Emerging Topics in Computer Architecture: Programmable Accelerator, Solid-state Drive, and Security

Junghee Lee

Georgia Institute of Technology

Your Speaker

- Education
  - B.S. Seoul National University, 2000
  - M.S. Seoul National University, 2003
  - Ph.D. Georgia Institute of Technology, 2013
- Work Experience
  - Samsung Electronics, 2003-2008
  - Soteria Inc., 2013-present
  - Research scientist, 2013-present
- Publications
  - 10 Journal papers including 4 papers in ACM and IEEE transactions
  - 13 Conference papers, 2 of which were nominated for the best paper award

Georgia Tech

Research Experience

- Electronic system-level design (SoC/embedded system)
  - Electronic system-level model verification methodology
- Hardware-based load balancing (computer architecture)
- Networks-on-Chip (computer architecture)
  - Ring-based on-chip router architecture
  - Control and data packet segregation
- Programmable hardware accelerator (heterogeneous computer architecture)
- Solid-state drives (embedded system)
  - Preemptive garbage collection
  - Write cache design for an array of solid-state drives
- Hardware-assisted security (security)

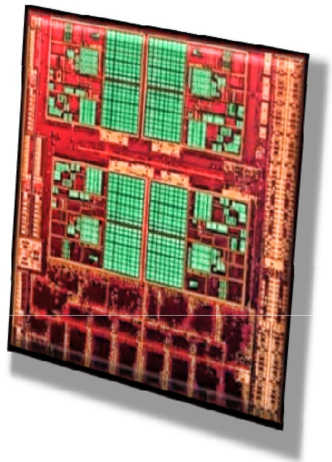
Programmable Accelerator

- Introduction
- Execution Model
- Hardware Architecture
- Evaluation
- Conclusion

Introduction

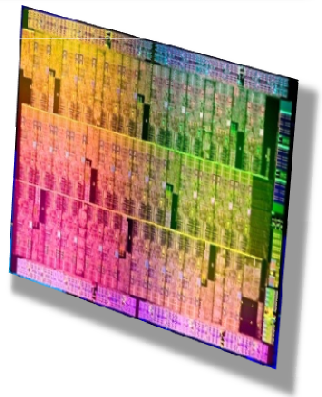
Massively Parallel Processing Array

Fusion



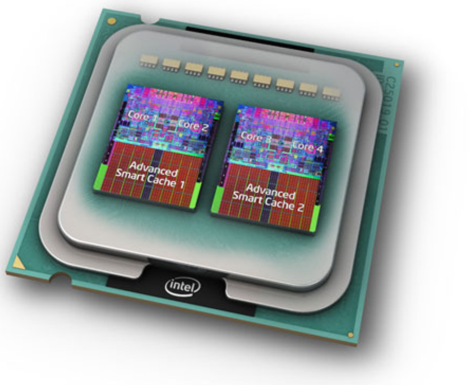
Powerful cores +
H/W accelerator
in a single die
Ex) AMD Fusion

Many Core

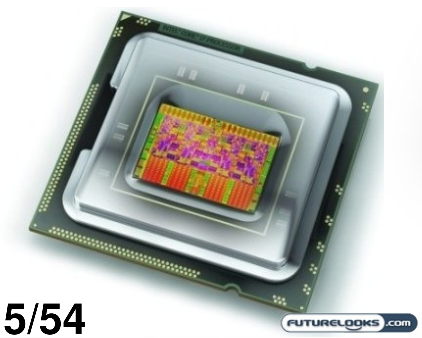


Programmable
Hardware
Accelerator
Ex) GPGPU

Multi Core

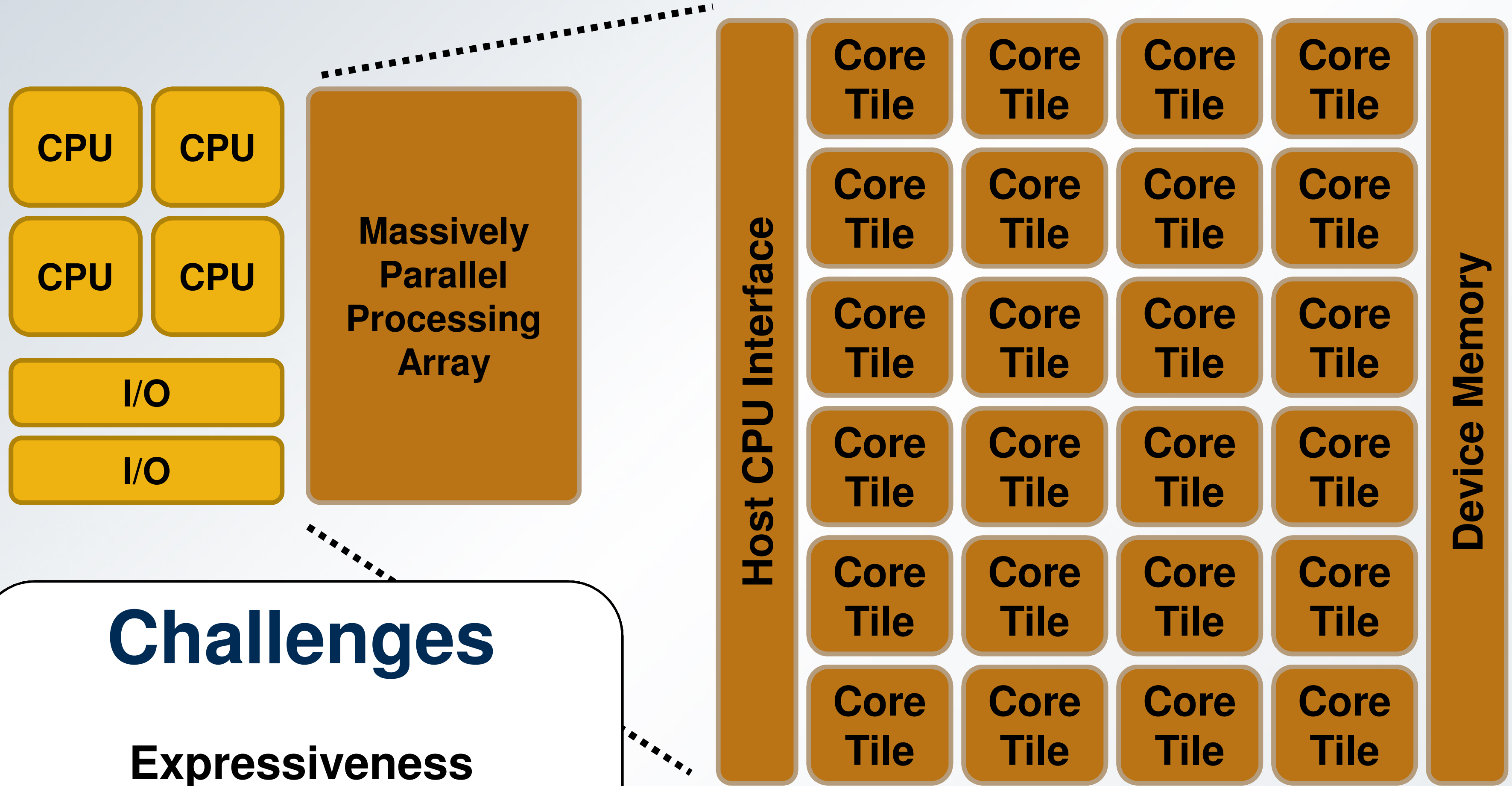


Single Core



Georgia Tech

MPPA as Hardware Accelerator



Challenges

Expressiveness
Debugging
Memory Hierarchy Design

Related Works

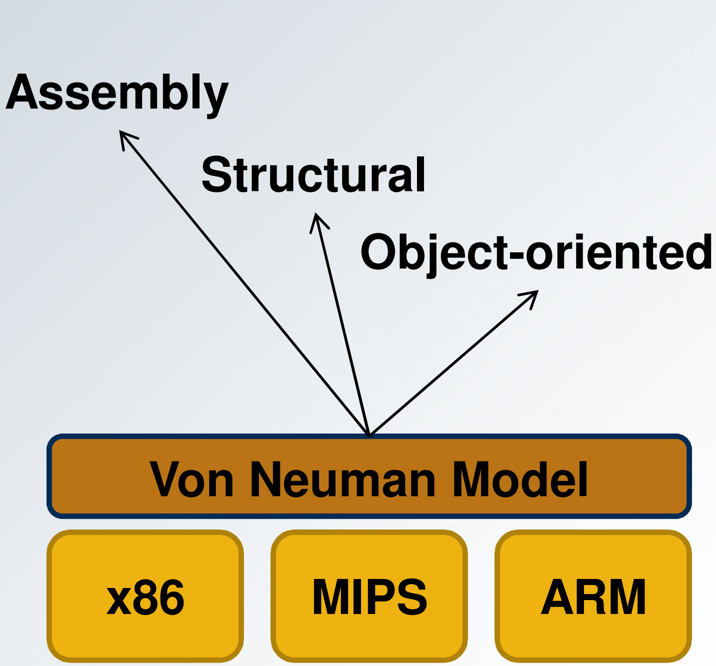
	Expressiveness	Debugging	Memory
GPGPU AMD Fusion	SIMD	Multiple debuggers Event graph	Scratch-pad memory Cache
Tilera	Multi-threading	Multiple debuggers	Coherent cache
Rigel	Multi-threading	Not addressed	Software-managed cache
Ambric	Kahn process network	Formal model	Scratch-pad memory
Proposed MPPA	Event-driven model	Inter-module debug Intra-module debug	Scratch-pad memory Prefetching

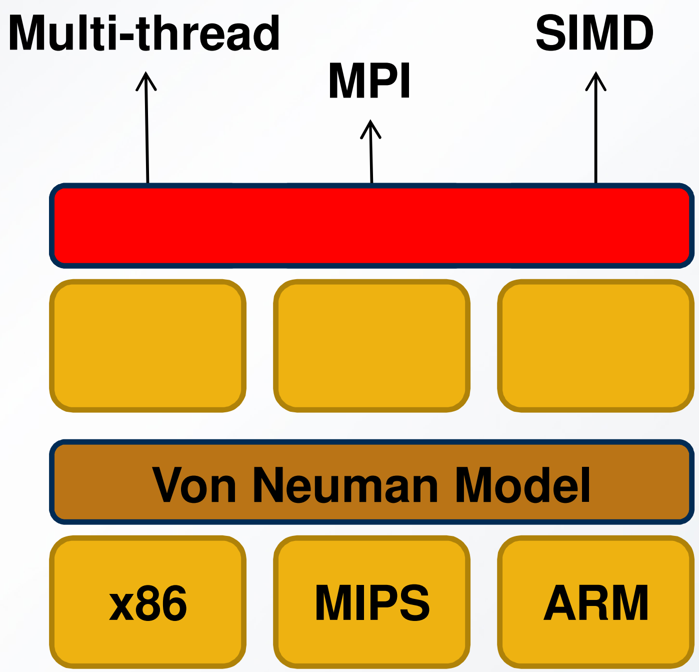
Georgia Tech

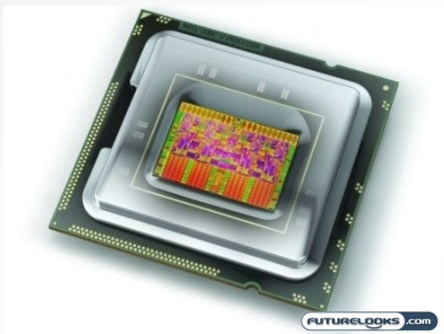
Programmable Accelerator

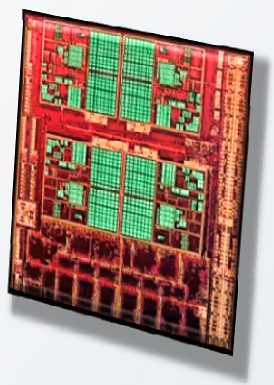
- Introduction
- Execution Model
- Hardware Architecture
- Evaluation
- Conclusion

Execution Model









Requirements

- Decoupling
  - The execution model should decouple the programming model and the execution model of the parallel hardware
- Hardware perspective
  - Low implementation overhead
  - Heterogeneity
  - Scalability
- Software perspective
  - Easy to program
  - Easy to debug
  - Performance

Event-driven Execution Model

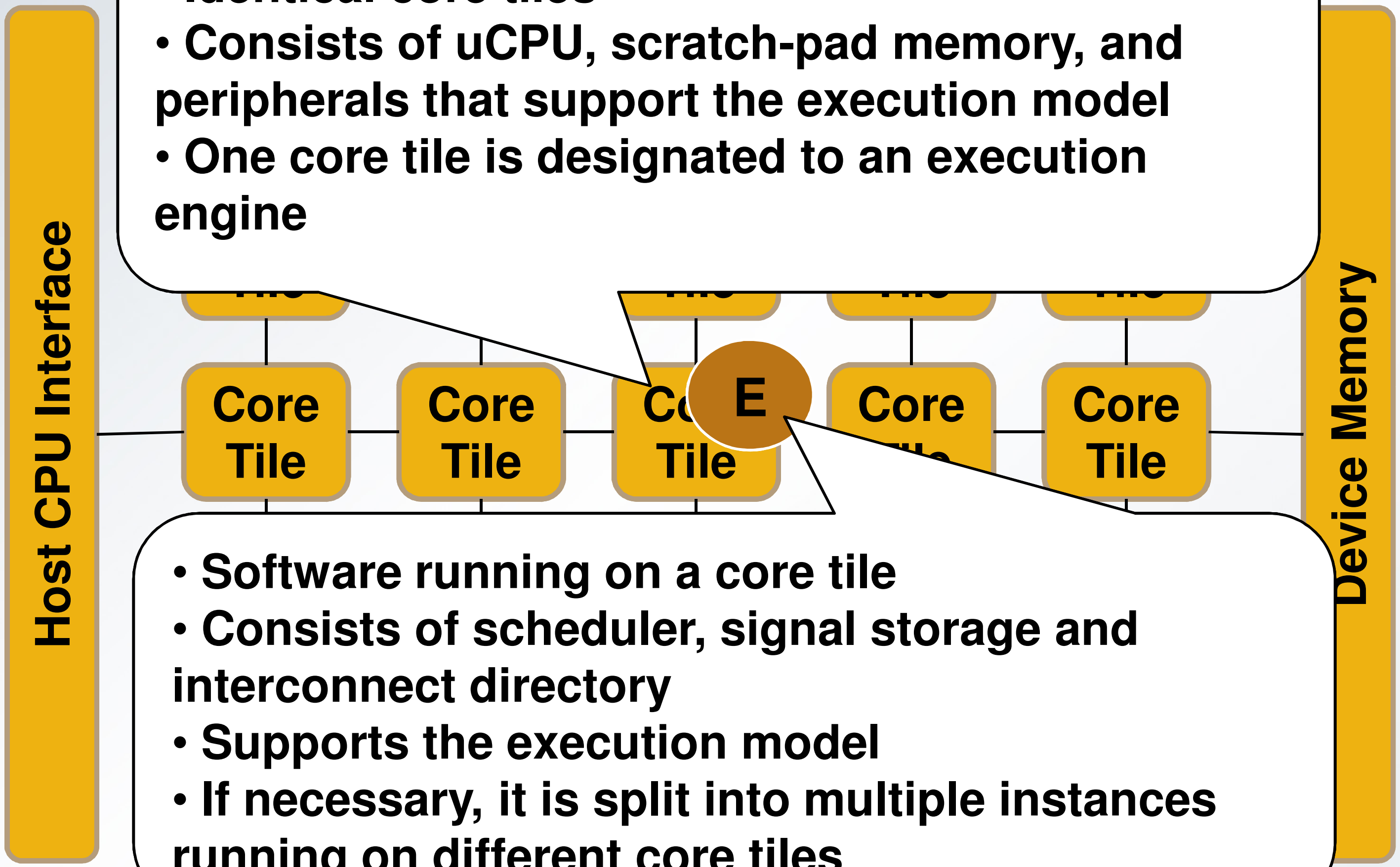
- Specification
  - Module = (b, P_i, P_o, C, F)
    - b = Behavior of module
    - P_i = Input ports
    - P_o = Output ports
    - C = Sensitivity list
  - Signal
  - Net = (d, K)
    - d = Driver port
    - K = A set of sink ports
- Semantics
  - A module is triggered when any signal connected to C changes
  - Function calls and memory accesses are limited to within a module
  - Non-blocking write and block read
  - The specification can be modified during run-time

Georgia Tech

Programmable Accelerator

- Introduction
- Execution Model
- Hardware Architecture
- Evaluation
- Conclusion

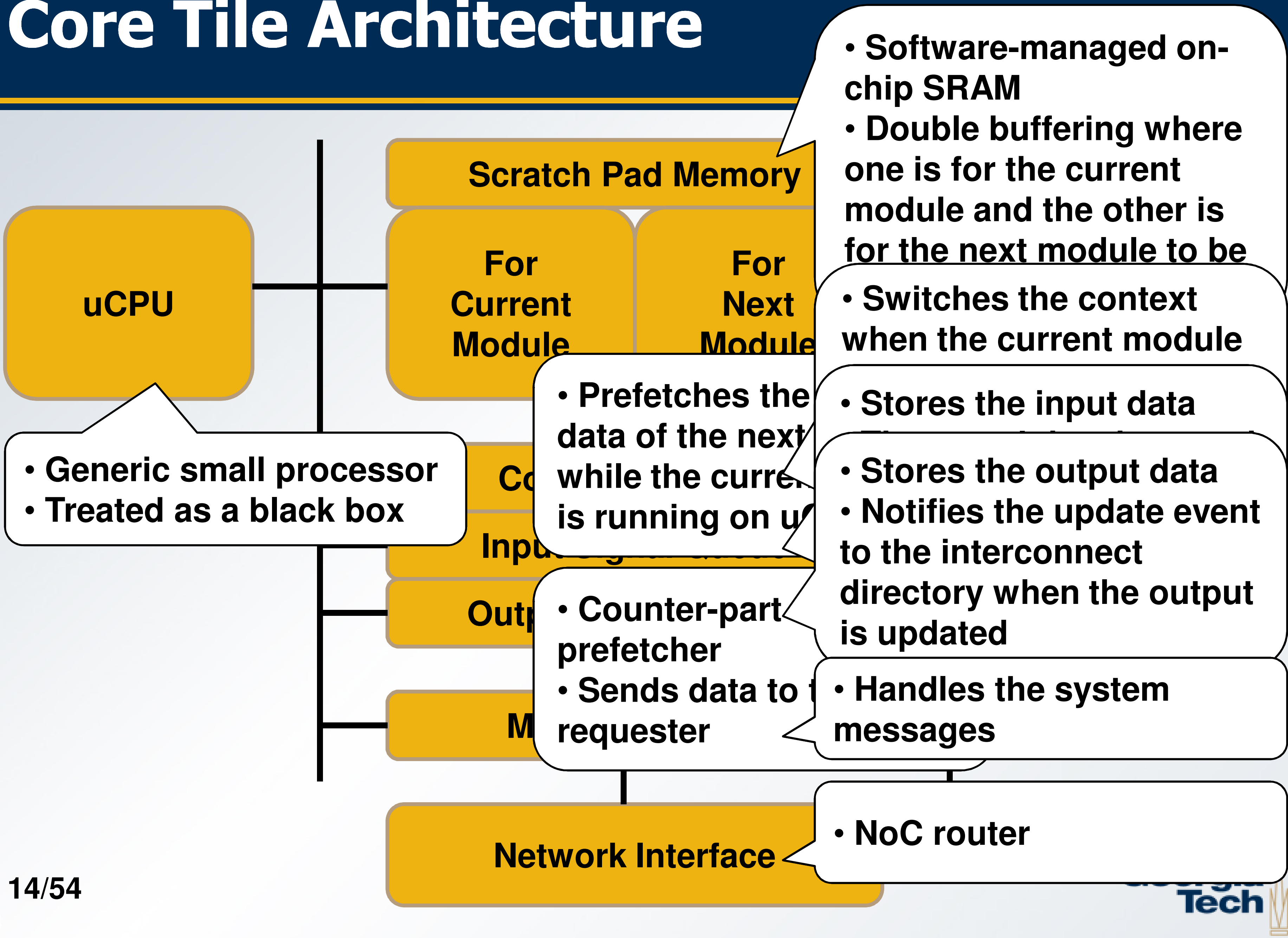
MPPA Microarhitecture



- Identical core tiles
- Consists of uCPU, scratch-pad memory, and peripherals that support the execution model
- One core tile is designated to an execution engine

- Software running on a core tile
- Consists of scheduler, signal storage and interconnect directory
- Supports the execution model
- If necessary, it is split into multiple instances running on different core tiles

Core Tile Architecture



Execution Engine

- Most of its functionality is implemented in software while the hardware facilitates communication
→ Software implementation gives us flexibility in the number and location of the execution engine
- One way to visualize our MPPA is to regard the execution engine as an event-driven simulation kernel
- The execution engine interacts with modules running on other core tiles through messages

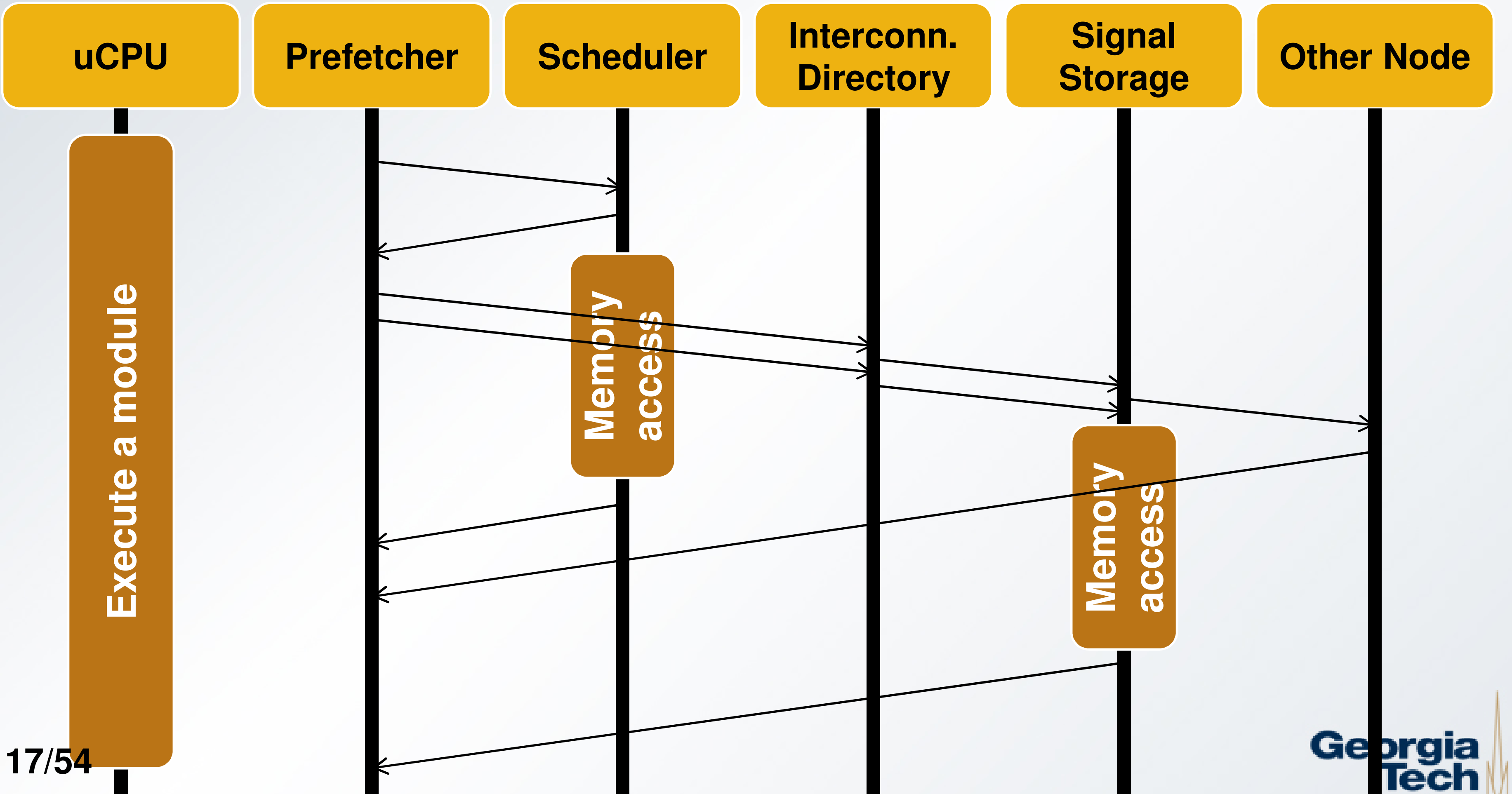
Type	From	To	Payload
REQ_FETCH_MODULE	Prefetcher	Scheduler	Request a new module
RES_FETCH_MODULE	Scheduler	Prefetcher	Module ID and list of input ports
MODULE_INSTANCE	Scheduler	Prefetcher	Code of the module
REQ_SIGNAL	Prefetcher	Interconnect	Port ID
RES_SIGNAL	Signal storage or a node	Prefetcher	Data

Components of Execution Engine

- Scheduler
  - Keeps track of the status and location of modules
  - Maintains three queues: wait, ready and run queue
- Signal storage
  - Stores signal values in the device memory
  - If a signal is updated but its value is still stored in the node, the signal storage invalidates its value and keeps the location of the latest value
- Interconnect directory
  - Keeps track of connectivity of signals and ports
  - Maintains the sensitivity list

Module-Level Prefetching

- Hides the overhead of the dynamic scheduling
- Prefetches the next module while the current module is running



Georgia Tech

Programmable Accelerator

- Introduction
- Execution Model
- Hardware Architecture
- Evaluation
- Conclusion

Benchmark

- Recognition, Synthesis and Mining (RMS) benchmark
- Fine-grained parallelism: dominated by short tasks
  - Small memory foot print
  - High run-time scheduling overhead
- Task-level parallelism: exhibits dependency
  - Hard to be implemented with GPGPU

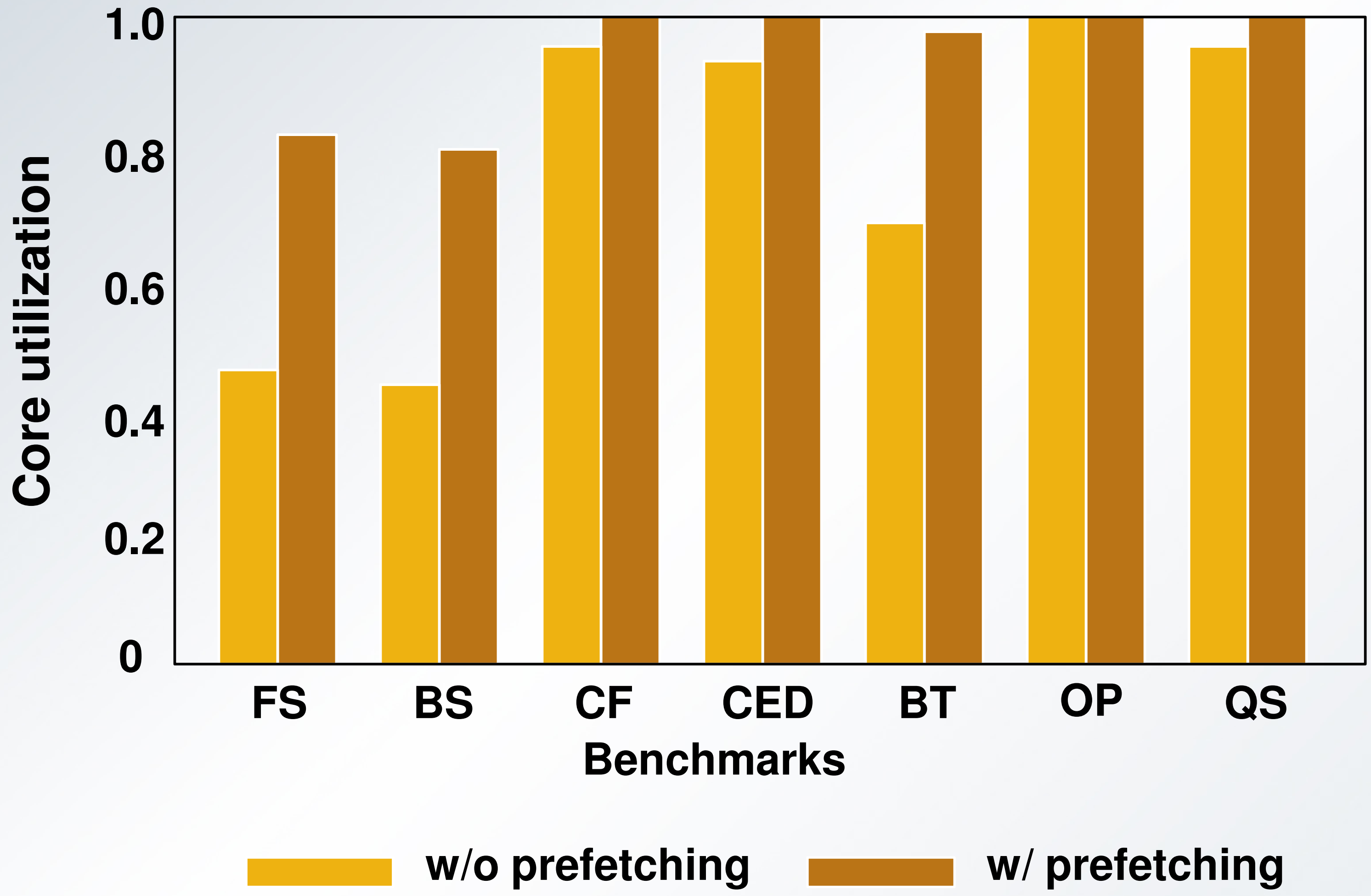
Benchmark	Min	Max	Average
Forward Solve (FS)	26	646	336.00
Backward Solve (BS)	42	569	305.50
Cholesky Factorization (CF)	151	11800	789.35
Canny Edge Detection (CED)	330	5011	669.68
Binomial Tree (BT)	117	4506	462.71
Octree Partitioning (OP)	1441	6679	2678.70
Quick Sort (QS)	88	47027	683.70

Simulator

- In-house cycle-level simulator
- Parameters

Parameter	Value
Number of core tiles	32
Memory access time	1 cycle for scratch-pad memory 100 cycles for device memory
Memory size	8 KB scratch-pad memory 32 MB device memory
Communication Delay	4 cycles per hop

Utilization



Conclusion

- A novel MPPA architecture is proposed that employs an event-driven execution model
  - Handles dependencies by dynamic scheduling
  - Hides dynamic scheduling overhead by module-level prefetching
- Future works
  - Supports applications that require larger memory footprint
  - Adjusts the number of execution engines dynamically
  - Supports inter-module debugging

Solid-state Drive

- Introduction
- Background and Motivation
- Semi-Preemptive Garbage Collection
- Evaluation
- Conclusion

High Performance Storage Systems

- Server centric services
  - File, web & media servers, transaction processing servers
- Enterprise-scale Storage Systems
  - Information technology focusing on storage, protection, retrieval of data in large-scale environments

High Performance Storage Systems

Storage Unit
Hard Disk Drive

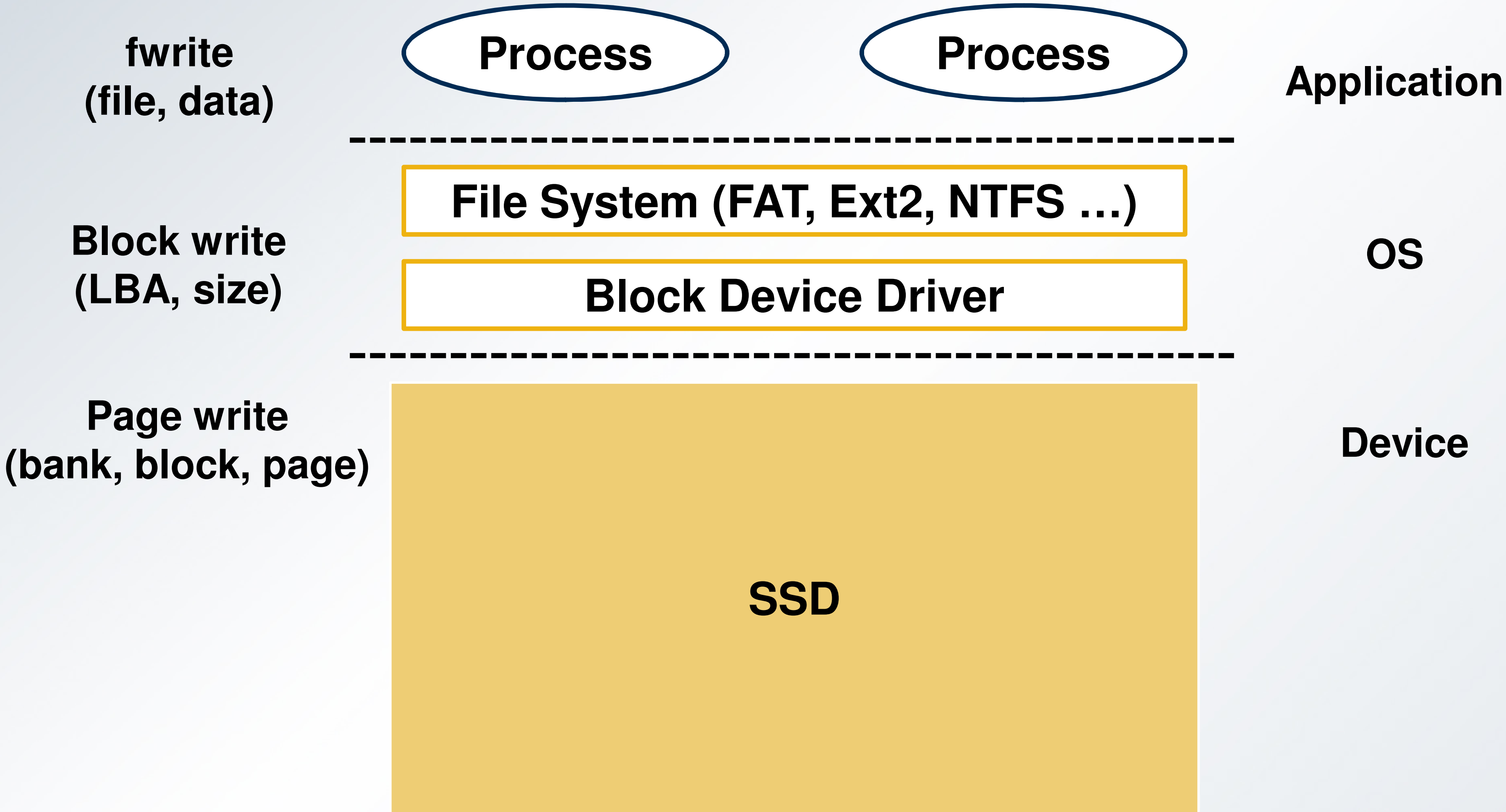
Emergence of NAND Flash based SSD

- NAND Flash vs. Hard Disk Drives
  - Pros:
    - Semi-conductor technology, no mechanical parts
    - Offer lower access latencies
      - *μs* for SSDs vs. *ms* for HDDs
    - Lower power consumption
    - Higher robustness to vibrations and temperature
  - Cons:
    - Limited lifetime
      - 10K - 1M erases per block
    - High cost
      - About 8X more expensive than current hard disks
    - ***Performance variability***

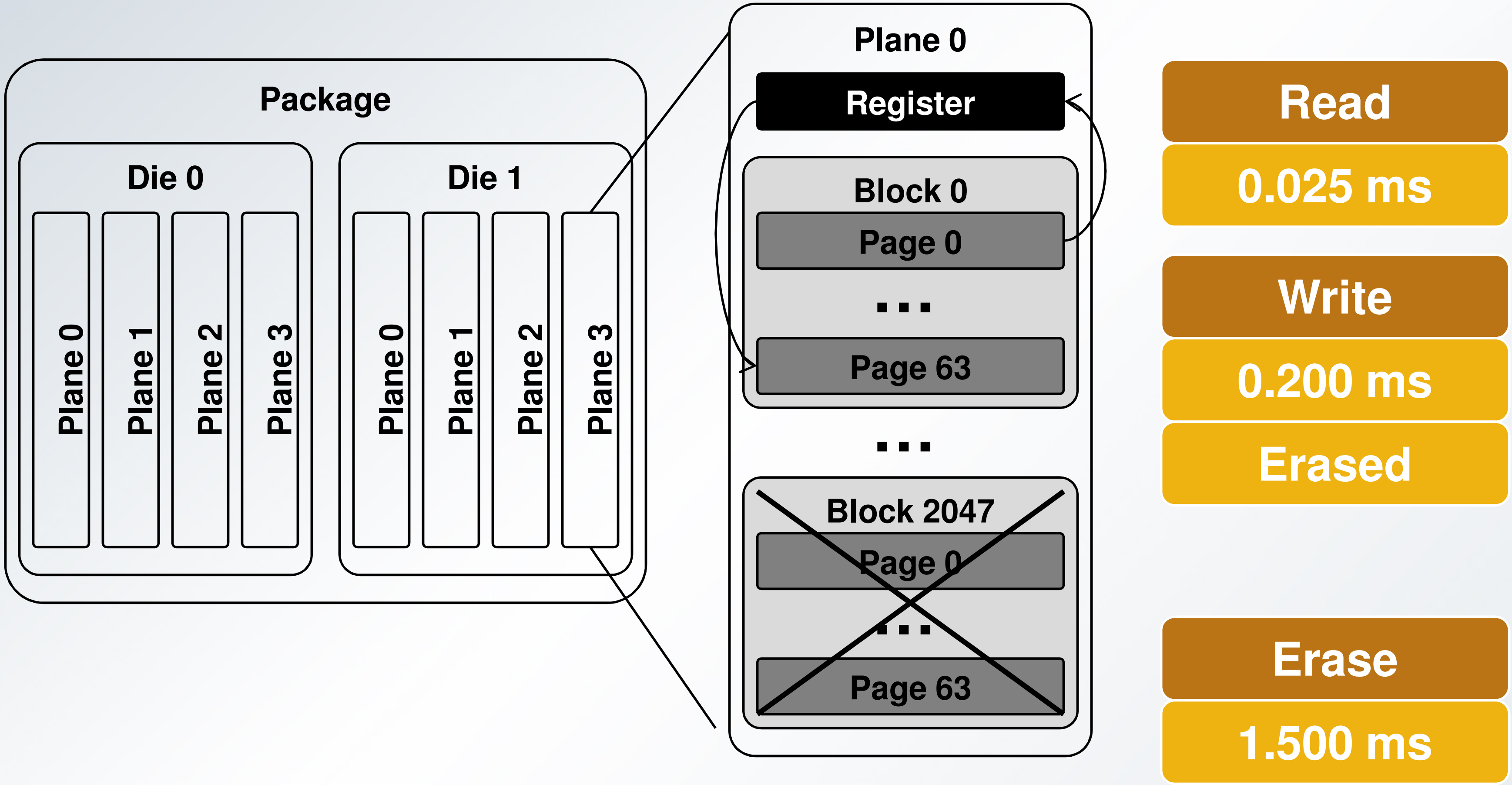
Solid-state Drive

- Introduction
- Background and Motivation
- Semi-Preemptive Garbage Collection
- Evaluation
- Conclusion

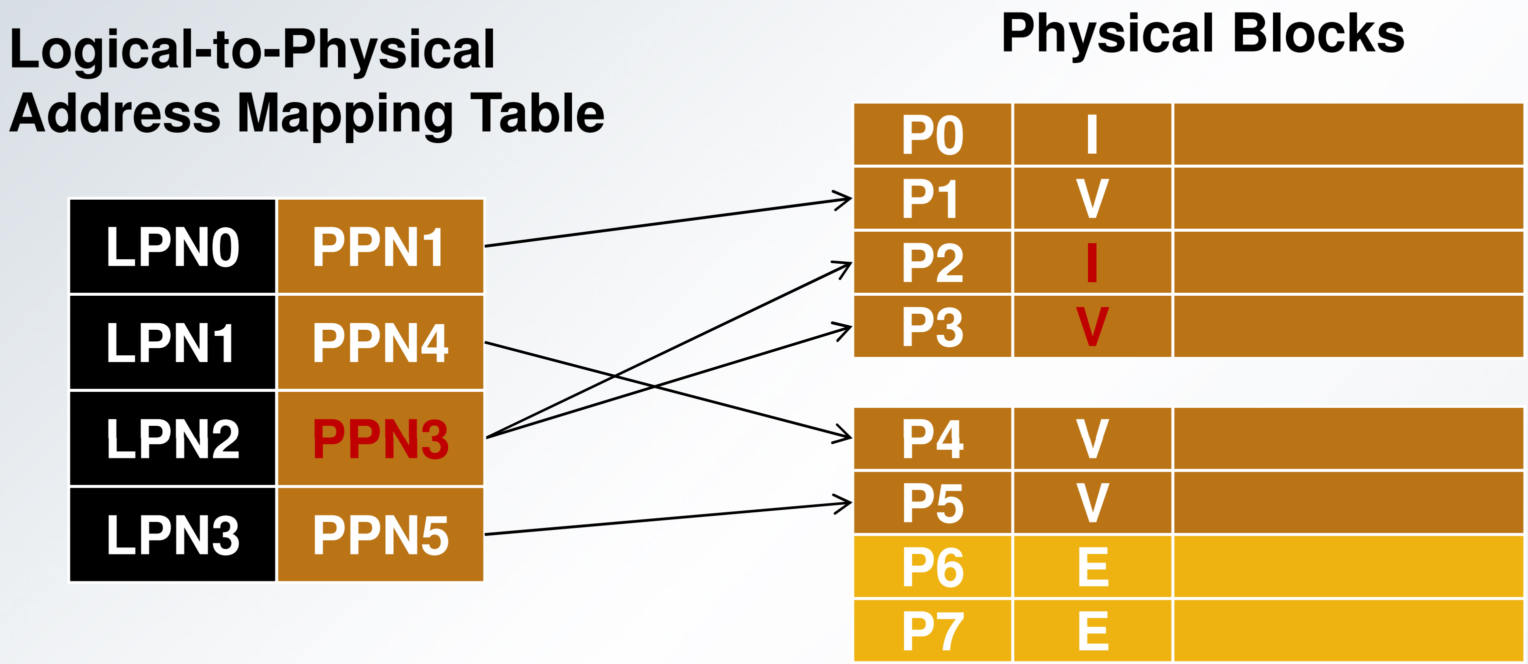
NAND Flash based SSD



NAND Flash Organization



Out-Of-Place Write



Write to LPN2

Invalidate PPN2

Write to PPN3

Update table

Garbage Collection

Select Victim Block

Move Valid Pages

Erase Victim Block

Physical Blocks

P0	E	
P1	E	
P2	E	
P3	E	
P4	V	
P5	V	
P6	V	
P7	V	

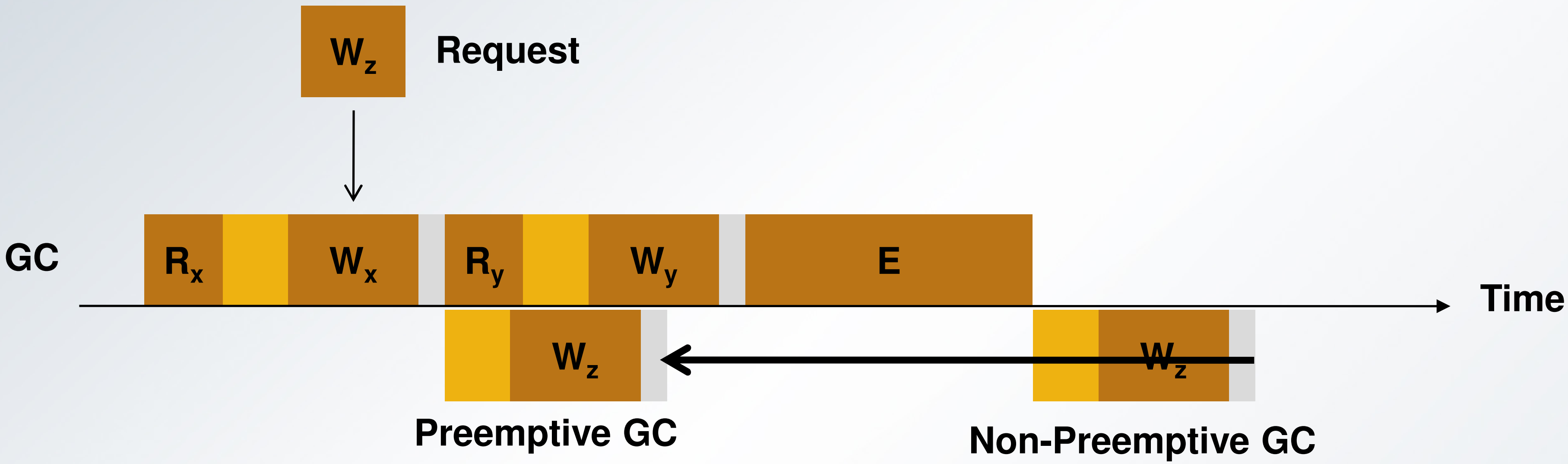
2 reads + 2 writes + 1 erase= 2*0.025 + 2*0.200 + 1.5 = 1.950(ms) !!

Georgia Tech

Solid-state Drive

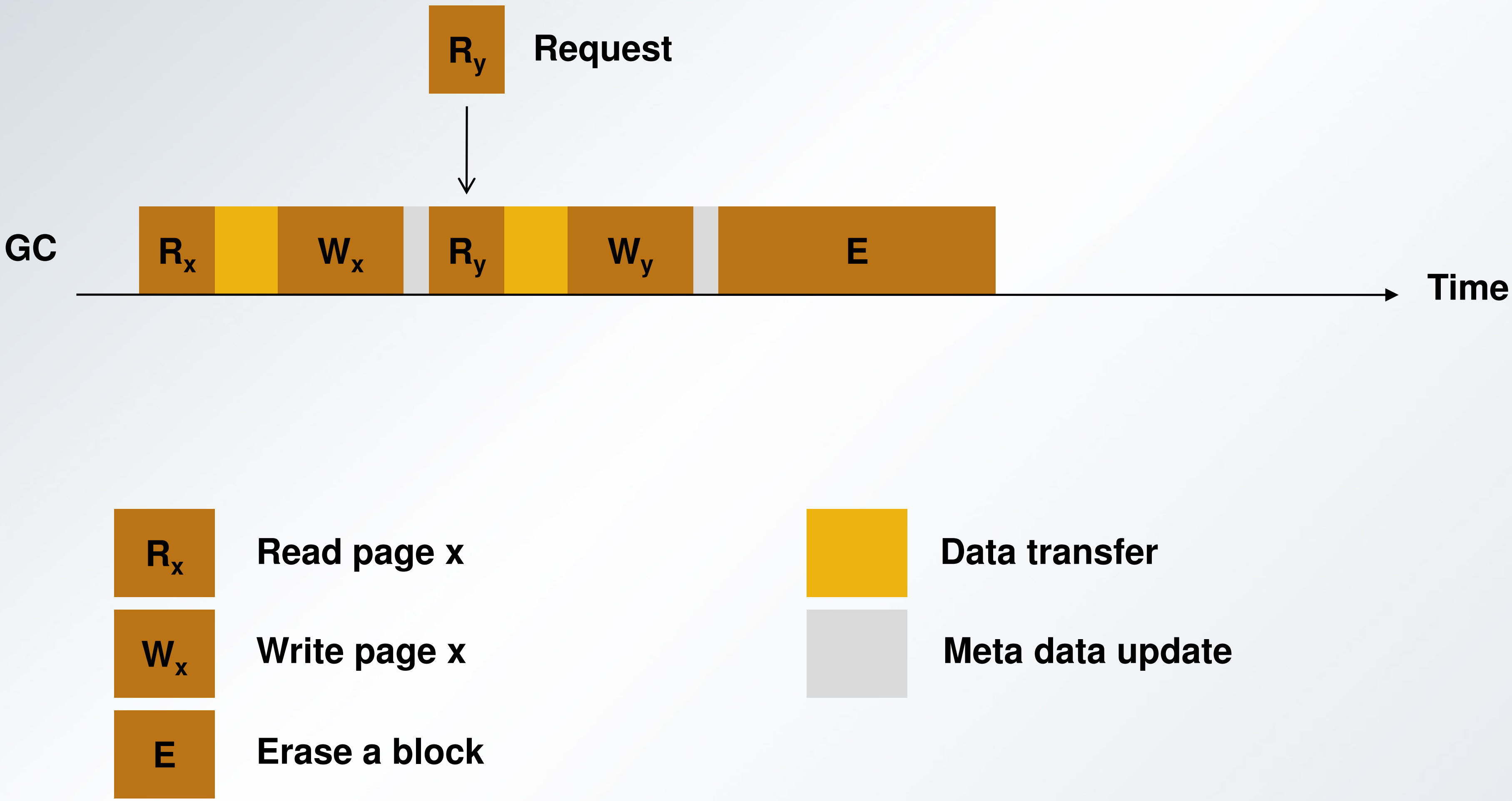
- Introduction
- Background and Motivation
- Semi-Preemptive Garbage Collection
- Evaluation
- Conclusion

Technique #1: Semi-Preemption

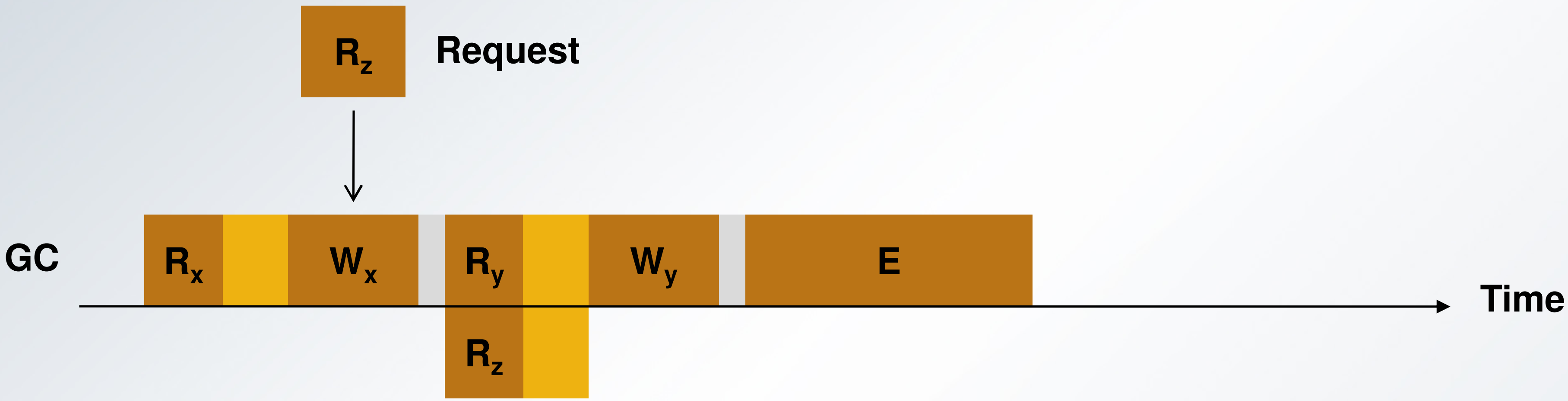


R_x	Read page x
W_x	Write page x
E	Erase a block
(yellow)	Data transfer
(gray)	Meta data update

Technique #2: Merge



Technique #3: Pipeline



- R_x — Read page x
- W_x — Write page x
- E — Erase a block
- Data transfer
- Meta data update

Level of Allowed Preemption

- Drawback of PGC
  : The completion time of GC is delayed
  → May incur lack of free blocks
  → Sometimes need to prohibit preemption
- States of PGC

	Garbage collection	Read requests	Write requests
State 0	X		
State 1	O	O	O
State 2	O	O	X
State 3	O	X	X

Solid-state Drive

- Introduction
- Background and Motivation
- Semi-Preemptive Garbage Collection
- Evaluation
- Conclusion

Setup

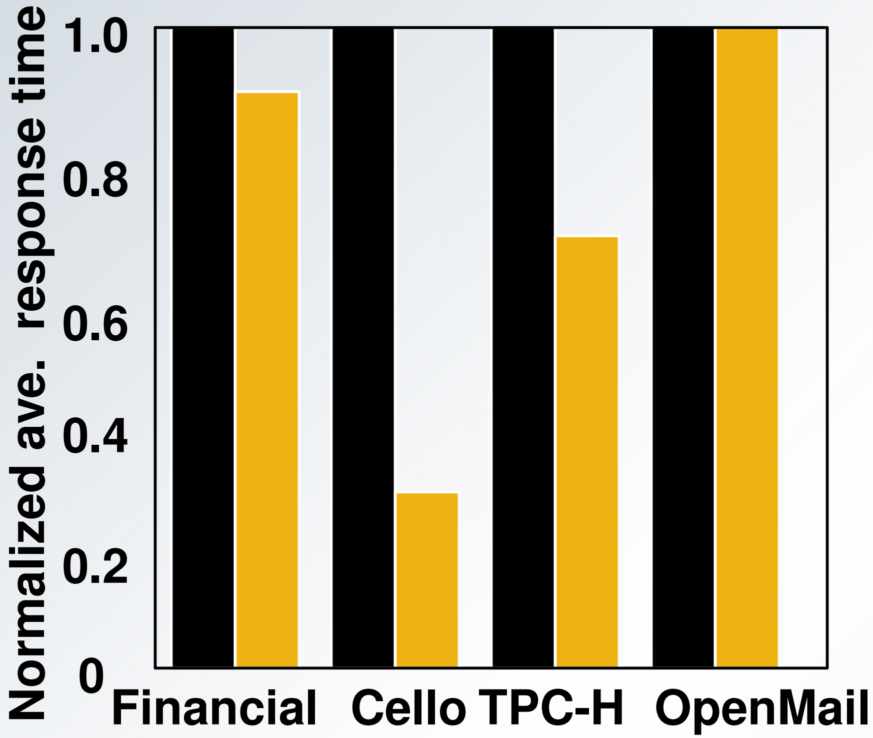
- Simulator
  - MSR's SSD simulator based on DiskSim
- Workloads

	Workloads	Average request size (KB)	Read ratio (%)	Arrival rate (IOP/s)
Write dominant	Financial	7.09	18.92	47.19
	Cello	7.06	19.63	74.24
Read dominant	TPC-H	31.62	91.80	172.73
	OpenMail	9.49	63.30	846.62

Georgia Tech

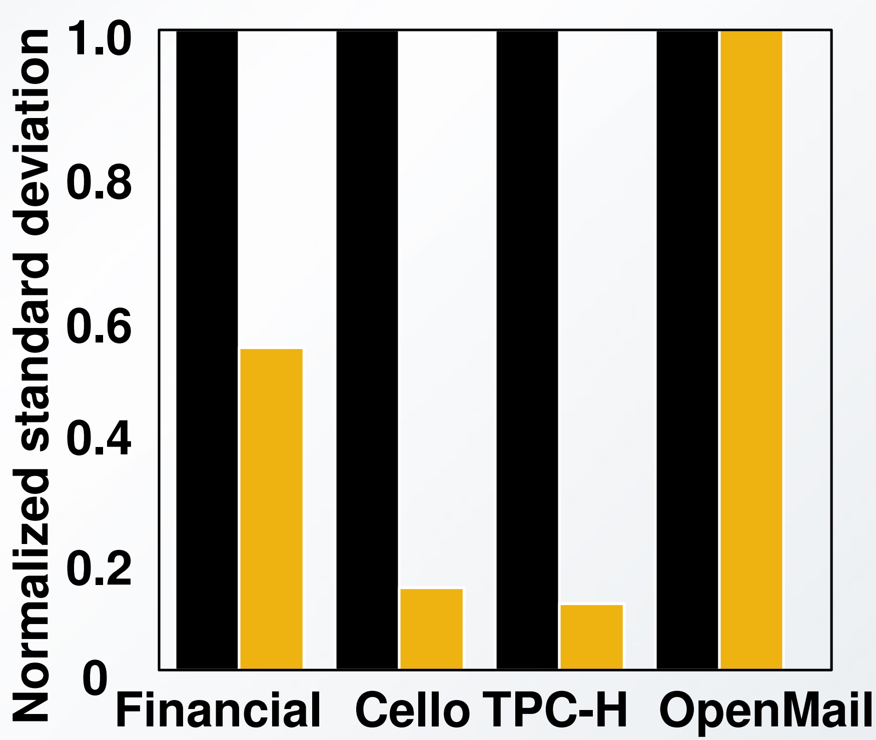
Performance Improvement for Realistic Workloads

- Average Response Time



Improvement of average response time by 6.5% and 66.6% for Financial and Cello.

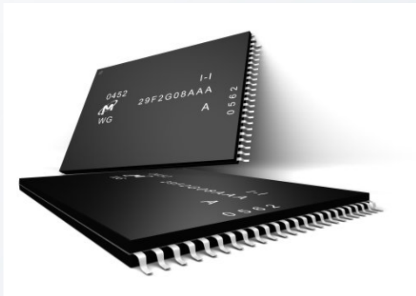
- Variance of Response Times



Improvement of variance of response time by 49.8% and 83.3% for Financial and Cello.

Conclusions

- Solid state drives
  - Fast access speed
  - Performance variation ← garbage collection
- Semi-preemptive garbage collection
  - Service incoming requests during GC
- Average response time and performance variation are reduced by up to 66.6% and 83.3%



Security

- Introduction
- Append-only Storage
- Use Cases
- Conclusion

Evolutionary Digital Systems Advance

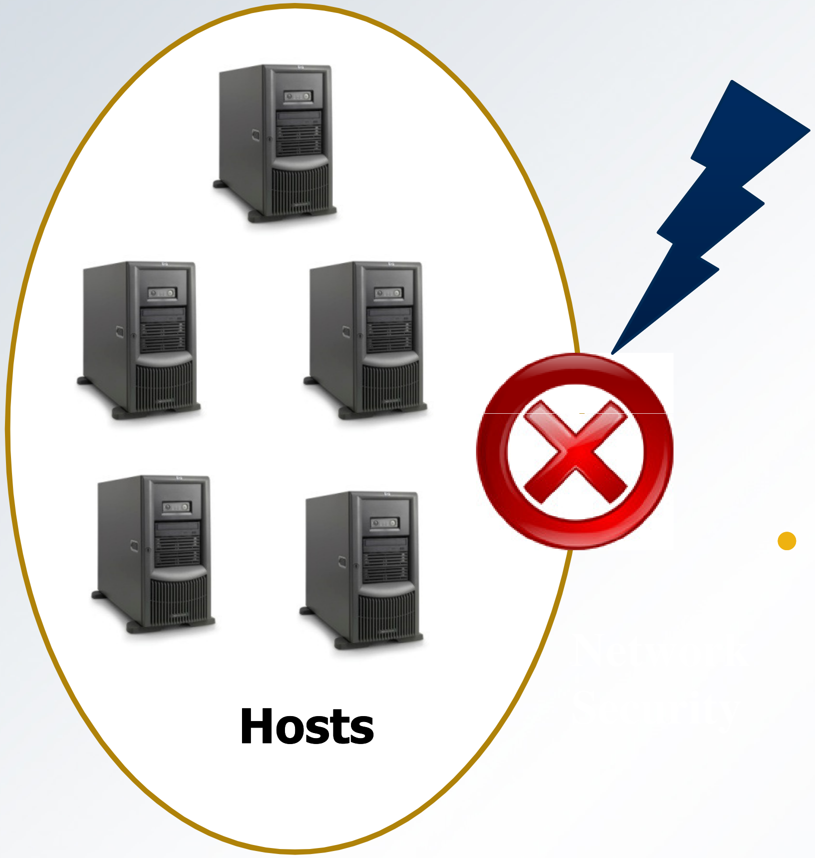
- IoT (Internet of Things)
  - By 2015, 5 billion individuals will be connected to the Internet (source: GKP)
  - 100 billion uniquely identifiable objects will be connected to the Internet by 2020
- Big Data Visualization
  - Digital data is doubling every other year
- Cloud Computing and Mobile Computing
- Cybersecurity
  - New business models based on innovative thinking will be needed

Georgia Tech

Financial Impact

- Computer crimes cost firms who detect and verify incidents between $145 million and $730 million each year (NCSA Annual Worry Report)
- A company that experiences a computer outage lasting more than 10 days will never fully recover financially. 50 percent will be out of business within five years ("Disaster Recovery Planning: Managing Risk & Catastrophe in Information Systems" by Jon Toigo)
- 43% of lost or stolen data is valued at $5 million or more
- 43% of companies experiencing data disasters never reopen, and 29 percent close within two years (McGladrey and Pullen)
- It is estimated that 1 out of 500 data centers will have a severe disaster each year (McGladrey and Pullen)

Scope



- Network Security
  - Efficient
    - It can protect numerous hosts by securing only the perimeter
  - But not perfect
    - Although data centers are equipped with various network security techniques, it is estimated 1 out of 500 data centers will have a severe disaster each year (McGladrey and Pullen)
  - The ultimate goal is protecting hosts
- Host Security
  - Protect hosts directly
  - Compatibility issue
    - Heterogeneity of the hosts (different version and types of OS and different hardware)
  - Performance overhead

Hardware-assisted Security

- Trusted Platform Module (TPM)
  - Key burnt in hardware
- Intel vPro
  - Trusted Execution Technology
    - Virtualization (TrustZone of ARM)
  - Identity Protection Technology
    - One-time password
- Monitoring
  - Copilot, RKRD, KI-Mon
    - Coprocessor-based

Georgia Tech

Security

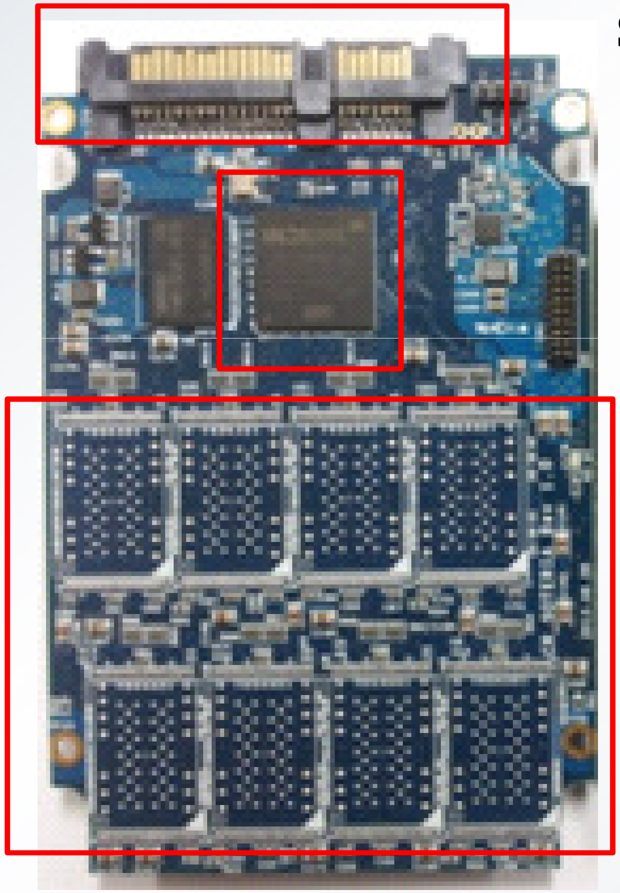
- Introduction
- Append-only Storage
- Use Cases
- Conclusion

Elevator Pitch

Protect reference data from unauthorized modification by using Append-only Storage

- Write-only read many (WORM) devices: CD or DVD

Soteria Security Card (SSC)



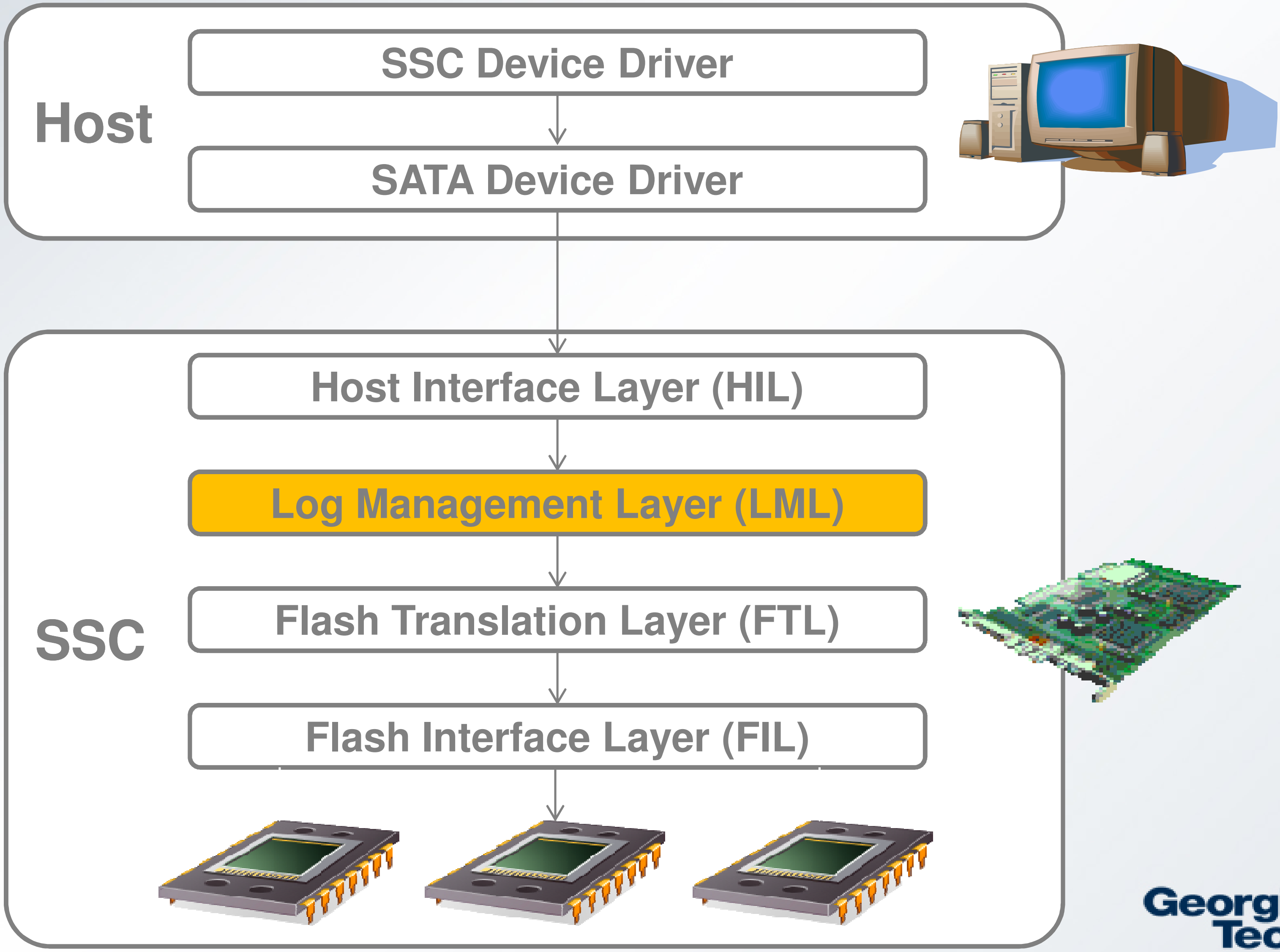
SATA interface

ARM7-based controller

NAND flash memories

Georgia Tech

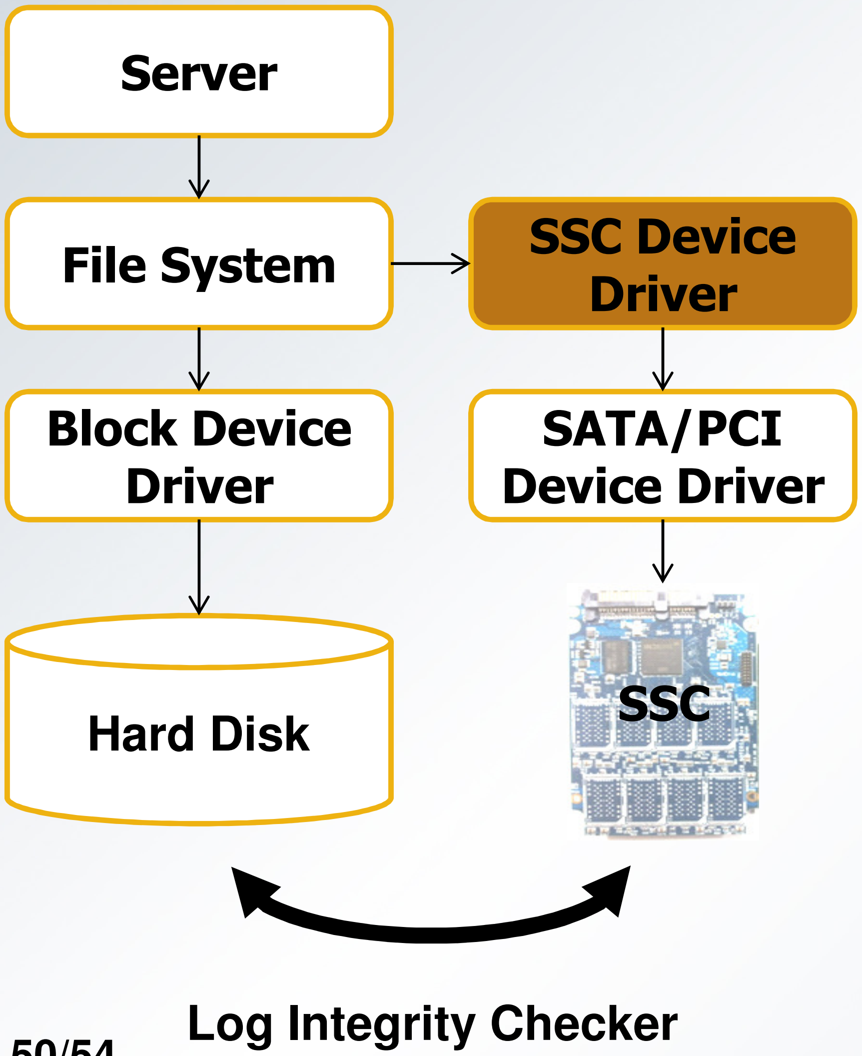
SSC Firmware



Security

- Introduction
- Append-only Storage
- Use Cases
- Conclusion

Use Case #1: Log Protection



- Using SSC
  - Logs are stored in both the hard disk and SSC
  - Log integrity checker checks if the logs are contaminated by comparing those in the hard disk against those in SSC
- Performance
  - Performance degradation of the response time of the Apache web server is 0.88% employing a separate process to store logs

Current Practice

- Log protection techniques
  - Logging server
    - Vulnerabilities involved in collecting and transferring logs
  - Encryption
    - Encryption is secure only if the key is not revealed
    - According to 2012 Verizon Data Breach report, 76% of data breaches exploited weak or stolen credentials
  - Hypervisor
    - Who protects hypervisor itself?
- Does this really happen?
  - According to a police officer in charge of cyber crime investigation,
    - some attackers delete their traces from logs, and
    - some attackers delete everything from the hard disk, which includes logs

Use Case #2: File Integrity Check

- File integrity
  - File modification is usually (if not always) a prerequisite or a result of malware
  - Therefore, file integrity checking is a powerful tool to find out the cause of attacks and malware
- Using SSC
  - The integrity information of files is stored in the hardware
  - By comparing against the stored integrity information, unauthorized modifications can be detected
- Performance
  - Since the file integrity checker is an off-line utility, the performance impact can be minimized by assigning a low priority
  - Malware detectors and integrity checkers detect malicious activities by comparing against some reference data

Conclusion

- Soteria Security Card:
  - Prevents reference data from unauthorized modification
  - Stored data cannot be modified or erased
- Use cases
  - Log protection
  - File integrity checking
  - File access monitoring
  - Non-repudiation
  - Medical record
  - Financial transaction

Thank you!

Appendix

Georgia Institute of Technology®

A Programmable Processing Array Architecture Supporting Dynamic Task Scheduling and Module-Level Prefetching

Junghee Lee[*], Hyung Gyu Lee[*], Soonhoi Ha[†], Jongman Kim[*], and Chrysostomos Nicopoulos[‡]

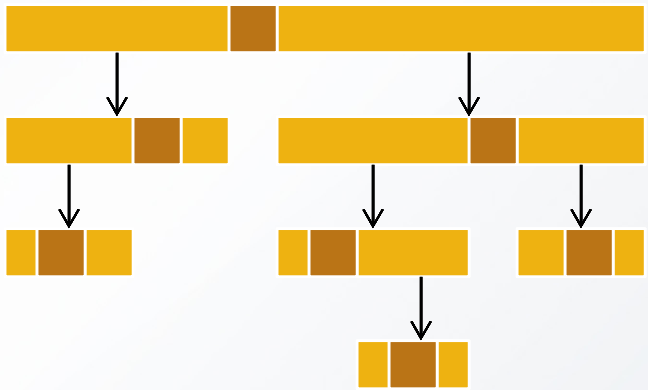
* Georgia Institute of Technology

† VERI TAS LUX MEA

‡ University of Cyprus

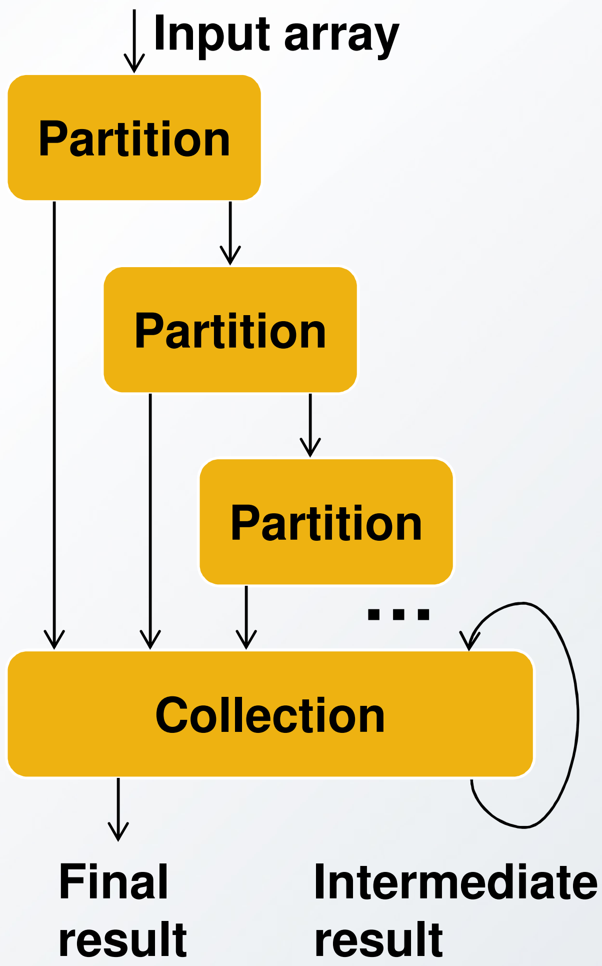
Example

- Quick sort
  - Pivot is selected
  - The given array is partitioned so that
    - The left segment should contain smaller elements than the pivot
    - The right segment should contain larger elements than the pivot
  - Recursively partition the left and right segments
- Specifying quick sort
  - Multi-threading
    - OK but hard to debug
  - SIMD
    - Inefficient due to input dependency
  - Kahn process network
    - Impossible due to the dynamic nature

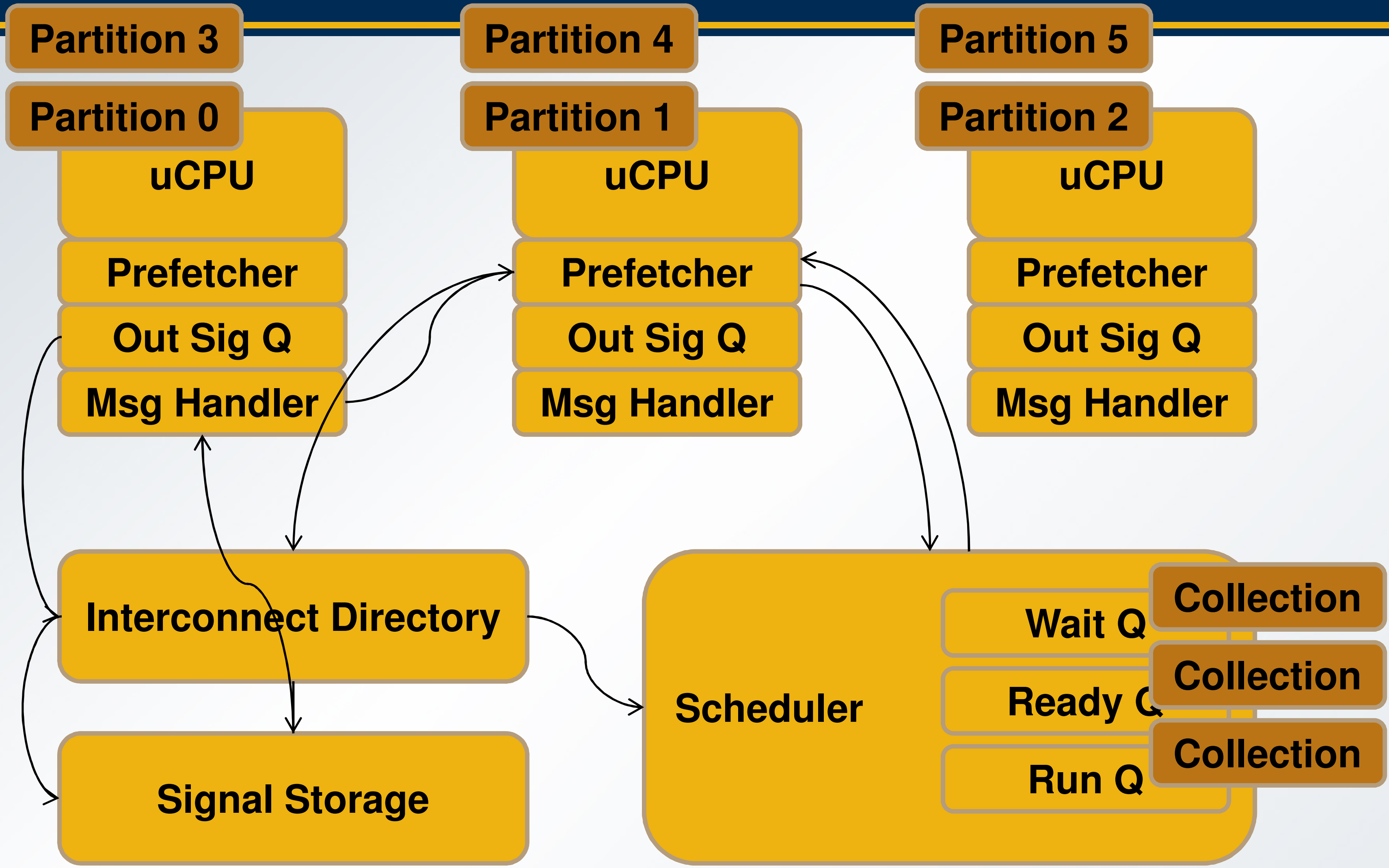


Specify Quick Sort with Event-driven Model

- Partition module
  - *b* (behavior): select a pivot, partition the input array, instantiate another partition module if necessary
  - P_i (input port): input array and its position
  - P_o (output port): left and right segments and their position
  - *C* (sensitivity list): input array
  - *P* (prefetch list): input array
- Collection module
  - *b* (behavior): collect segments
  - P_i (input port): sorted segments and intermediate result
  - P_o (output port): final result and intermediate result
  - *C* (sensitivity list): sorted segments
  - *P* (prefetch list): sorted segments and intermediate result

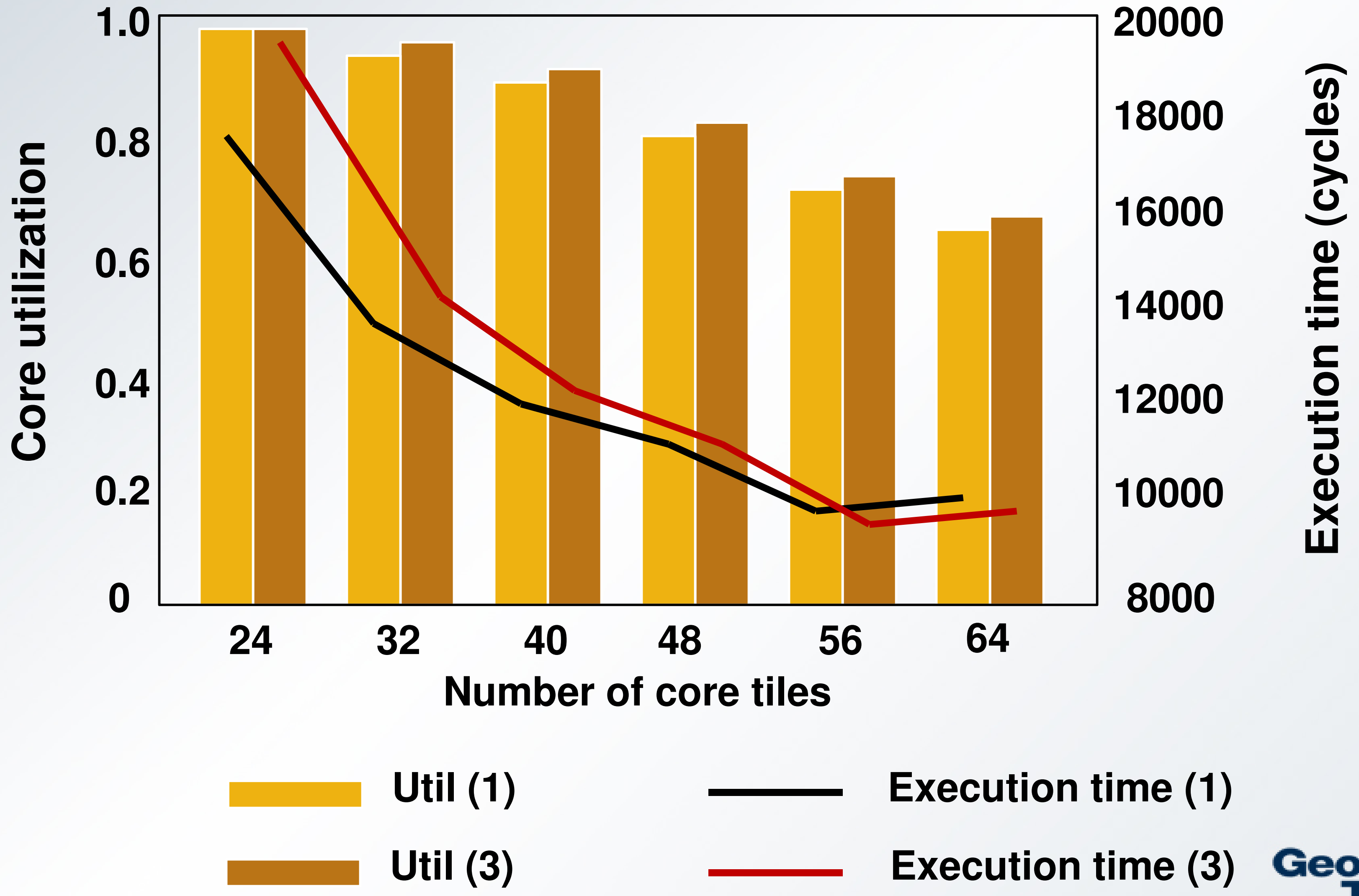


Illustrative Example



Georgia Tech

Scalability



A Semi-Preemptive Garbage Collector for Solid State Drives

Junghee Lee, Youngjae Kim, Galen M. Shipman, Sarp Oral, Feiyi Wang, and Jongman Kim

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Georgia Institute of Technology®

Spider: A Large-scale Storage System

- Jaguar
  - Peta-scale computing machine
  - 25,000 nodes with 250,000 cores and over 300 TB memory
- Spider storage system
  - The largest center-wide Lustre-based file system
  - Over 10.7 PB of RAID 6 formatted capacity
    - 13,400 x 1 TB HDDs
  - 192 Lustre I/O servers
    - Over 3TB of memory (on Lustre I/O servers)

Georgia Tech

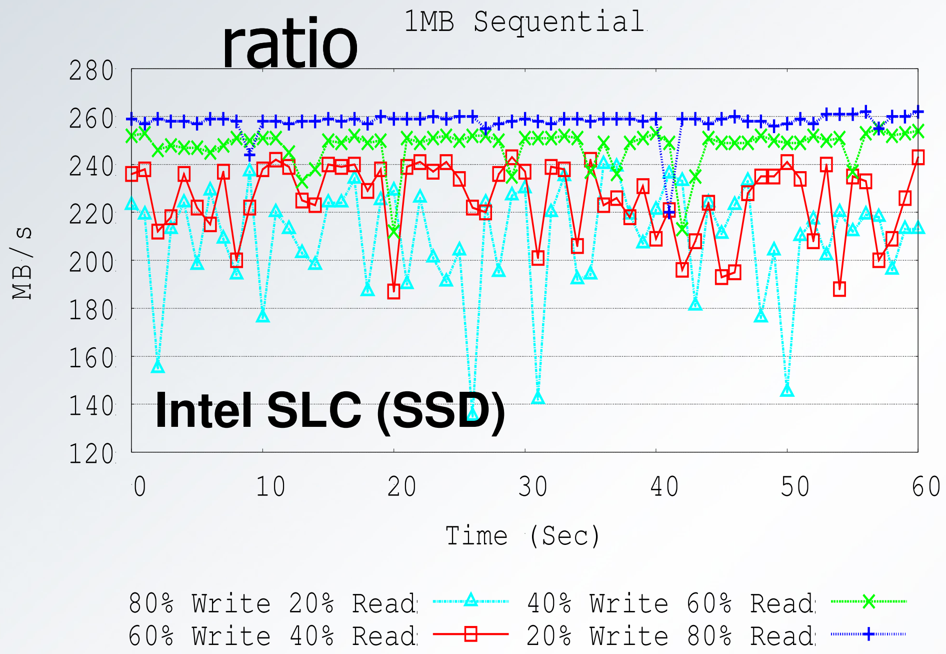
Pathological Behavior of SSDs

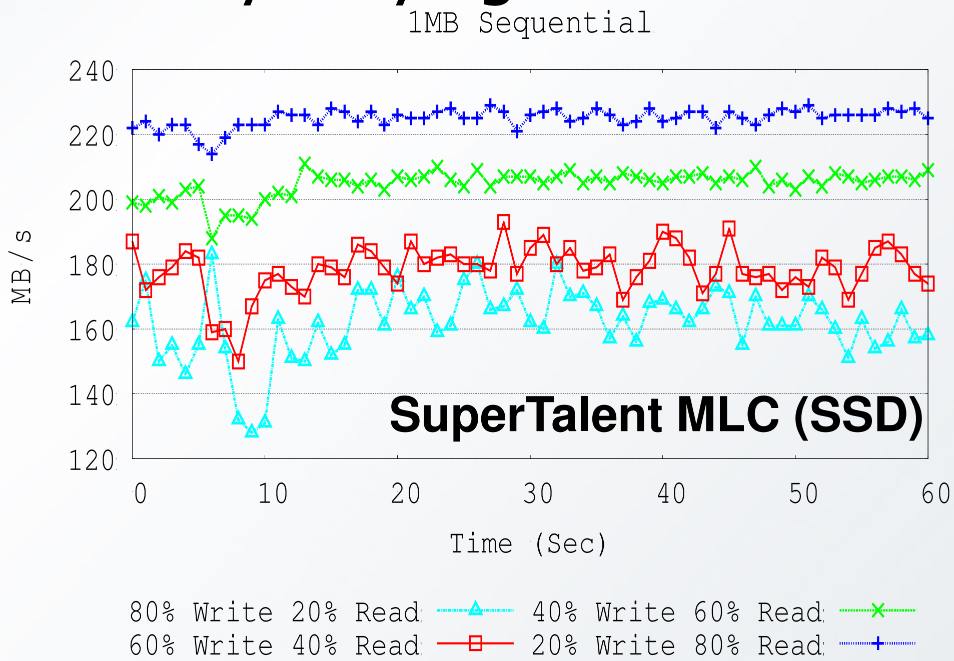
- Does GC have an impact on the foreground operations?
  - If so, we can observe sudden bandwidth drop
  - More drop with more write requests
  - More drop with more bursty workloads

- Experimental Setup
  - SSD devices
    - Intel (SLC) 64GB SSD
    - SuperTalent (MLC) 120GB SSD
  - I/O generator
    - Used *libaio* asynchronous I/O library for block-level testing

Bandwidth Drop for Write-Dominant Workloads

- Experiments
  - Measured bandwidth for 1MB by varying read-write ratio



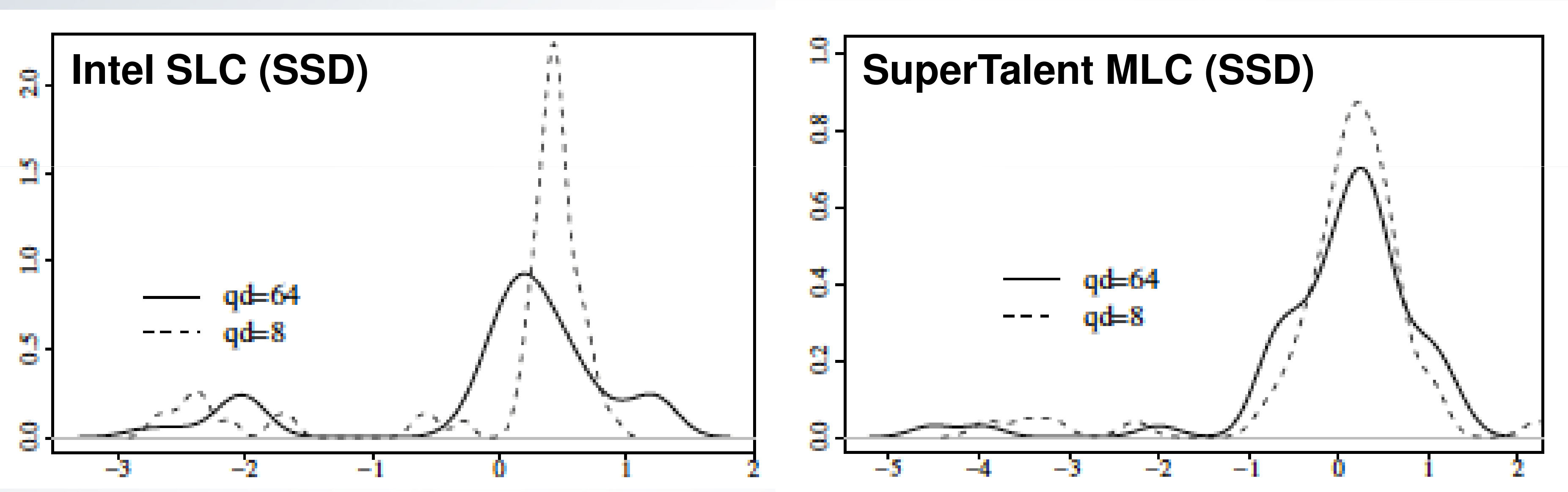


Performance variability increases as we increase write-percentage of workloads.

Georgia Tech

Performance Variability for Bursty Workloads

- Experiments
  - Measured SSD write bandwidth for queue depth (qd) is 8 and 64



Performance variability increases as we increase the arrival-rate of requests (bursty workloads).

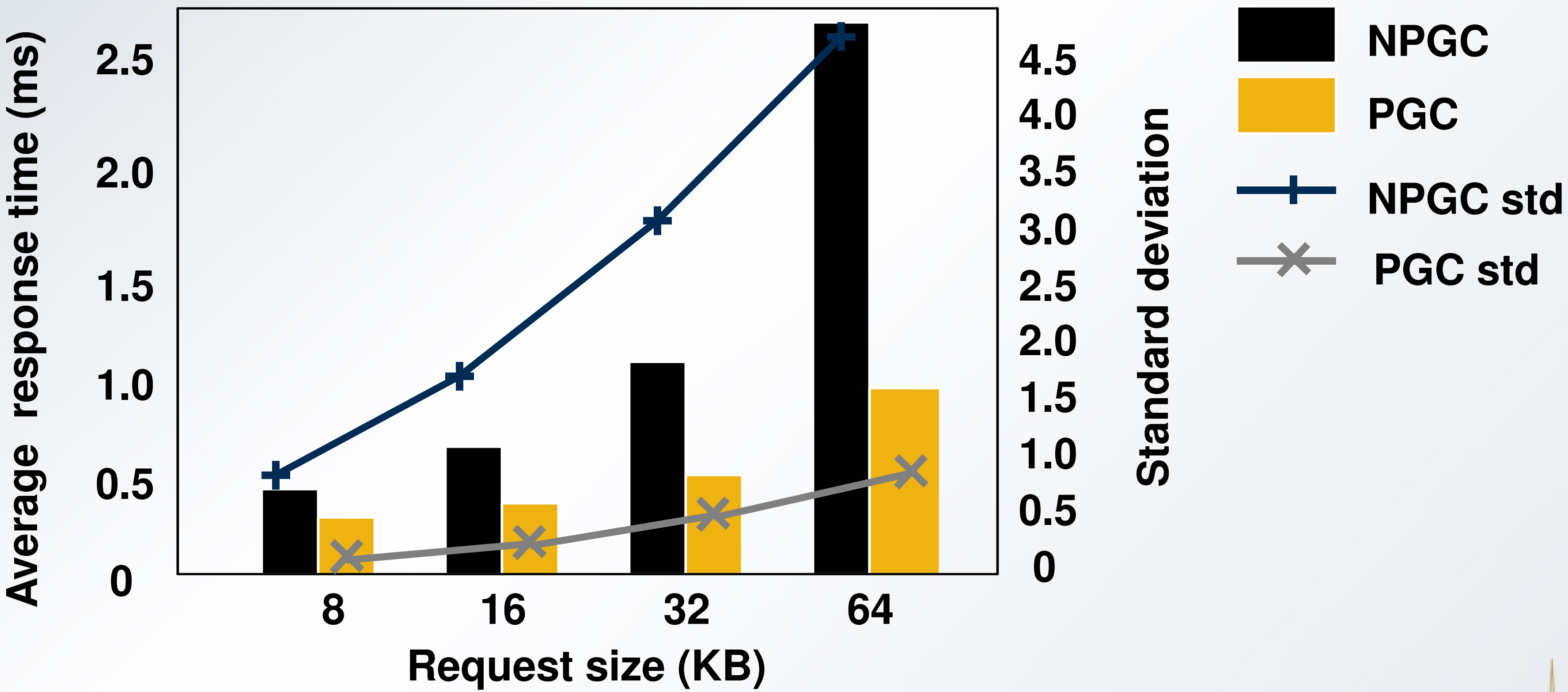
Lessons Learned

- From the empirical study, we learned:
  – Performance variability increases as the percentage of writes in workloads increases.
  – Performance variability increases with respect to the arrival rate of write requests.

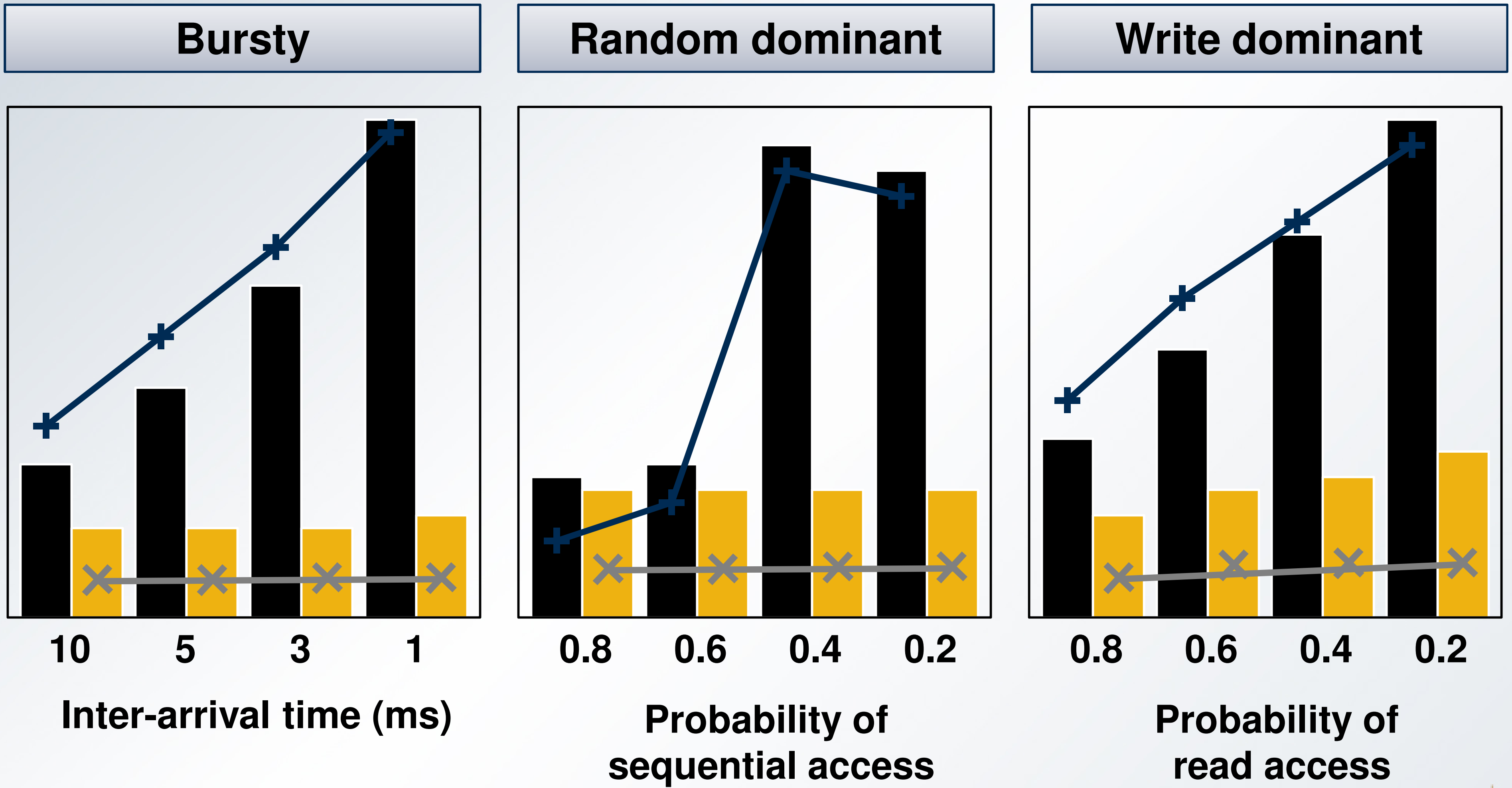
- This is because:
  – Any incoming requests during the GC should wait until the on-going GC ends.
  – ***GC is not preemptive***

Performance Improvements for Synthetic Workloads

- Varied four parameters: request size, inter-arrival time, sequentiality and read/write ratio
- Varied one at a time fixing others



Performance Improvement for Synthetic Workloads (con't)



Georgia Tech

Hardware-assisted Intrusion Detection by Preserving Reference Information Integrity

Junghee Lee, Chrysostomos Nicopoulos,
Gi Hwan Oh, Sang-Won Lee, and Jongman Kim

Georgia Institute of Technology®