From Black Box to Grey Box: Is it Feasible for Flash?

NVRAMOS'14
2014. 10. 31

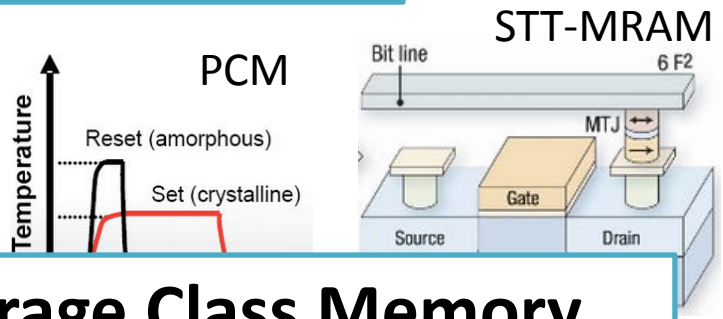
Seoul National University
Prof. Heon Y. Yeom

New NVRAM Storage Systems

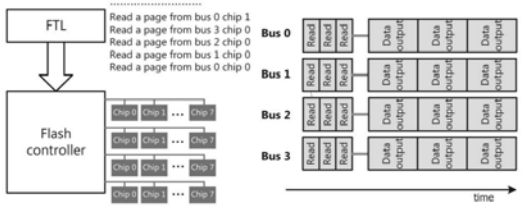
HDD

Response Time =
Seek (10ms)
+ Rotational Delay (0.8ms)
+ Transfer Time

Flash Memory



Storage Class Memory



Ozone(O3): An out-of-order Flash memory Controller Architecture, IEEE Trans. On Computes, May 2011.

Parallel Architecture

Ordinary Practice to Use SSD
“Storage as a Black Box”

No modification to Software

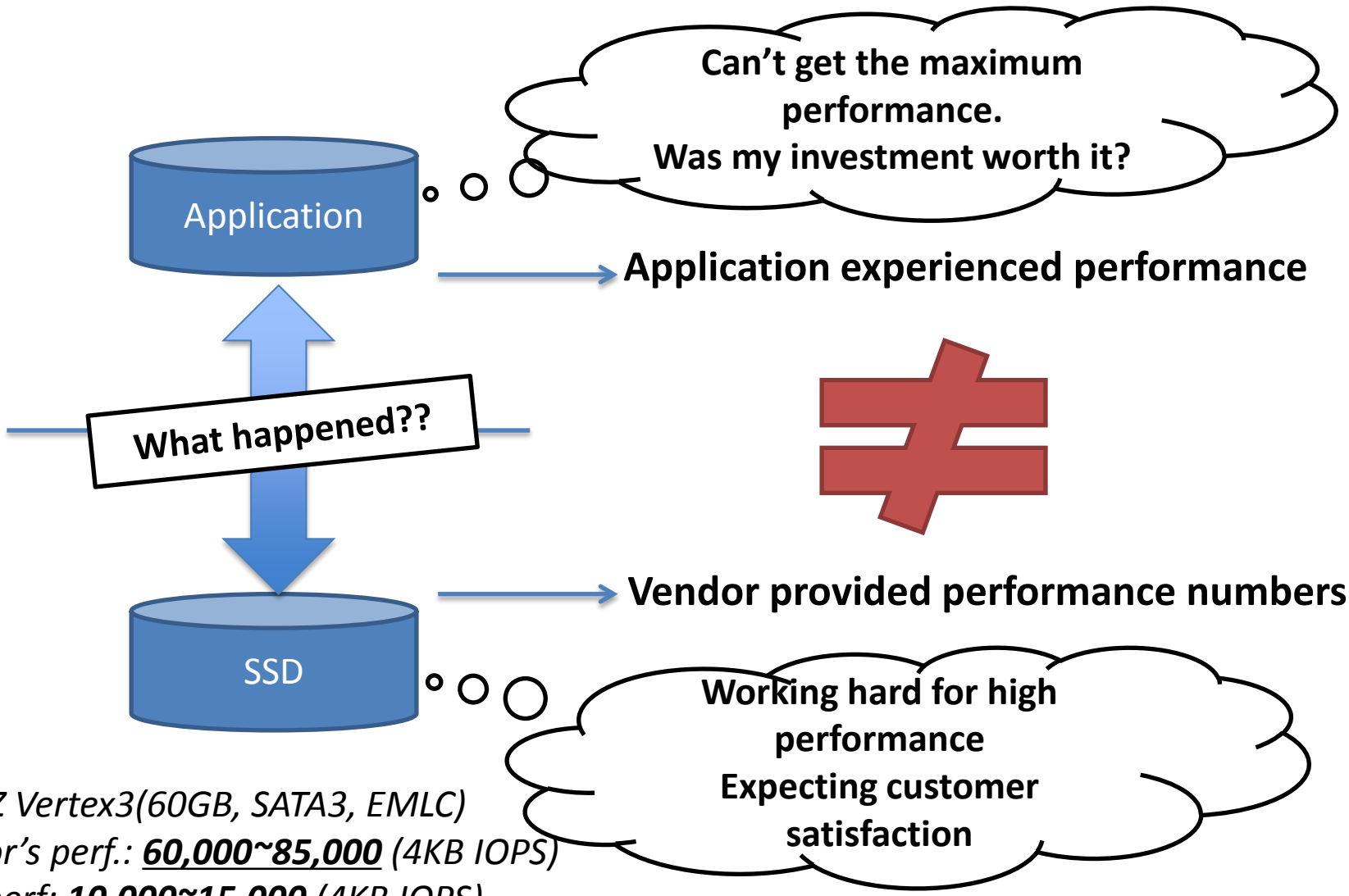
Application

Operating System

Replacing the h/w only

Drop in replacement: “Free Lunch” Performance

Unsatisfied Expectations

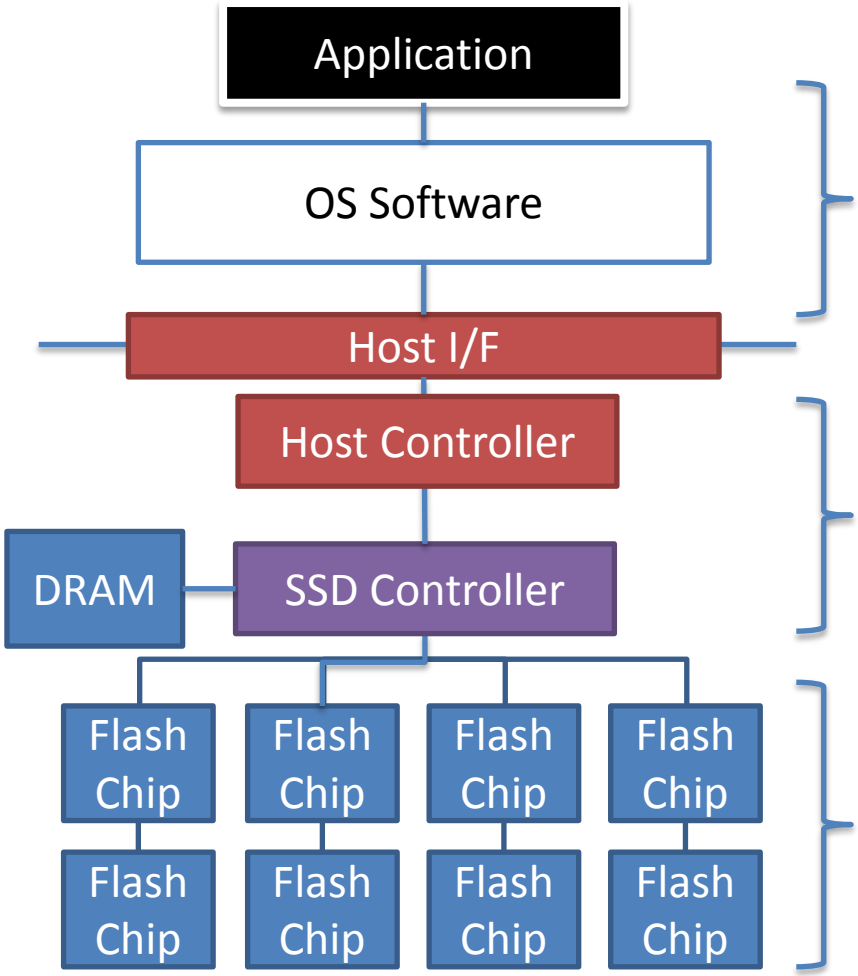


e.g.) OCZ Vertex3(60GB, SATA3, EMLC)

➔ *Vendor's perf.:* ***60,000~85,000*** *(4KB IOPS)*

➔ *Fio's perf:* ***10,000~15,000*** *(4KB IOPS)*

“Inefficient Resource Usage”



Unexpected Performance Drop

- Unpredictable SSD performance
- Unexpected performance degradation
- OS Software overhead

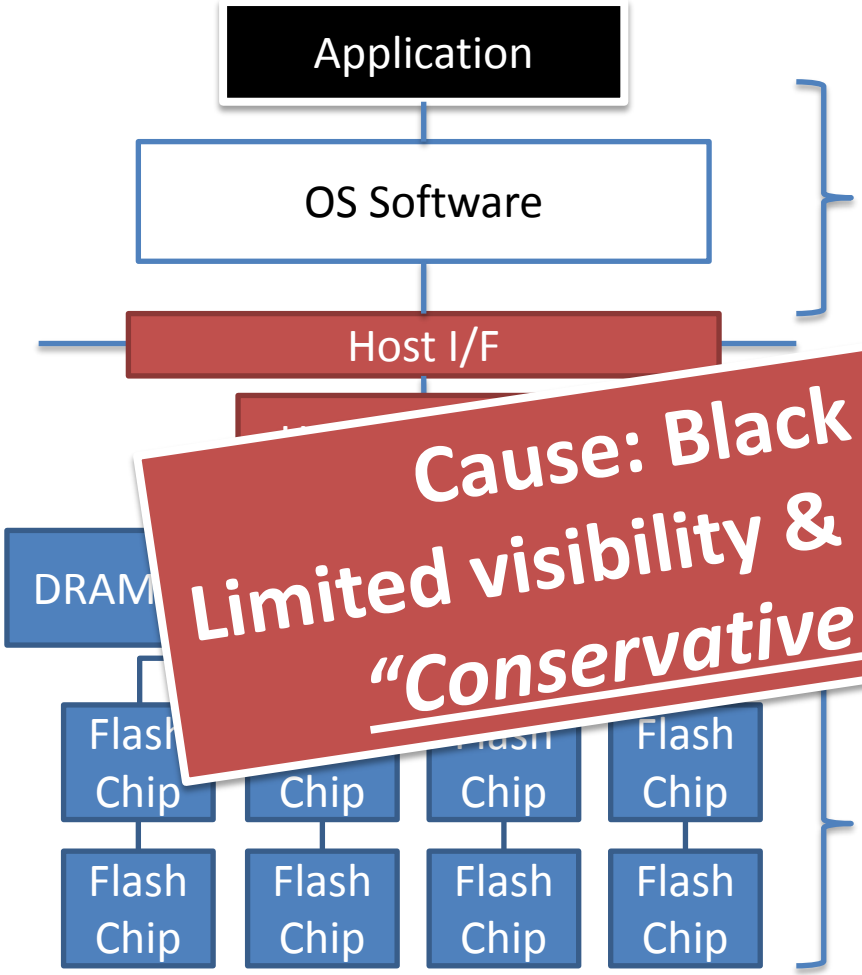
Low Bandwidth Utilization

- Excessive round trips / low bandwidth
- Good-put vs Bad-put (I/O amplification)

Low Resource Utilization

- Unpredictable application behavior
- Bad I/O patterns (bad cache / prefetch hits)
- Low parallelism, hot spots / collisions

“Inefficient Resource Usage”



Unexpected Performance Drop

- Unpredictable SSD performance
- Unexpected performance degradation
- OS Software ov

Cause: Black Box Approach
Limited visibility & access across borders
“Conservative Resource Usage”

width
cation)

Low Resource Utilization

- Unpredictable application behavior
- Bad I/O patterns (bad cache / prefetch hits)
- Low parallelism, hot spots / collisions

Storage is still a “Black Box”

Category	“SCSI”, “SAS”	“SATA”	“NVM-e”
Physical Interface	SAS, FC, PCI-e, SATA*	SATA, PCI-e*	Not specified*
Scope	FF, Phy, Link, Transport, Reg. level I/F, Command protocol Programming arch.	FF, Phy, Link, Transport, Command protocol	Reg. level I/F, Command protocol
Target Devices	Tape, Printer, Storage array, Object Storage, CD/DVD, HDD, SSD, and more	CD-ROM (ejectable media), HDD, SSD	PCI-e SSD, Next generation memory
Register level I/F	Vendor specific, SCSI Express*	AHCI	NVM-Express
Command Protocol	SCSI command set (SCC,SPC	ATA-8/ATAPI command set	NVM-Express command set
Available Abstractions (in standard)	• (***remote***) ***Sequential*** / ***Random*** **Access Block Space** • Cache, Buffers • ***Queue (SCSI-express only)*** • etc: Speaker, Tape, Stream, and more…	• **Random Access Block Space** • Cache (+NVCache) • **Queue (short)** • Interrupts (MSI/MSI-x) • etc: power ctrl, swappable media, monitor(SMART), NVRAM (firmware)	• **Random Access Block Space** • Cache • **Queue Pairs** (***deep, multi***) • Interrupts (MSI/MSI-x) • etc: power ctrl, NVRAM (firmware), metadata/LBA (OOB), etc…

Storage is still a “Black Box”

Category	“SCSI”, “SAS”	“SATA”	“NVM-e”
Physical Interface	SAS, FC, PCI-e, SATA*	SATA, PCI-e*	Not specified*
Scope	FF, Phy, Link, Transport, Reg. level I/F, Command pro Program	FF, Phy, Link, Transport, Command protocol	Reg. l vel I/F, Comman otocol
Target Devices	Tape, Printer, Storage array, Object Storage, CD/DVD, HDD, SSD, and more	CD-ROM (ejectable media), HDD, SSD	PCI- D, Next genera ion memory
Register level I/F	Vendor specific, SCSI Express*	AHCI	NVM-Express
Command Protocol	SCSI command set (SCC,SPC	ATA-8/ATAPI command set	NVM-Express command set
Available Abstractions (in standard)	• (***remote***) ***Sequential*** / ***Random*** **Access Block Space** • Cache, Buffers • ***Queue*** ***(SCSI-e*** • etc: Speaker, Tape, Stream, and more…	• **Random Access Block Space** • Cache (+NVCache) • **Queue (short)** swappable media, monitor(SMART), NVRAM (firmware)	• **Random Access Block Space** • Cache • **Queue Pa** -x) • etc: powe rl, NVRAM (firmware), metadata/LBA (OOB), etc…

Target Domain

Target Domain: Smaller Scope (Specialized)
(Standard & target devices get coupled)

Abstractions: Storage is always a black box
Random access block space with caches & queues

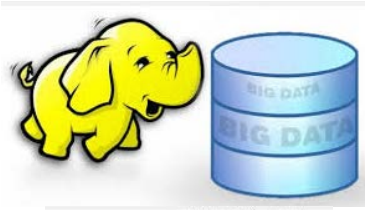
Towards Resource Efficiency?
"Model based Control"

I/O access pattern

Application Behavioral Model

Device Firmware

Resource Allocation
Results (performance)



I/O latency / throughput

Host S/W

SSD Behavioral Model

I/O control

Towards Resource Efficiency?
"Model based Control"

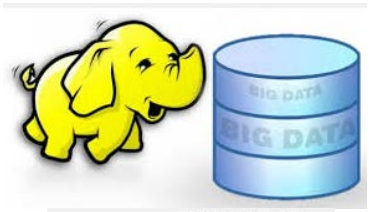
I/O access pattern

Application Behavioral Model

Device Firmware

Resource Allocation

Results (performance)



Attempt to compensate the non-visibility & non-accessible peer resources by utilizing black box behavioral models

Not Quite Successful!!

Not accurate & Not free

I/O latency / throughput

Host S/W

SSD Behavioral Model

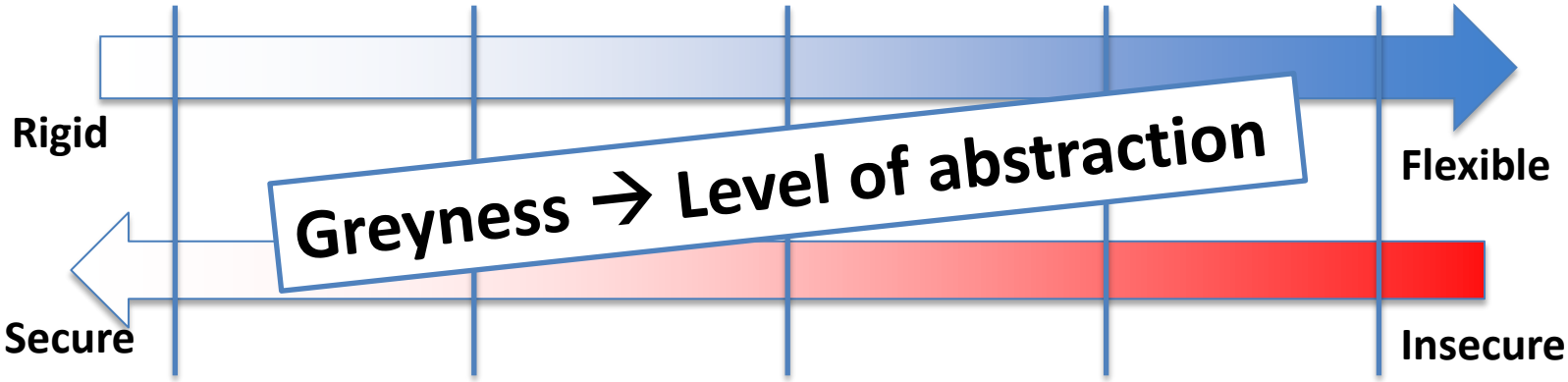
I/O control

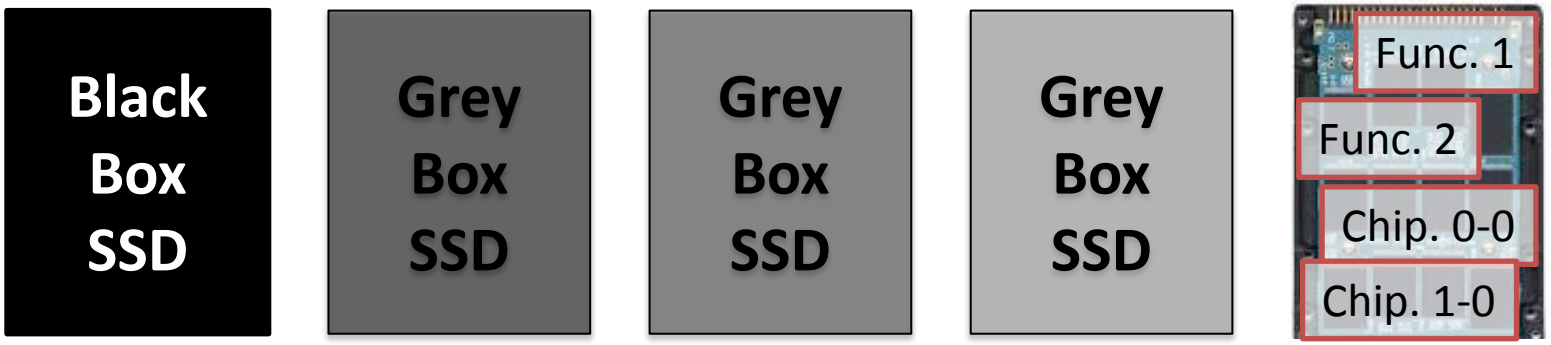
Solution: "Open Up!!"
Grey Box Approach with SSDs

- **SSD internals exposed to host S/W via I/Fs**
  - Provides means of visibility of peer resources
  - Provides means of access to peer resources
  - via well defined interfaces

- **In a managed way**
  - Resources ***abstracted at a proper level*** to ***hide proprietary details*** while ***providing flexibility***
  - Preserve security, robustness, orthogonality

Grey Box Approach with SSDs





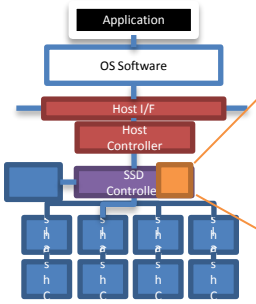
Read(LBA,size)
Write(LBA,size)

All Open

Flexibility for the sake of efficient resource usage

Appropriate level of abstraction to protect proprietary details

What to Expose?



Abstracted Resource

Set of operations (API)

Resource info
(Resource Behavioral Model)
Application info
(Application model)

Misc. operations

Interface defined for Abstracted Resources

High lvl.

Examples>

Low lvl.

GC scheduler

Pre-fetcher

Channels

Resource Model: GC thresholds

Application Model (code): Read stream context

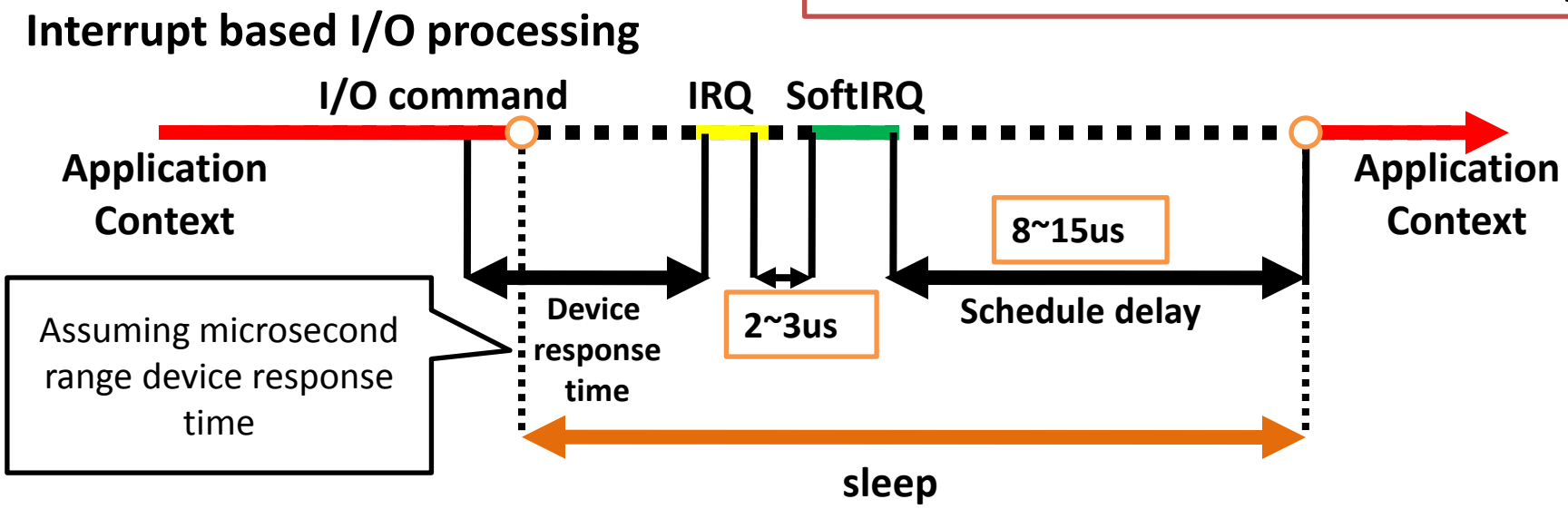
Resource Model:
read(), write(), erase() timings
per channel block mapping

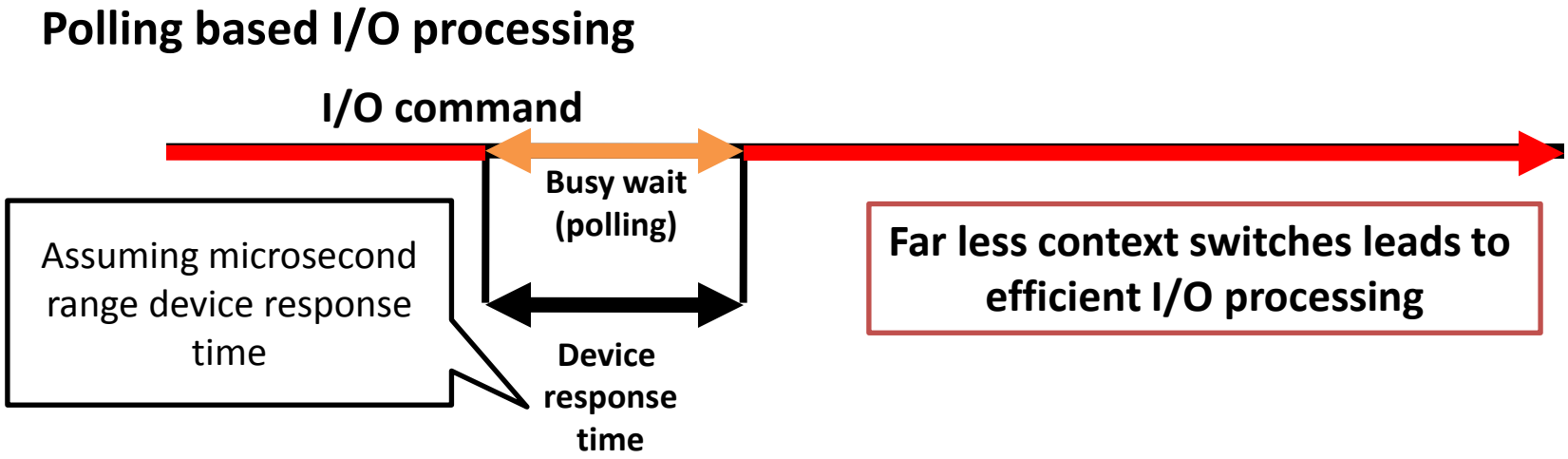
Case Studies:
Towards Efficient Resource Utilization
w/ the Grey Box Approach

- **Optimizing I/O completion**
- **Optimizing DB Transaction I/O**
- **Optimizing on-storage graph traversal**
- **Optimizing SSD latency**
- SSD cache prefetching
- Multi-streamed SSD
- Computation offloading
  (query processing, filters, compression & etc.)

Optimizing I/O Completion (1/3)

To Poll or to wait for an Interrupt

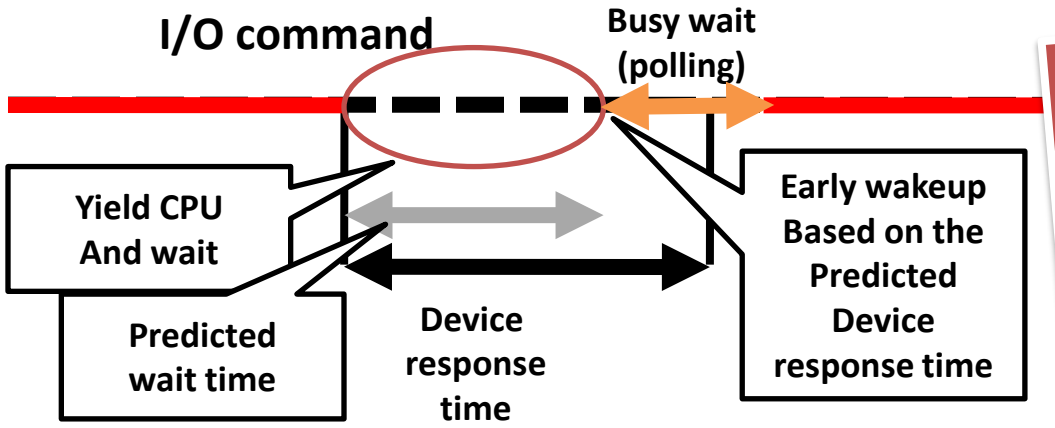




Optimizing I/O Completion (2/3)

- **Problem with polling**
  - High CPU usage
  - High bus utilization (frequent control register access)
  - Low parallelism
- **Dynamic poll**
  - D. Shin et al, "Dynamic Interval Polling and Pipelined Post I/O Processing for Low-Latency Storage Class Memory," HotStorage 2013
  - ***Solves the problem of polling by predicting device response time***

Dynamic Polling based I/O processing



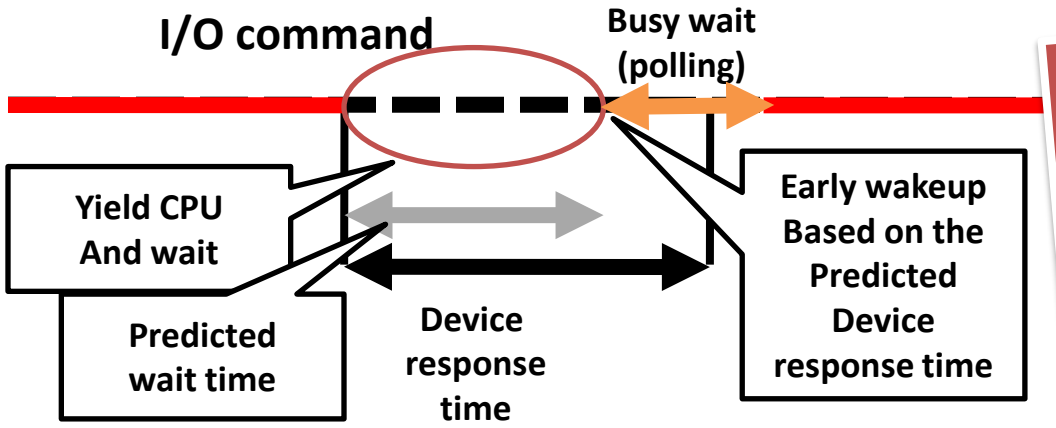
Not possible with Flash SSDs

Cannot predict Device response time (GC, Buffer flush, collisions & etc)

Optimizing I/O Completion (2/3)

- **Problem with polling**
  - High CPU usage
  - High bus utilization (frequent control register access)
  - Low parallelism
- **Dynamic poll**
  - D. Shin et al, "Dynamic Interval Polling and Pipelined Post I/O Processing for Low-Latency Storage Class Memory," HotStorage 2013
  - ***Solves the problem of polling by predicting device response time***

Dynamic Polling based I/O processing

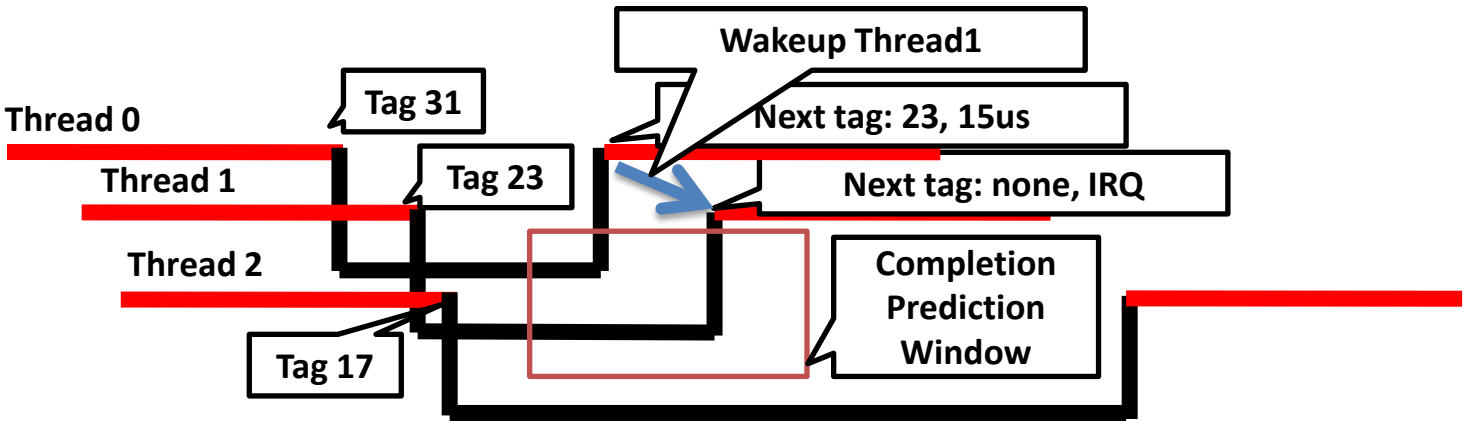


Not possible with Flash SSDs

What if?
Device informs the OS (i.e., time left to the next completion?)

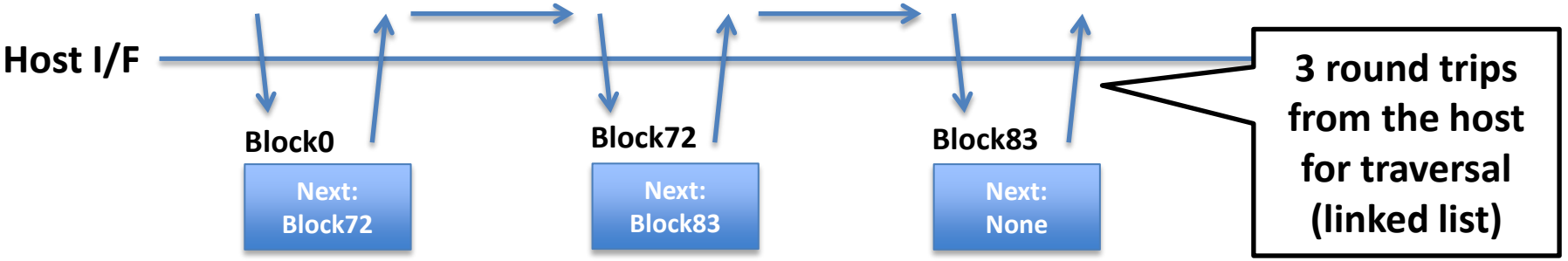
Optimizing I/O Completion (3/3)

- **Grey Box Approach**
  - **Idea:** The SSD explicitly informs the OS software for the I/O completion
  - **Method:** Piggy back the tag and time left for the next I/O completion on each I/O
  - **Behavioral model: "Best effort time to completion"**
    - Based on information of I/O requests in the completion queue
    - Inform the next I/O processor to prepare next I/O completion
  - **Interface:**
    - Piggy backed info: "Time to next completion"
  - **Impact:**
    - Improves I/O processing latency & throughput



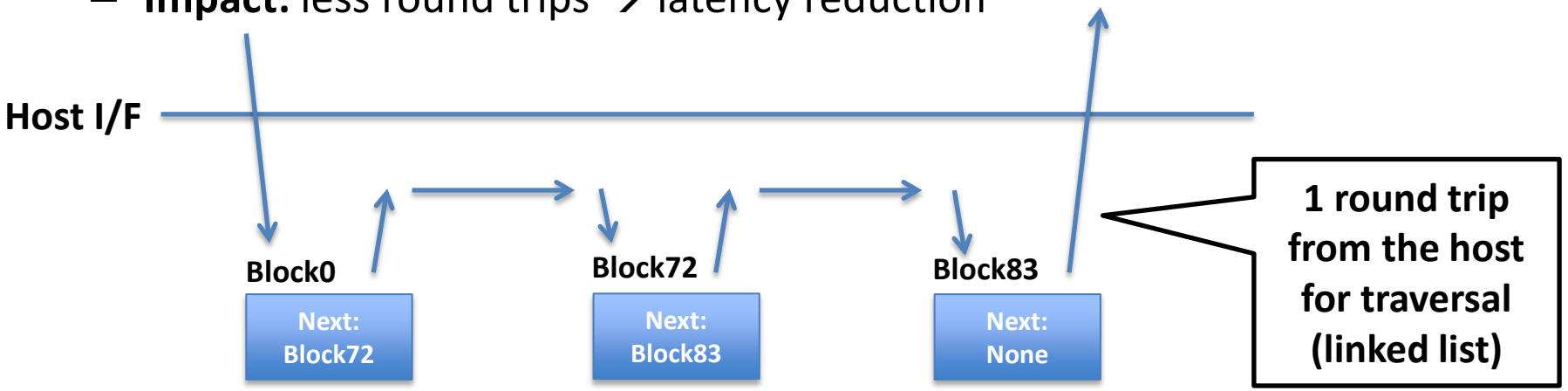
Optimizing On-storage Graph Traversal (1/2)

- **On-storage graph traversal**
  - Read I/O on a series of blocks which have dependency (i.e., i+1th block requires the ith block read)
  - i.e., B-tree lookup, social graph traversal
- **Problem**
  - Low parallelism (cannot batch: can't predict next move)
  - Multiple round trips (flash reads) for graph traversal



Optimizing On-storage Graph Traversal (2/2)

- **Grey Box Approach**
  - **Idea:** Inform SSD with the block traversal semantics
  - **Method:** Trusted traversal code execution, or block format info
  - **Application Behavioral Model:** Application block traversal logic provided to the SSD
  - **Interface:** Means to inform the SSD with the application logic (trusted code?)
  - **Impact:** less round trips → latency reduction



Optimizing DB Transaction I/O (1/3)

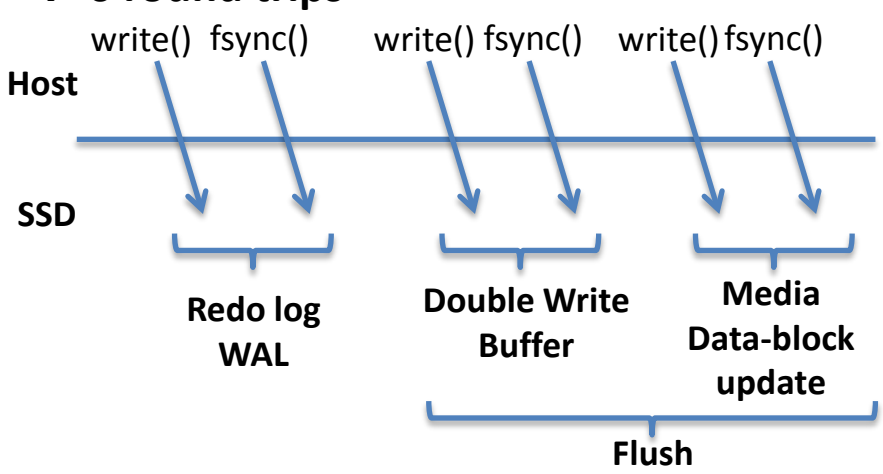
- **Problems with current storage with transactions**
  - Current storages are not ‘stable’: ***should avoid partial writes***
  - Current storages do not guarantee ‘durability / order’ on the common case: ***durable writes require multiple costly cache flushes***
  - Multiple writes (write amplification / multiple round trips) required to preserve both ‘stable’, ‘durable’ and ‘order’ properties

- **Transactions with flash SSDs?**
  - Implementing a ‘stable’ storage with flash SSDs can be efficient: append only writes (out of place updates)
  - Can simplify DB storage engine designs w/ transactional support

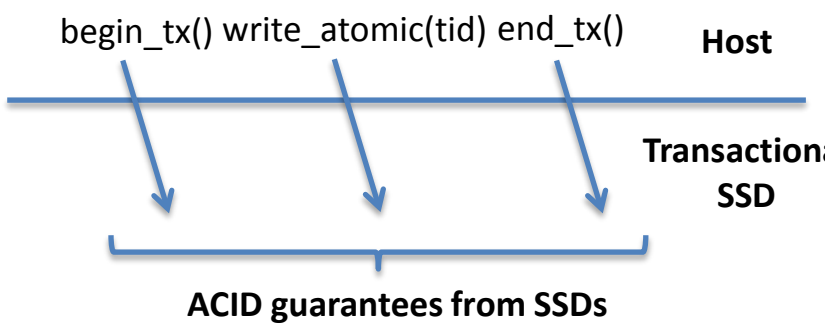
Optimizing DB Transaction I/O (2/3)

- **Grey Box Approach**
  - **Idea:** Let SSDs have transactional support
  - **Method:** SSDs provide transactional features and guarantees
  - **Application Model:** ACID properties on writes, WAL semantics, commit protocol
  - **Interface:** atomic write, begin_tx, end_tx, abort & etc…
  - **Impact:** less round trips, less writes, efficient storage usage (append only)

e.g., InnoDB write protocol:
→ 3 writes & fsyncs for an update
→ 6 round trips



I/O to a transactional SSD
→ 1 atomic write w/ tx begin&end
→ 3 round trips



Optimizing DB Transaction I/O (3/3)

- **Related Systems**
  - [**TxFlash**] V. Prabhakaran, T. L. Rodeheffer, and L. Zhou, ***"Transactional flash,"***, OSDI'08
  - [**AtomicWrites**] X. Ouyang, D. Nellans, R. Wipfel, D. Flynn, and D. K. Panda, ***"Beyond block I/O: Rethinking traditional storage primitives,"*** HPCA'11
  - [**LightTx**] Y. Lu, J. Shu, J. Guo, S. Li, and O. Mutlu, ***"LightTx: A lightweight transactional design in flash-based SSDs to support flexible transactions,"*** ICCD'13
  - [**Mobius**] W. Shi, D. Wang, Z. Wang, and D. Ju, ***"Mobius : A High Performance Transactional SSD with Rich Primitives,"*** MSST'14

Optimizing SSD Latency (1/4)

- **Problem with resource collisions**
  - Reads, Writes, GC (valid page copy & erase)
  - I/O operations colliding on
    SSD internal channels, chips, dies and planes
  - Uncontrollable & unexpected latency spikes
    - Long tail latency of SSDs

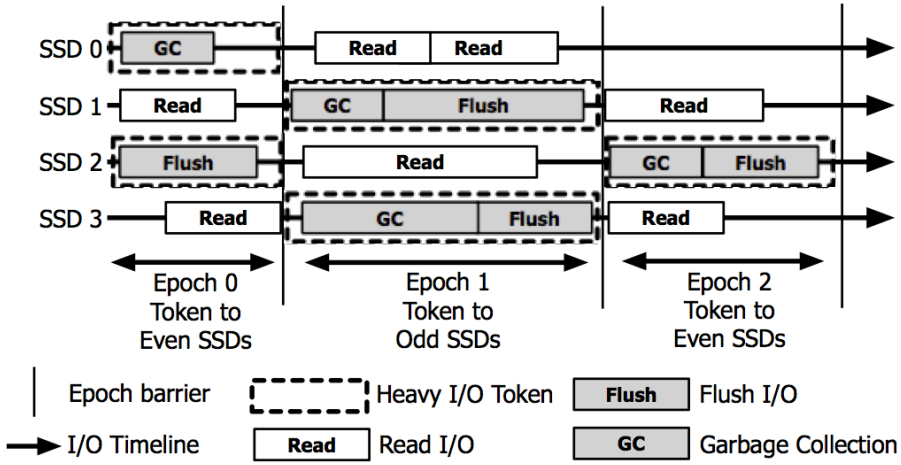
- **Cause: Non-visibility & non-accessible SSD internals**
  - Cannot control when to trigger GC operations
  - Cannot see which channel is idle

Optimizing SSD Latency (2/4)

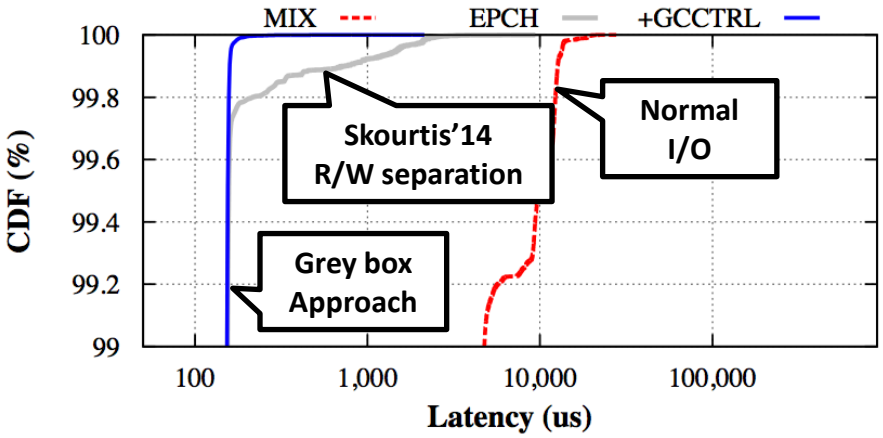
- **Grey Box Approach**
  - **Idea:** Have the application explicitly schedule I/O & GC operations on multiple channels
  - **Method:** Expose GC & I/O operations w/ queue abstractions on each channels, data replicated on distinct channels (2 replicas)
  - **Model:** GC initiating threshold & current level of free blocks
  - **Impact:** suppress latency spikes

Optimizing SSD Latency (3/4)

- **Mockup Grey Box Approach Experiment**
  - Use multiple SSDs instead of SSD channels
    (Requires H/W resource visibility)
  - Latency sensitive & latency heavy I/O separation using replicas placed on redundant H/W resources
    (similar to read / write separation in Skourtis14)
    - [**Skourtis14**] D. Skourtis, et al, "Flash on Rails : Consistent Flash Performance through Redundancy", ATC'14
  - GC control API enhanced SATA 6.0Gb/s SSDs provided by Samsung



Latency spike suppression
(Epoch based R/W/GC separation)



Cutting the long tail

Optimizing SSD Latency (4/4)

- **Related Projects**
  - [**SDF**] J. Ouyang, S. Lin, S. Jiang, Z. Hou, Y. Wang, and Y. Wang, ***"SDF: Software-Defined Flash for Web-Scale Internet Storage Systems,"*** ASPLOS '14
  - [**Rails**] D. Skourtis, D. Achlioptas, N. Watkins, C. Maltzahn, and S. Brandt, ***"Flash on Rails : Consistent Flash Performance through Redundancy,"*** ATC'14.
  - [**HIOS**] M. Jung, W. Choi, and S. Srikantaiah, ***"HIOS: A host interface I/O scheduler for Solid State Disks,"*** ISCA '14
  - [**PIQ**] C. Gao, L. Shi, M. Zhao, C. J. Xue, K. Wu, and E. H.-M. Sha, ***"Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives,"*** MSST'14

Challenges

- **So many specialized API instances?**
  - Specialization leads to multiple instances of APIs
  - Need a way to lower the cost of API development and maintenance
- **Market Adoption, Business model etc.**
  - Would there be a market large enough?
  - What is the killer application of the approach?

Suggestions

- **Programmable SSDs**
  - ***Define and develop a generic programmable SSD platform*** to enable easy SSD behavior modification
    - ***Ex> Willow (OSDI'14) UCSD***
  - Similar to Nvidia CUDA GP-GPU platform, Apple iOS app platform, Android app platform

- **Looking for killer apps**
  - ***"Provide a generic programmable SSD platform to the community"***
  - ***Collective intelligence*** of multiple ***seed developer groups*** in the ***industry*** and the ***academia*** looking for killer apps (i.e., Open-Source SSD APIs)
  - Expect emerging abstractions, models and applications based on customer needs (industry) or research results (academia)

Conclusion

- **Inefficient resource usage caused by the Black Box storage approach**
  - Non-visible & non-accessible peer resources
    Conservative I/O strategies
- **Solution: Grey Box storage approach**
  - SSD internals exposed to host S/W via I/Fs in a managed way
- **Case studies:**
  - Host S/W can schedule resources to enhance the efficiency of the system → ***Feasible!***
- **Future studies:**
  - Looking for a reliable way to use Grey Box SSDs

Thank You!