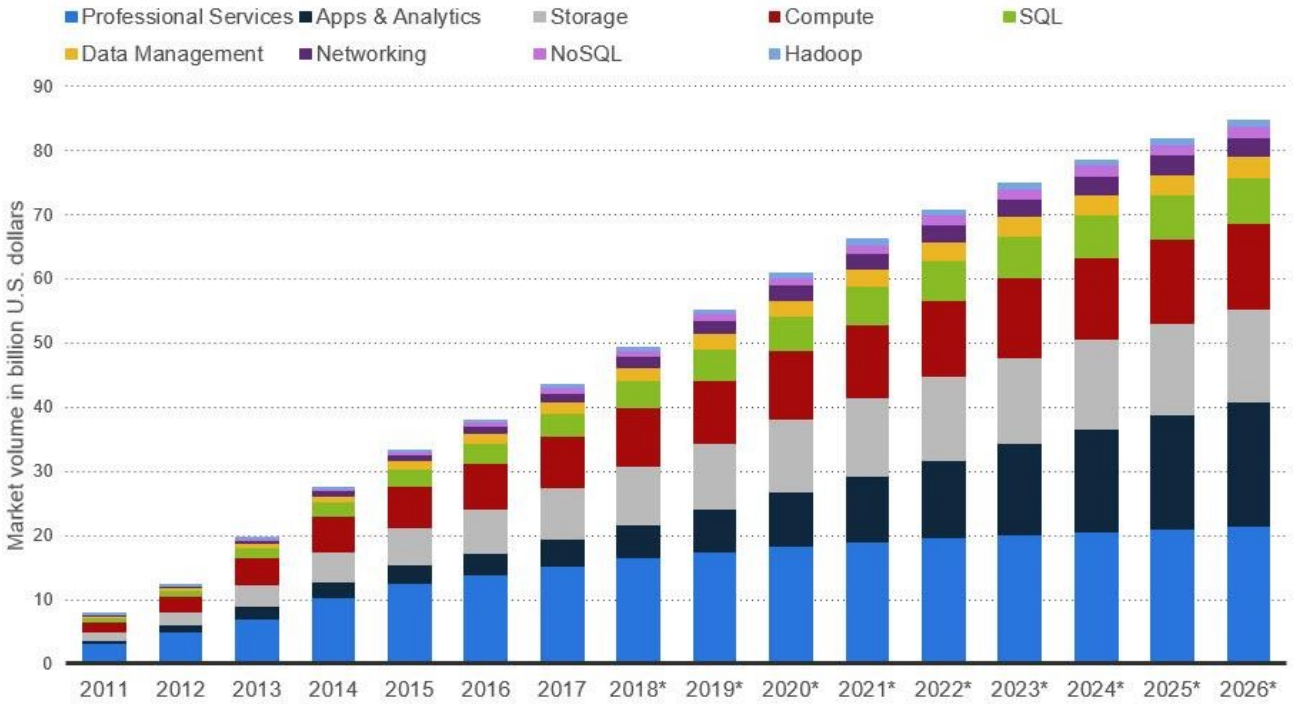
Tse-Yuan Wang[1,2] , Chun-Feng Wu[1,2] , Che-Wei Tsao[1,2] , Yuan-Hao Chang[2] , and Tei-Wei Kuo[1,3]

[1]Department of Computer Science and Information Engineering, National Taiwan University,
[2] Institute of Information Science, Academia Sinica
[3]Hong Kong Institute for Advanced Study, Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

Outline

- Background and Motivation
- Scheduling-Aware Prefetching
- Performance Evaluation
- Conclusion

Big Data Market Forecast Worldwide from 2011 to 2026



Source: https://www.forbes.com/sites/louiscolumbus/2018/05/23/10-charts-that-will-change-your-perspective-of-big-datas-growth/#39a38dbf2926

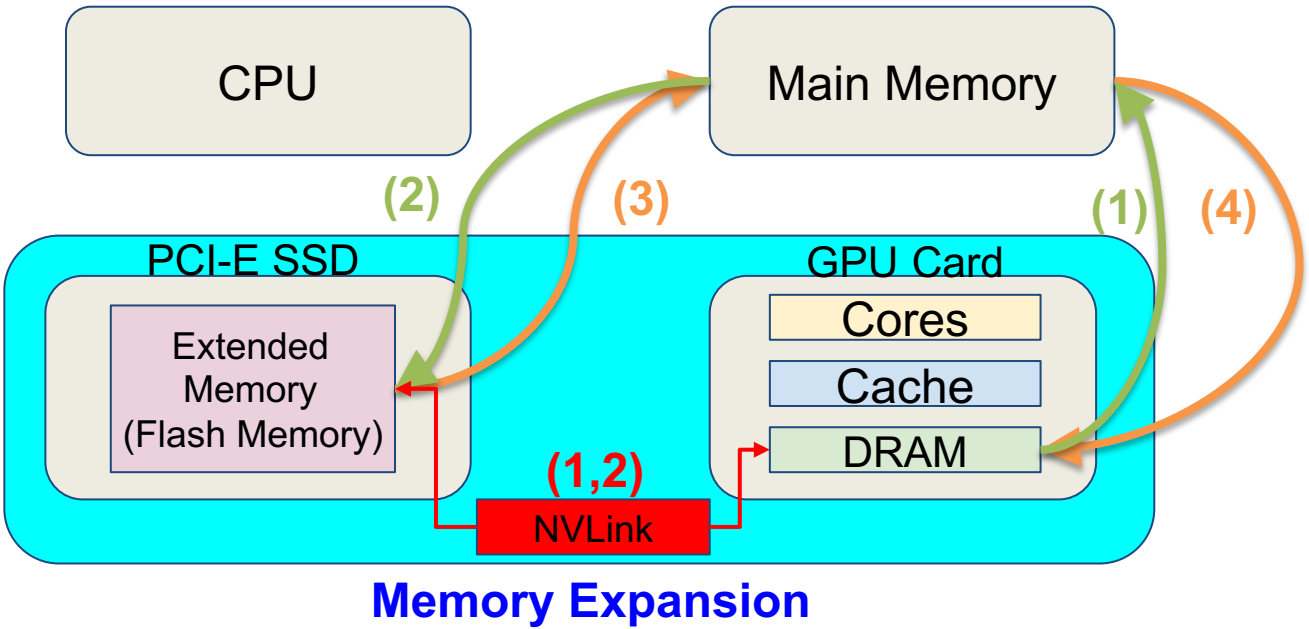
Observation

- GPUs are widely used to accelerate these data-intensive big data applications, such as computer graphic and deep learning
- These applications also need a huge DRAM size of GPU device to as the temporary storage for data models
- The growth rate of the required memory size of applications is higher than the process technology of DRAM
  - High leakage power with huge DRAM size

Observation

- NVIDIA provides Unified Memory Architecture (memories of Host and GPU inside the same address space)
  - Reducing the complexity of writing the CUDA program (programmers do not need to consider the data movement between Host and GPU device)
  - Ultra-scale memory drive with high access latency is not considered
- The new interface “RDMA” between the PCIe device and GPU device is provided to reduce the loading of host
- GPU is data parallelism, not task parallelism, and the segments of the program are decided to perform by the internal warp schedulers. The programmer and host are difficult to catch or create the data access locality.

The Complex Data Movement among Host, GPU Device, and PCIe SSD



Motivation

- Performance degradation
  - Although GPUs can directly access PCIe SSD without moving the data to host DRAM beforehand, it leads to serious performance degradation due to the large performance gap between the PCIe SSD and the off-chip DRAM in the GPUs
  - How to pre-fetch the data between DRAM and Flash Memory?

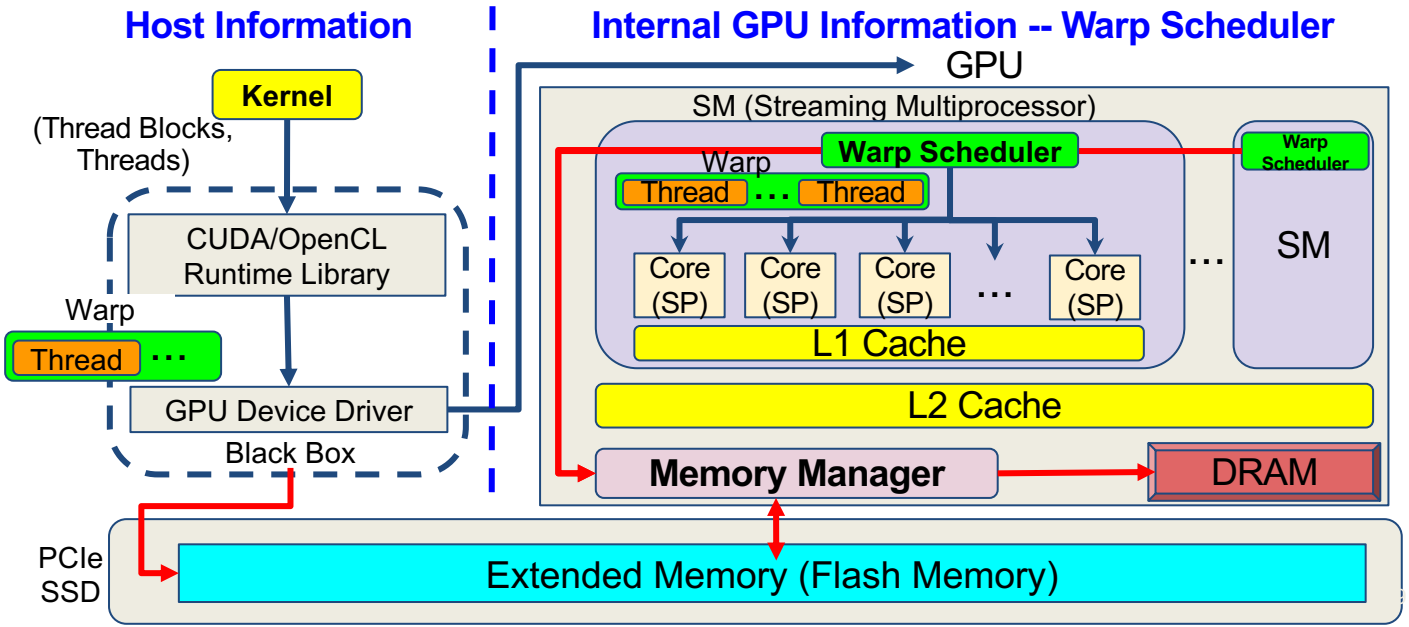
Type	DRAM	Flash SLC
Read time	60 ns	50 us
Write time	60 ns	550 us

Outline

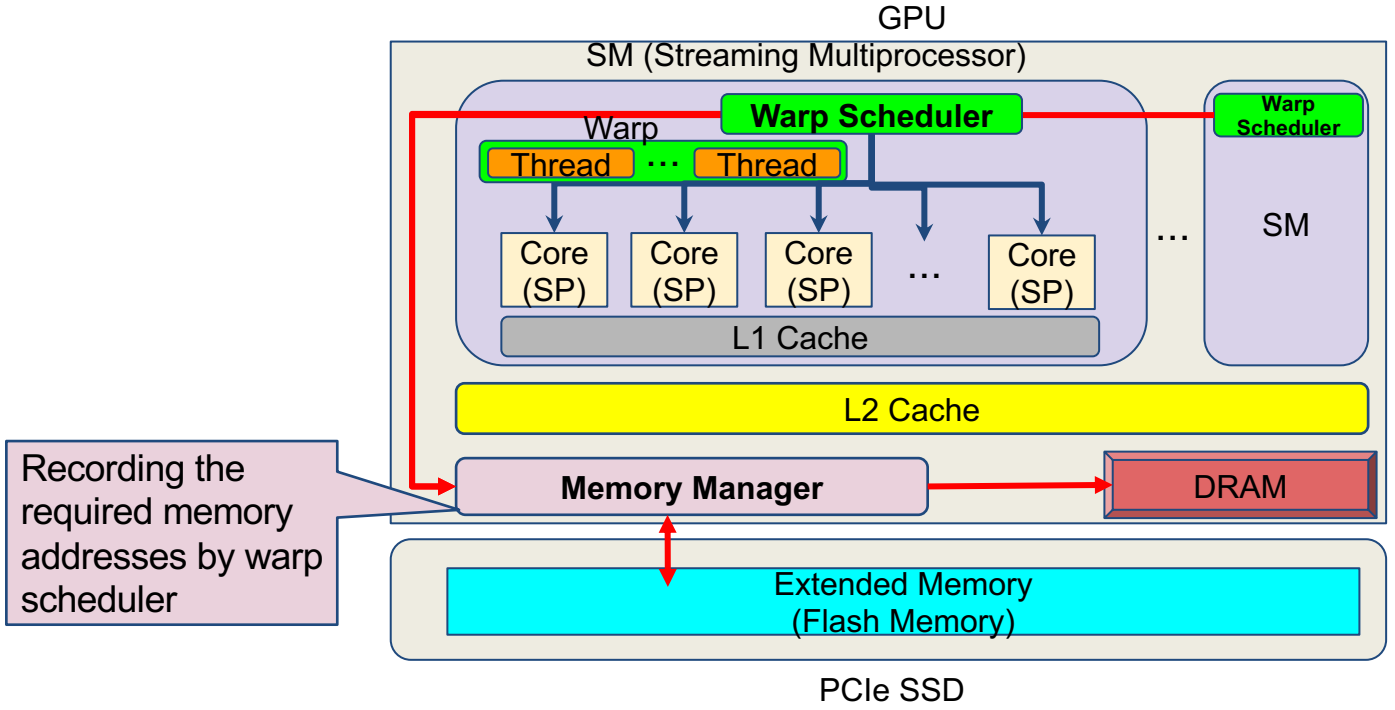
- Background and Motivation
- Scheduling-Aware Prefetching
- Performance Evaluation
- Conclusion

System Architecture

- Without modifying the host design (e.g., GPU device driver) and the programming way of the programmer
  - A new component **Memory Manager** is needed to coordinate between DRAM and SSD with considering the warp schedulers



Internal GPU Information -- Warp Scheduling

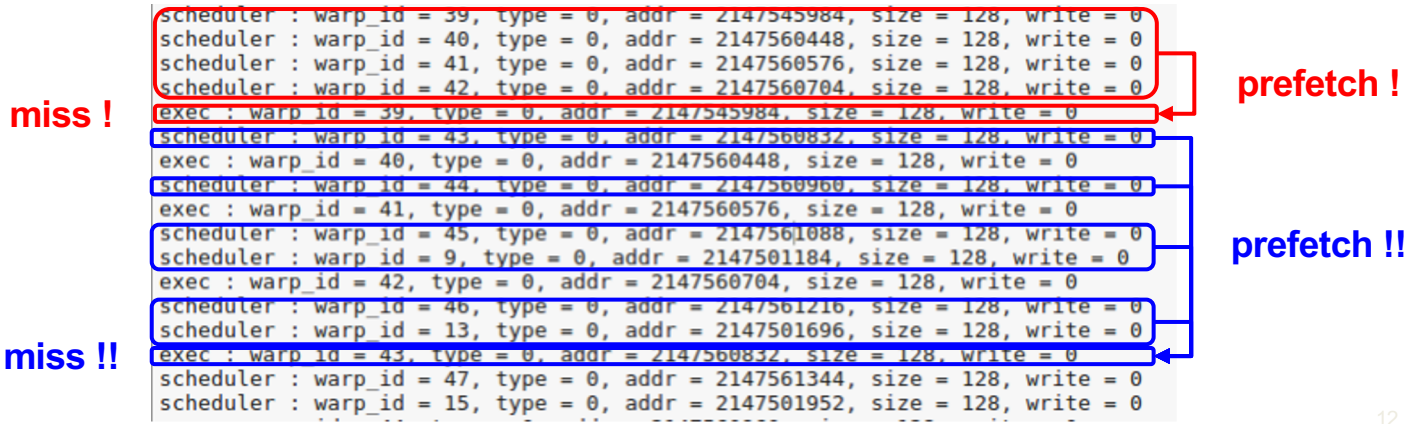


Internal GPU Information -- Warp Scheduling (cont.)

- The instructions are issued and executed per **warp**, and **memory operations** are also issued per warp
  - When executing a memory instruction, **each thread in a warp** provides a memory address which is loading or storing
- Memory Behavior Tracking
  - Warp = **32 threads**
  - The required memory addresses of threads in a warp are **continuous**
  - The memory request unit at once is based on the **warp**

Internal GPU Information -- Warp Scheduling (cont.)

- When op_code of the warp in warp scheduler is **LOAD** or **STORE**, **the required memory addresses** will notify **Memory Manager**
- When data of the accessed memory addresses are **missing in DRAM**, Memory Manager uses warp scheduler information to **pre-fetch** data to DRAM



Outline

- Background and Motivation
- Scheduling-Aware Prefetching
- Performance Evaluation
- Conclusion

Experiment Setup

Evaluated Unified Memory

Type	Access Latency
DRAM Random Access Latency (Read/Write)	60 ns
Flash Memory Random Access Latency (Read)	50,000 ns
Flash Memory Random Access Latency (Write)	550,000 ns
Flash Memory Serial Access Time (ns/Bytes)	5 ns

Benchmarks
ISPASS2009-benchmarks – AES, BFS, CP, LPS, MUM, and NN
Memory Trace and Warp Scheduler Information are collected by the modified GPGPU-Sim

Compared Strategies: LRU and FIFO

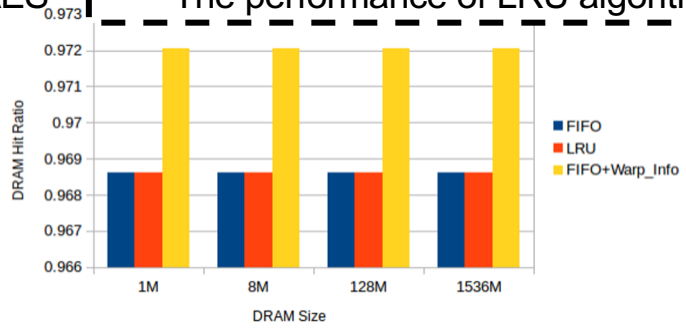
Source: https://github.com/gpgpu-sim/
https://github.com/gpgpu-sim/ispass2009-benchmarks

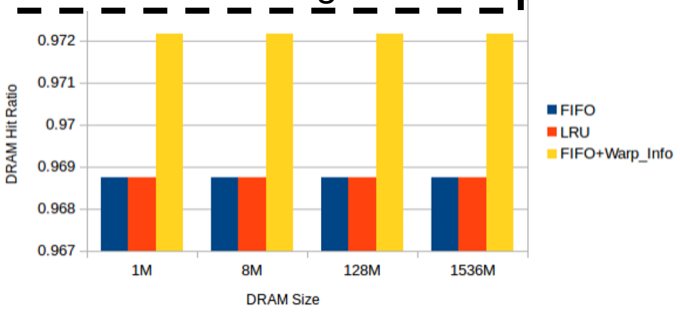
DRAM Hit Ratio on Different DRAM Size with Different Policy

AES

The performance of LRU algorithm is same to FIFO algorithm

CP

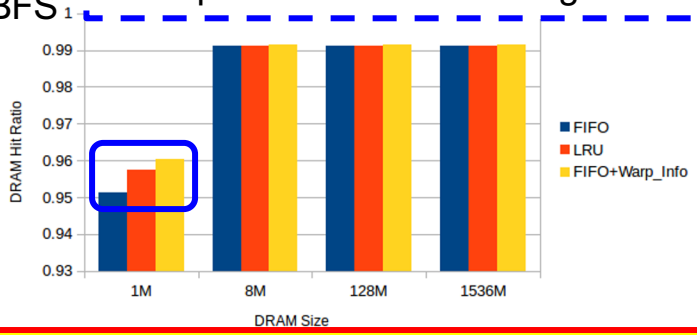


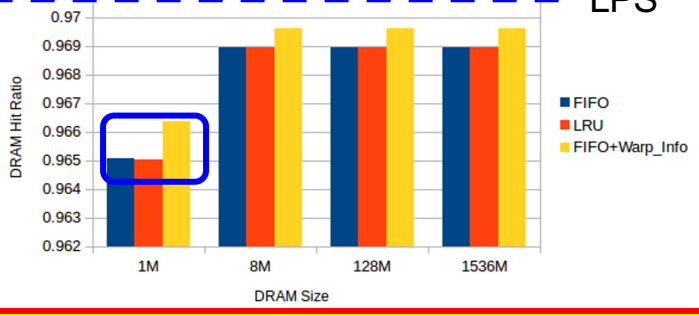


BFS

The performance of LRU algorithm is higher/lower thanFIFO algorithm

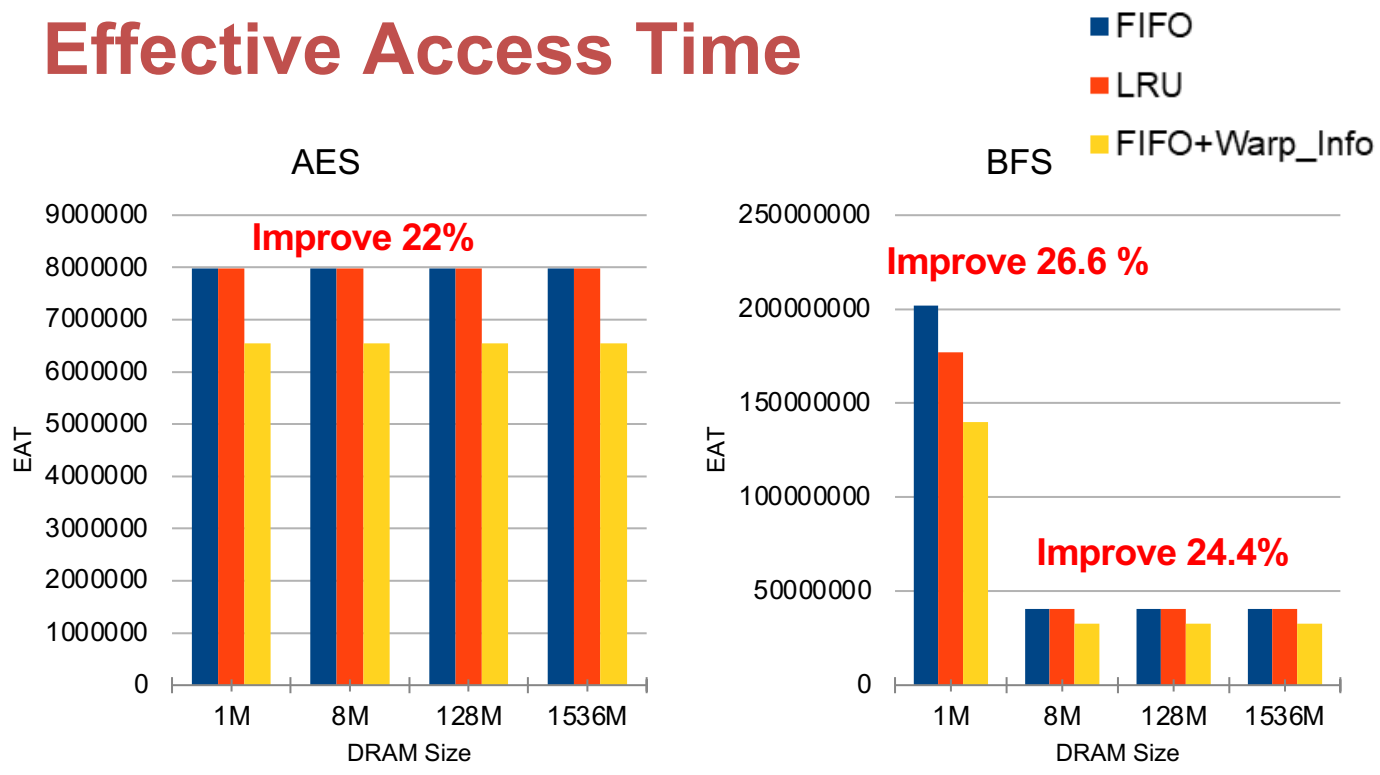
LPS



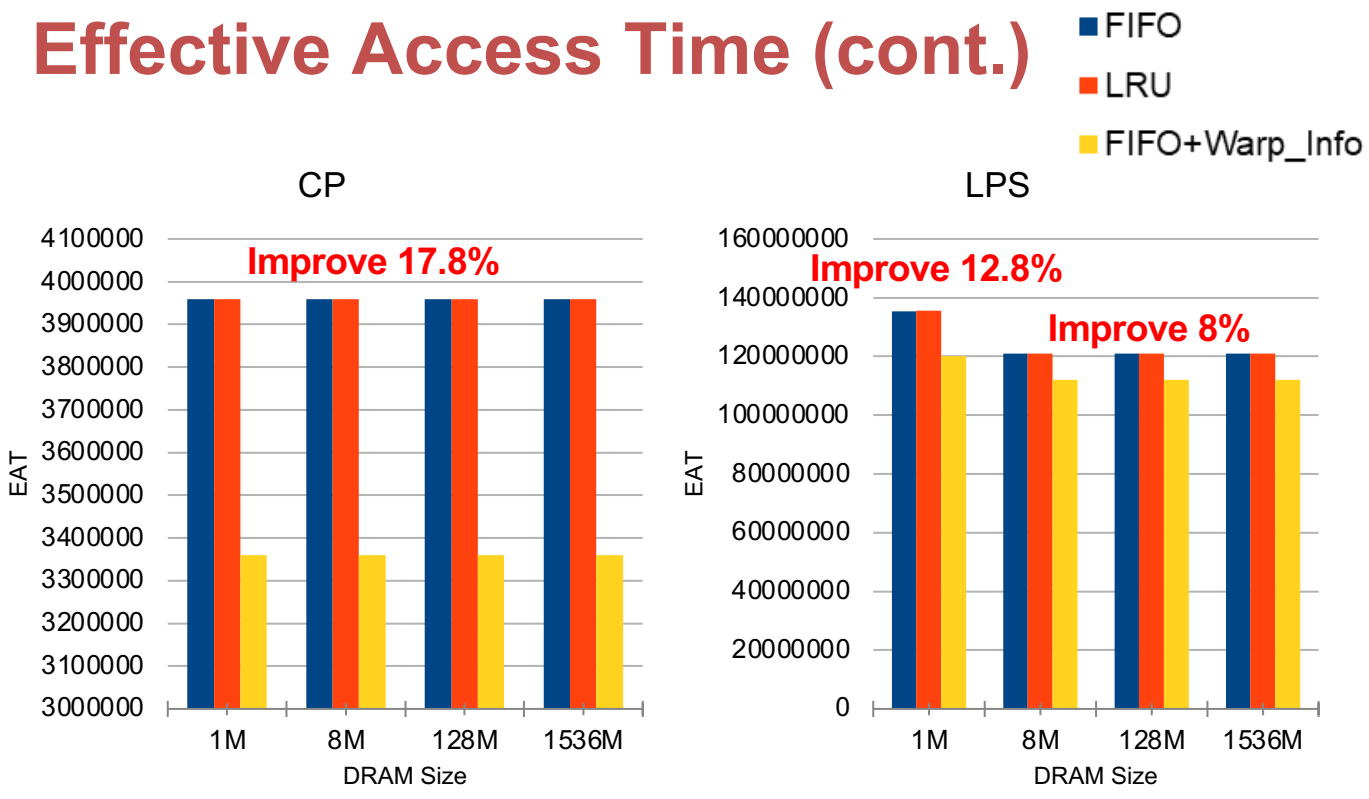


LRU algorithm is not suitable to manage the GPU Memory

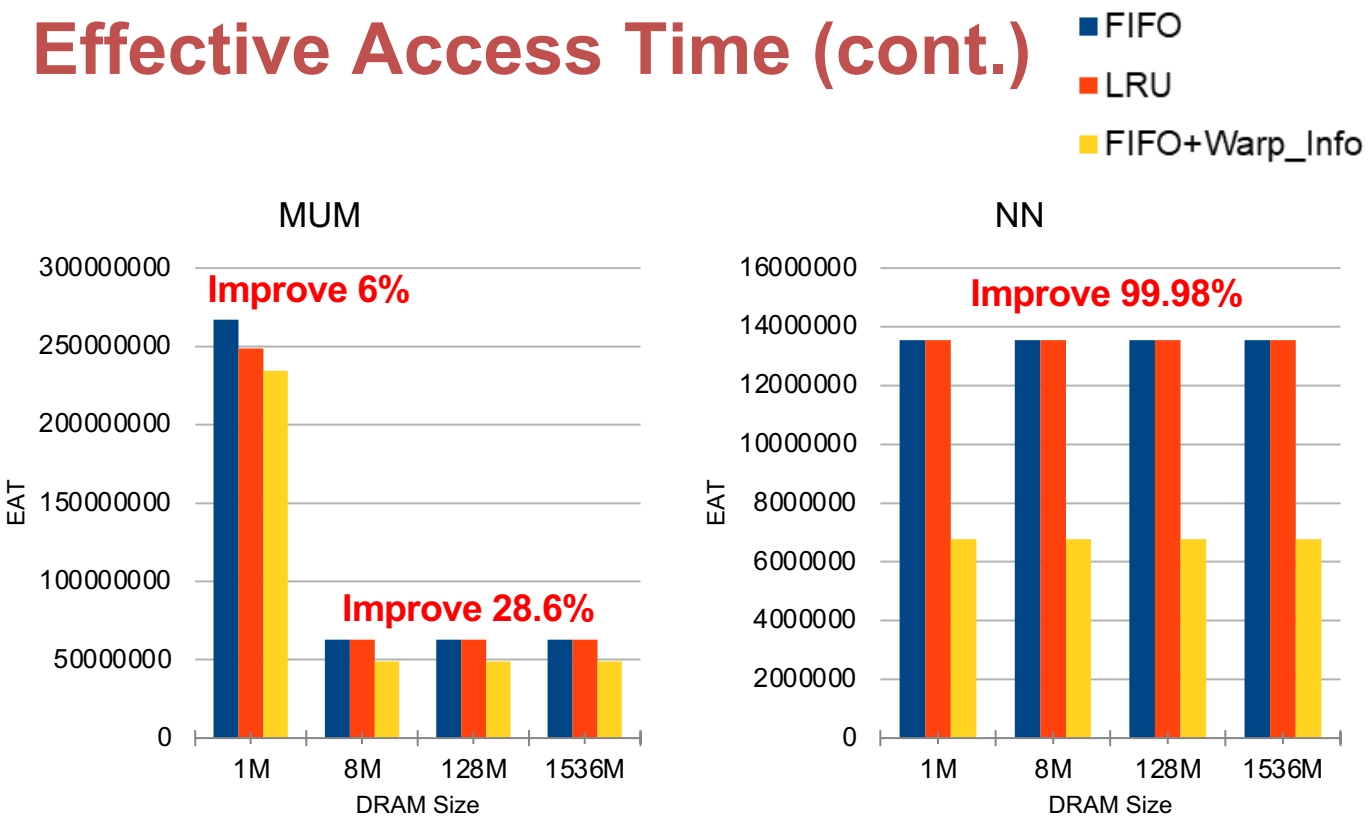
Effective Access Time



Effective Access Time (cont.)



Effective Access Time (cont.)



Outline

- Background and Motivation
- Scheduling-Aware Prefetching
- Performance Evaluation
- Conclusion

Conclusion

- We proposed a scheduling-aware prefetching design in Unified Memory for GPU Device to exploit the process information provided by the warp scheduler so as to predict memory access patterns and perform accurate data prefetching
- The experiment results show that scheduling-aware prefetching design can efficiently improve the DRAM cache hit ratio
- The proposed scheduling-aware prefetching design can improve up to 99% of effective access time compared to LRU algorithm.

Thanks for your attention

Q&A

Contact: Tse-Yuan Wang – tseyuan20@iis.sinica.edu.tw