

Building High-Performance and Cost-Effective Storage Systems with Flash Memory based Solid State Drives

Xiaodong Zhang

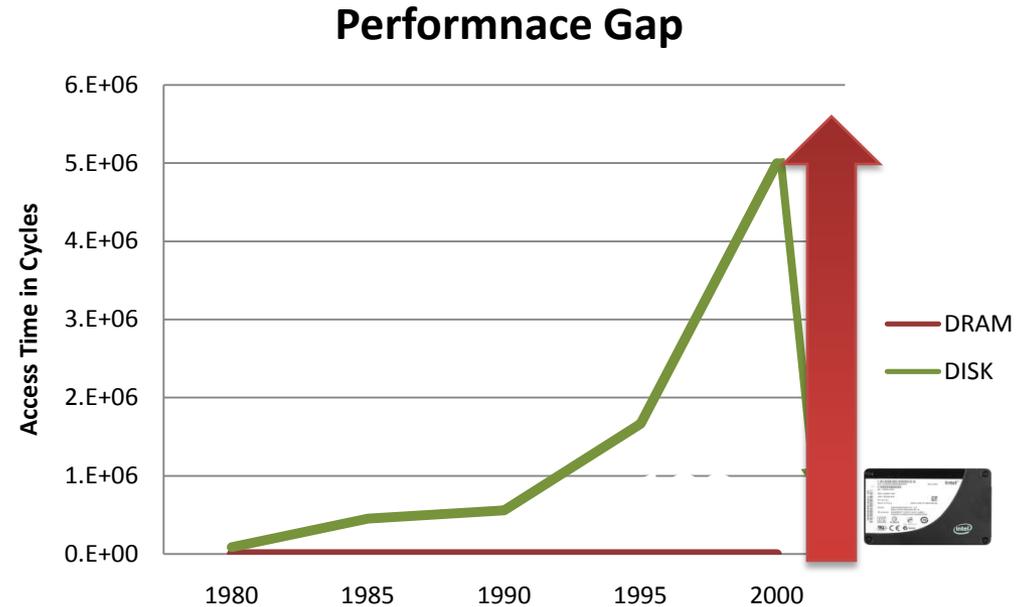
The Ohio State University

Supported in part by NSF CCF-0913050

Other Contributors: Feng Chen (Ohio State) and Intel Labs

Evolution of Storage and new Demand

- Hard disk drive (HDD)
 - Major storage device since 1956
- Merits
 - Large capacity, low cost
 - Most commonly used storage
- Mechanical Nature
 - Unsatisfactory performance
 - High power consumption



Source: Bryant and O'Hallaron, "Computer Systems: A Programmer's Perspective", Prentice Hall, 2003



1956: IBM 305 RAMAC computer with hard disk (5MB/1,200RPM)



1973: IBM 3340
35-70MB



2007: Hitachi GST
Deskstar 7K1000, 1st 1TB



**Emerging and
Future**

Evolution of the 5 Minute Rule

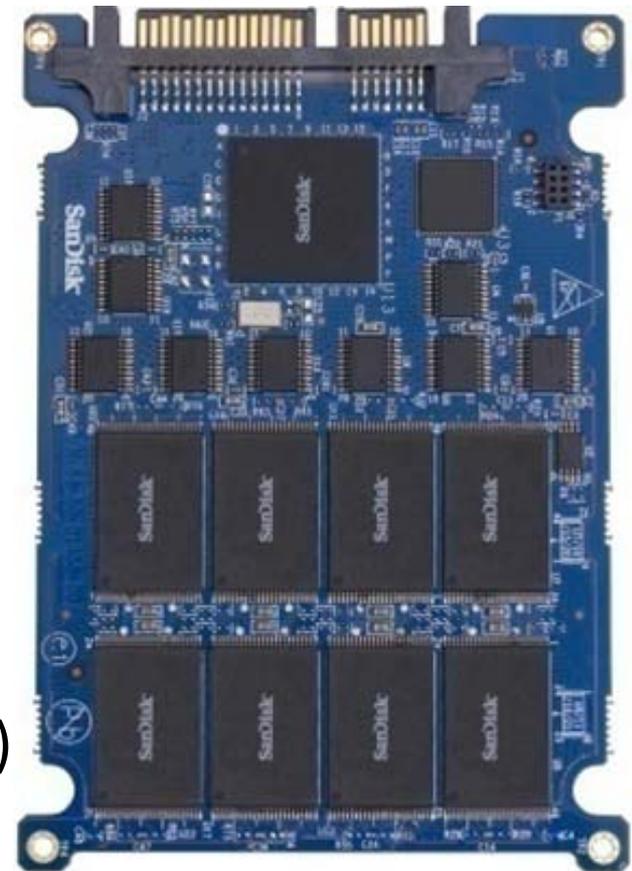
- **First version:** Jim Gray and Franco Putzolu (1987, SIGMOD)
 - **Background:** disk capacity is low and expensive, latency is not an issue
 - Accessing 1 KB data in disk costs \$2,000, but only \$5 in main memory
 - Rule: **pages referenced every 5 minutes should be memory resident**
- **Second version:** Jim Gray and P. Shenoy (2000, ICDE)
 - **Background:** capacity is up 1,000x, bandwidth only 40X, very low price
 - 5 minute rule becomes **a caching rule for performance** due to:
 - (1) Disk accesses slow 10X per decade; (2) disk scanning time increases
- **A recent version:** G. Graefe (CACM, 2009)
 - **Background:** SSD is still expensive, disk space is almost free, low speed
 - For small size blocks, 5 minute rule holds between DRAM/SSD
 - For very large size blocks, 5 minute rule holds between SSD/disks

HDD Improvement has been focused on Density

- Huge capacity disks with low price and small size still have
 - **Low speed and high energy consumption** (current stage)
 - High capacity causes high access latency (for more than 10 years)
- Specific issues and concerns
 - Capacity/bandwidth increases significantly , so does latency
 - Space is almost free, but to access data is increasingly more expensive
 - Economic model: a disk should be infrequently accessed for archival
 - DRAM buffer can address the performance issues, but not the power
- **A fast and low power storage is highly desirable.**

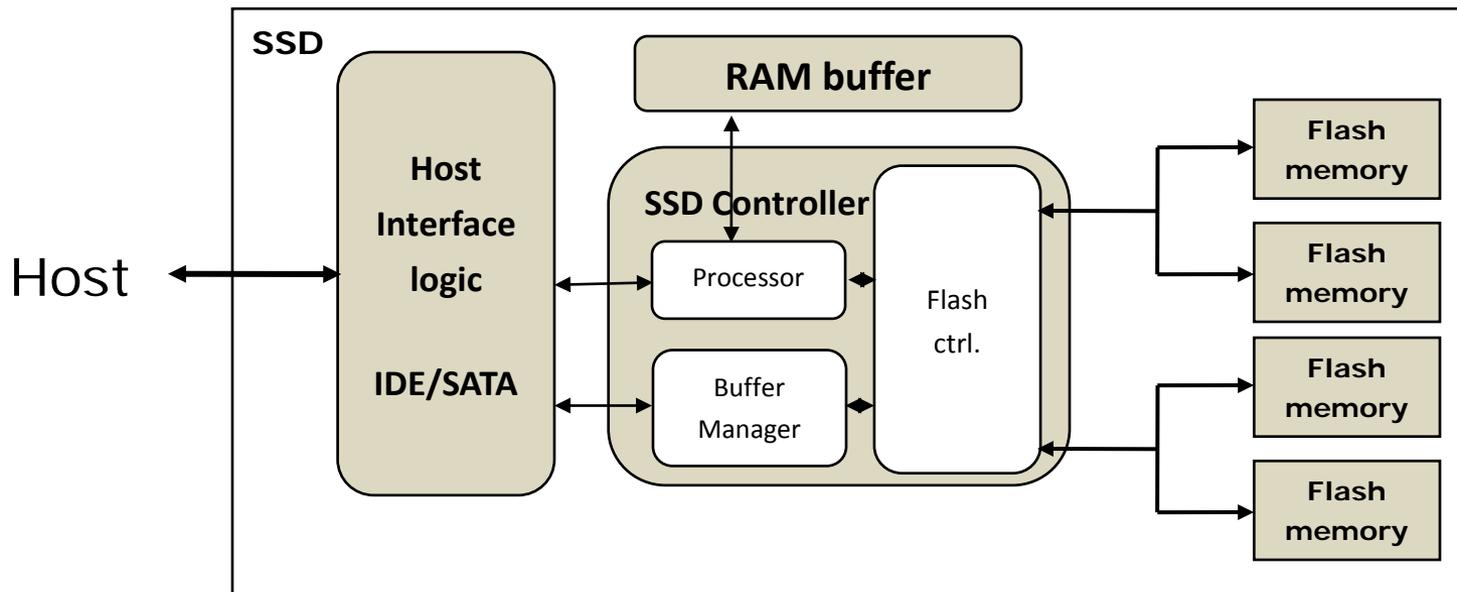
Flash Memory based Solid State Drive

- Solid State Drive (SSD)
 - A **semiconductor** device
 - Mechanical components free
- Technical merits
 - Low latency (e.g. 75 μ s)
 - **High bandwidth** (e.g. 250MB/sec)
 - **Low power:** 0.06 (idle)~2.4w (active)
 - Shock resistance
 - Lifespan: 100GB/day \rightarrow 5 years (x25-M)



Flash Memory based Solid State Drive

- Architecture of solid state drives (SSD)
 - Host interface logic – SATA, IDE, SCSI, etc.
 - **SSD Controller** – processor, buffer manager, flash controller
 - Integrated/Dedicate RAM buffer
 - An array of **flash memory** packages



Adapted from USENIX'08 (Agrawal et al.)

Performance- and Power-Efficient

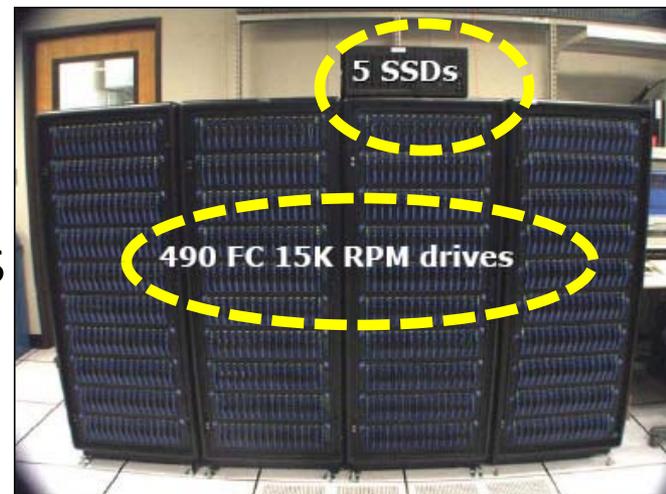


120x HDDs

- 36,000 IOPS
- 12GB/sec
- 1,452 watts
- 8.8TB
- Per HDD
 - 100MB/sec (R/W)
 - 300 IOPS
 - 12.1 watts (active)

• 120x SSDs

- 4,200,000 IOPS
- 36GB/sec
- 288 watts
- 3.8TB
- Per SSD
 - 250/170MB/sec (R/W)
 - 35,000 IOPS (Read)
 - 2.4 watts (active)



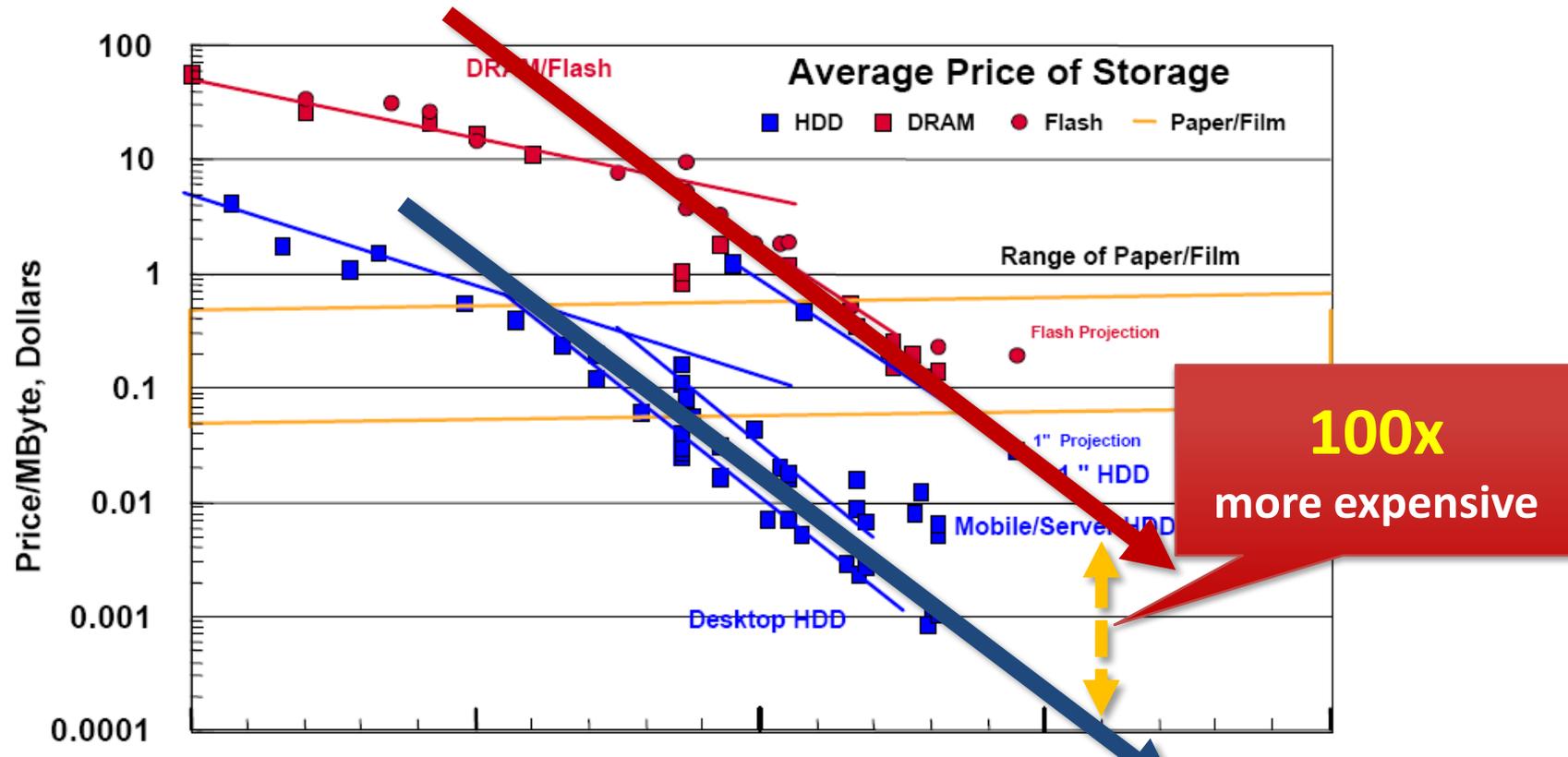
Read
throughput
(IOPS)
115x

Bandwidth
3x

Power Cost
5x

Challenge 1: Affordability

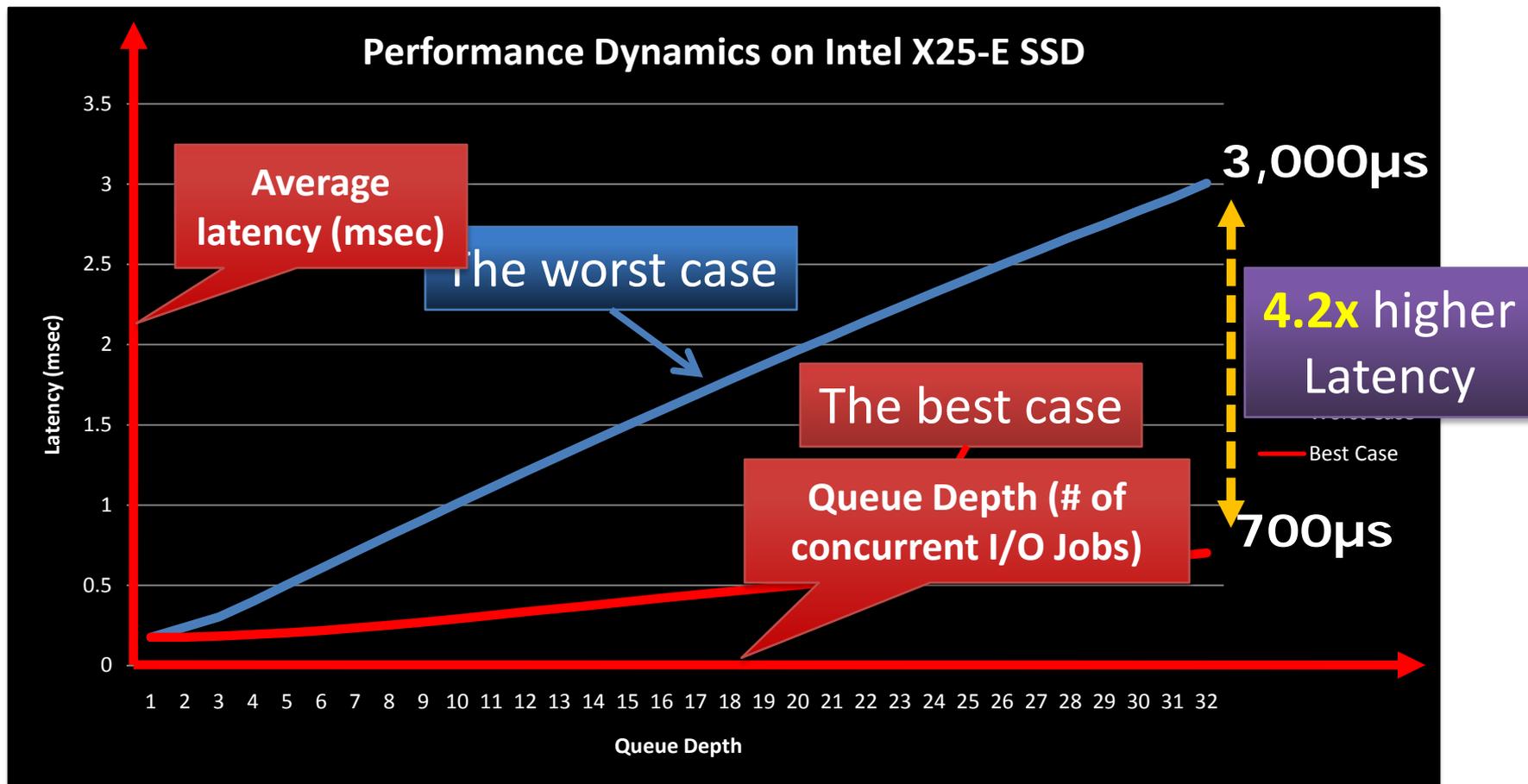
- Flash is about **100x** more expensive than disks



*We need to find a middle ground between SSD and HDD and strike a **right balance** between performance and cost.*

Challenge 2: Performance Dynamics

- Random read 4KB in the 1024MB space with 1~32 I/O jobs (different data allocations among flash chips result in different performance)

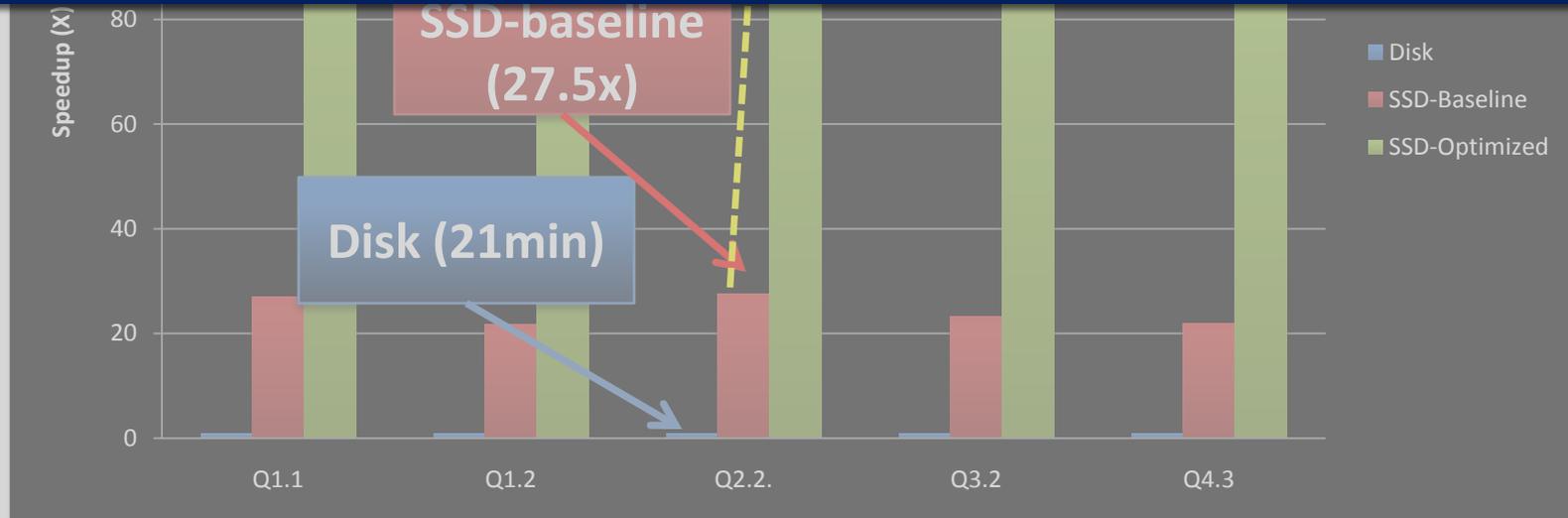


Challenge 3: Resource underutilization

- Database query executions



*However, the high performance potential of SSD cannot be automatically tapped without extensive **research efforts**.*



Critical Issues

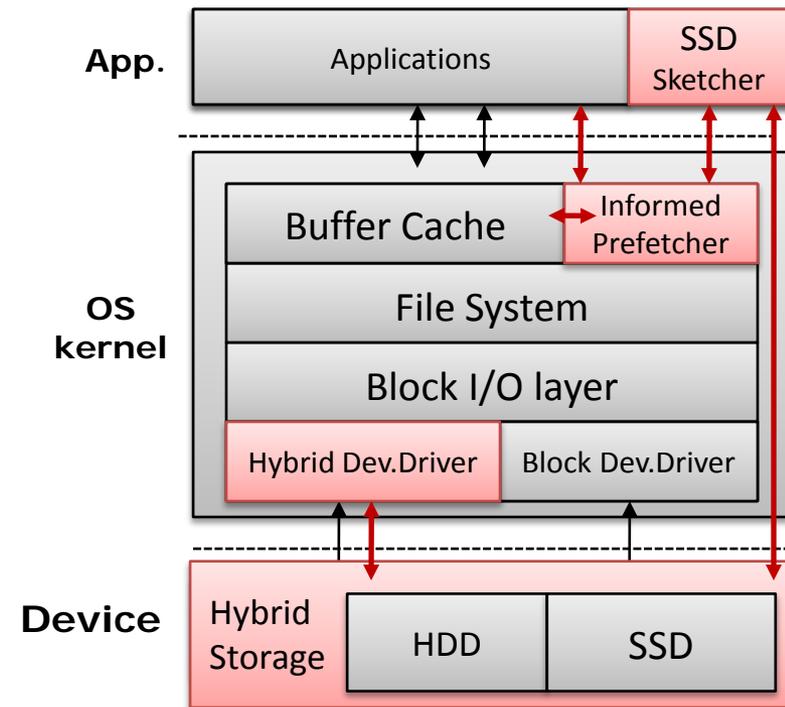
- **Performance dynamics** due to the unknown internals
 - A systematic effort is needed to timely and accurately detect the internal structures of the SSDs
- **Affordability** and **limited capacity** of SSDs
 - A hybrid storage is a best cost- and performance-effective solution
- **Underutilized** rich and hidden storage resources
 - System and application efforts to fully utilize the rich idle/hidden resources, such as internal parallelism
- **Reliability issues** caused by wear-out problem of flash
 - Technical advances are improving lifespan (e.g. 100GB/day → 5 years)

Critical Issues

- **Performance dynamics** due to the unknown internals
 - A systematic effort is needed to timely and accurately detect the internal structures of the SSDs
- **Affordability** and **limited capacity** of SSDs
 - A hybrid storage is a best cost- and performance-effective solution
- **Underutilized** rich and hidden storage resources
 - An effective solution is desirable to utilize the rich idle/hidden resources, in particular internal parallelism
- **Reliability issues** caused by wear-out problem of flash
 - Technical advances are improving lifespan (e.g. 100GB/day → 5 years)

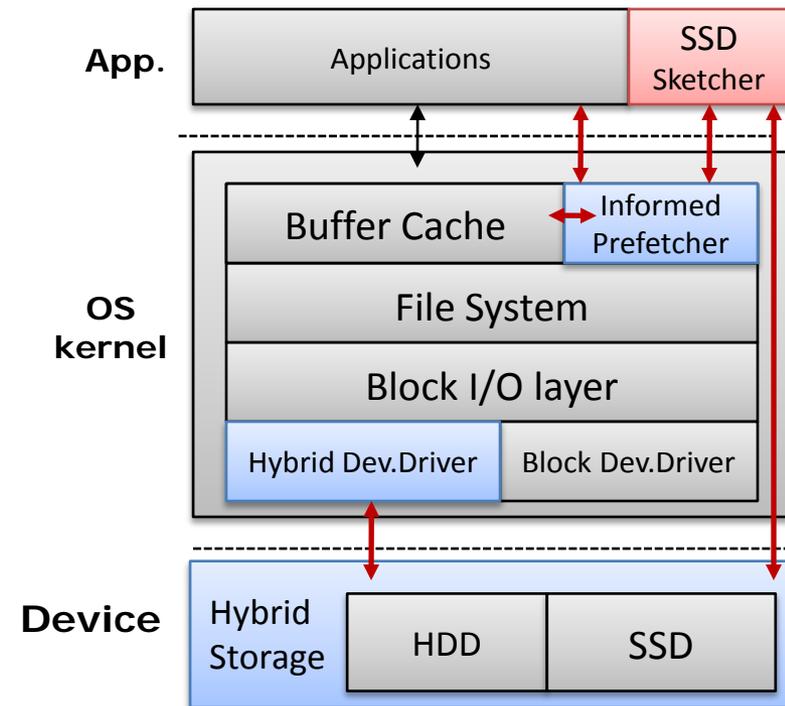
A Framework for a hybrid storage system

- **SSD Sketcher** – Detecting SSD internal structures
- **Hystor** – Providing hybrid storage services
- **Prefetcher** – utilizing the internal resources of SSD
- Other efforts by applications to fully utilize parallelisms.

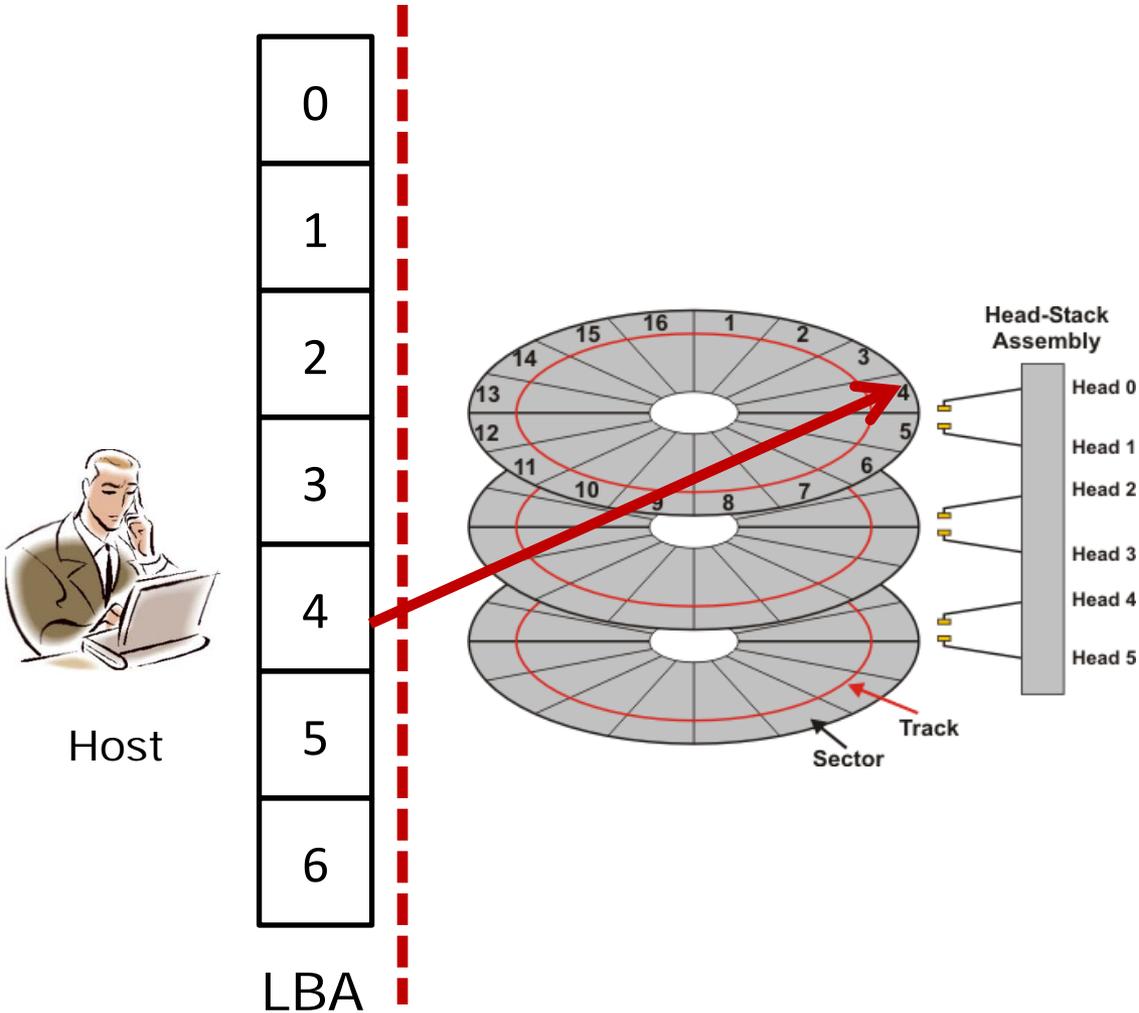


Outline

- Introduction
- Sketching SSD internals
- Hystor: A hybrid storage system
- Exploiting Internal Parallelism
- Conclusion
- Future Work



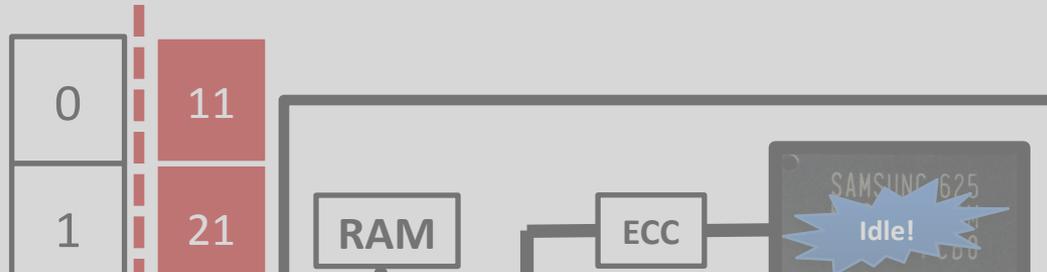
Physical data layout in HDD



- Data are stored on the surfaces of disk platters
- An array of **logical block addresses** (LBAs) as a logical interface
- LBAs are **statically** mapped to physical block addresses (PBAs) in an almost consistent way

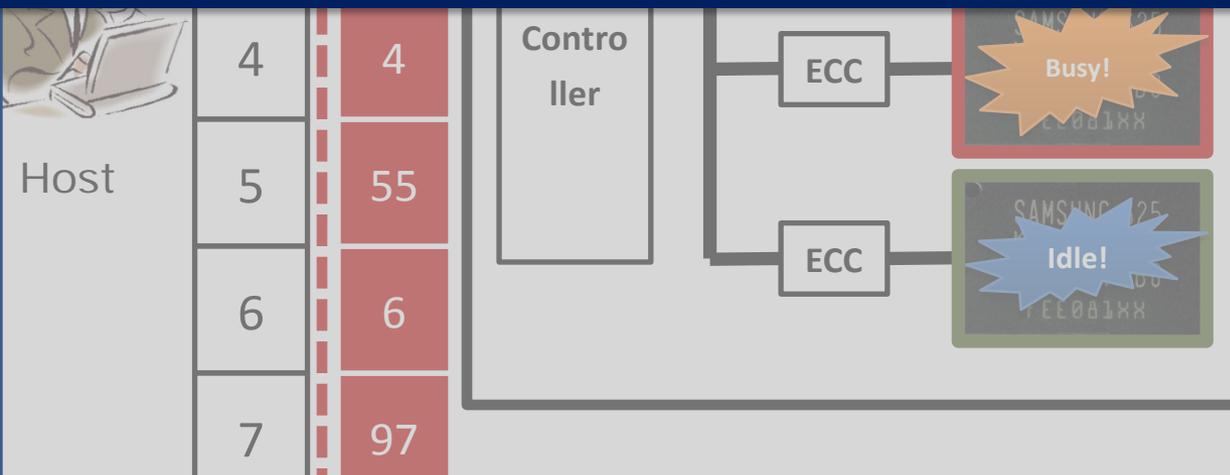
```
Read (LBA, size)
Write(LBA, size)
```

Physical data layout in SSD



- Data are stored in flash memory chips

The physical data mapping is an architectural feature, we must know the internal structures of SSDs.



- A mapping table tracks LBA/PBA mappings
- Internal data mapping is **dynamic** on the fly

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

The worst case

Asking for help from SSD manufacturers?

- **Intellectual property** issues
- **Limited unreliable info** in specification
- The strictly defined **standard** interface

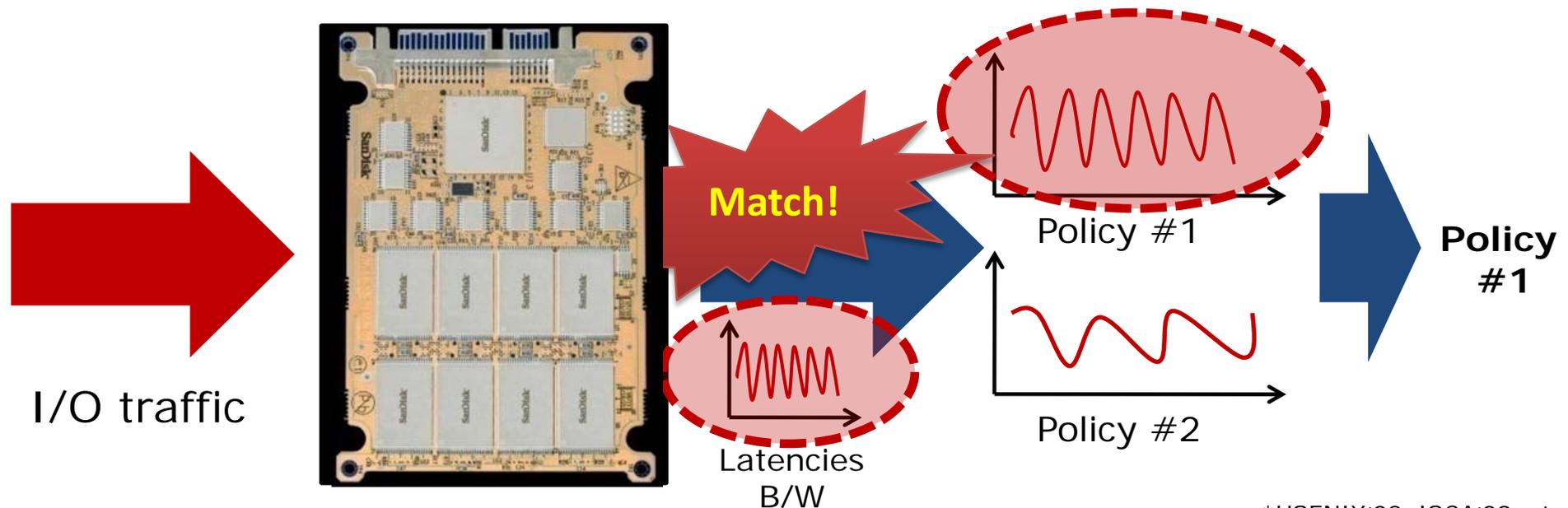
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH [REDACTED] PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN [REDACTED]'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, [REDACTED] ASSUMES NO LIABILITY WHATSOEVER, AND [REDACTED] DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF [REDACTED] PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO SUCH PRODUCTS OR ARISING FROM THIS DOCUMENT, FOR ANY PURPOSE, WHETHER SUCH PRODUCTS ARE OPERATED IN A SINGLE OR MULTIPLE FACILITY OR ARE USED IN CONNECTION WITH OTHER [REDACTED] PRODUCTS. [REDACTED] PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUPPORTING APPLICATIONS. [REDACTED] MAY MAKE CHANGES TO THE INFORMATION CONTAINED HEREIN WITHOUT NOTICE. [REDACTED] MAY HAVE MADE CHANGES TO THE INFORMATION PRESENTED SUBJECT MATTER. [REDACTED] DOES NOT ASSUME ANY LIABILITY OR RESPONSIBILITY FOR ANY ERRORS, OMISSIONS, OR OTHER DEFICIENCIES, WHETHER IN THIS DOCUMENT OR IN ANY OTHER DOCUMENTS, COPIES, OR OTHER INFORMATION. [REDACTED] RESERVES THESE RIGHTS. DESIGNERS MUST NOT RELY ON THE ABSENCE OR CHARACTERISTICS OF ANY FEATURES OR INSTRUCTIONS MARKED "RESERVED" OR "UNDEFINED". [REDACTED] RESERVES THESE RIGHTS FOR FUTURE DEFINITION AND SHALL HAVE NO RESPONSIBILITY WHATSOEVER FOR CONFLICTS OR INCOMPATIBILITIES ARISING FROM FUTURE CHANGES TO THEM. EXCEPT AS PERMITTED BY SUCH LICENSE, NO PART OF THIS DOCUMENT MAY BE REPRODUCED, STORED IN A RETRIEVAL SYSTEM, OR TRANSMITTED IN ANY FORM OR BY ANY MEANS WITHOUT THE EXPRESS WRITTEN CONSENT OF [REDACTED].



*How to get the architectural-level information **without** modifying the interface and hardware?*

Our approach

- Treat an SSD as a **black-box**
- Assume a **repeatable but unknown pattern**
- Inject I/O traffic to probe the device
- Observe the reactions of SSD in B/W and latencies
- Enumerate possible policies based on open documents*
- **Speculate** the internal structures



A general model



Intel® X25-E SSD

• Domain ✓

– A set of connected chips (how many?)

- 10x domains

Resource sharing:
e.g. channel, ECC eng.

– A basic unit of data block (how large?)

- 4KB

• Mapping ✓

– How chunks are allocated from their LBA to their PBA?

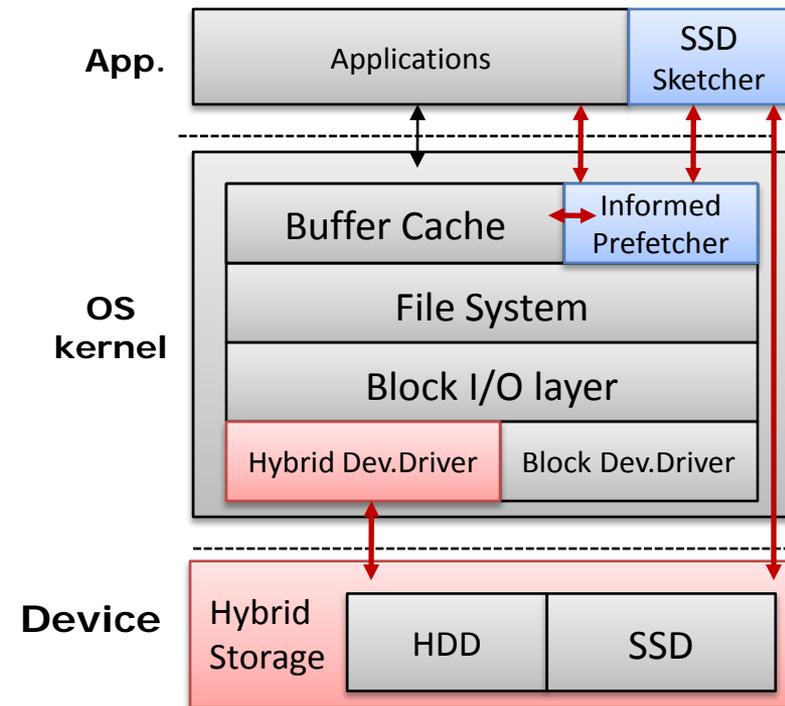
- Write-based

Interfacing to the system framework

- Chunk size
 - **Hybrid storage** – a basic unit for moving data across SSD/HDD
 - File System – align data allocation to chunks
 - I/O scheduler – avoid parallel accesses inside a chunk
- Number of domains
 - **Informed Prefetcher** – set a proper concurrency level
 - **Hybrid storage** – set a reasonable number of data migrating threads
 - I/O scheduler – decide how many requests should be released
- The mapping policy
 - I/O scheduler – insert a randomizer to permute block allocations
 - Applications – estimate physical data layout by observing writes
 - Virtual machine – **manipulate** physical data layout

Outline

- Introduction
- Sketching SSD internals
- **Hystor: A hybrid storage system**
- Exploiting Internal Parallelism
- Conclusion
- Future Work



Integrating SSD and HDD together



- Cache-based solutions
 - SSD – a secondary-level cache
 - HDD – the permanent storage
 - Cache replacement policy
- Limitations
 - Weak locality memory misses
 - Intensive write traffic
 - Non-trivial system changes
 - High-cost on-line replacement
 - Frequent on-access updates
 - 10-20x Larger SSD space

- Conquest [USENIX'02]
- SmartSaver [ISLPED'06]
- ReadyBoost [MS'06]
- TurboMemory [ToS'08]
- L2ARC [CACM'08]
- FlashCache [ISCA'08]
- other ...

Hystor: A cost-efficient hybrid storage*



- A small data set
 - **Semantically critical** – F/S metadata blocks
 - **Performance critical** – High-cost data blocks

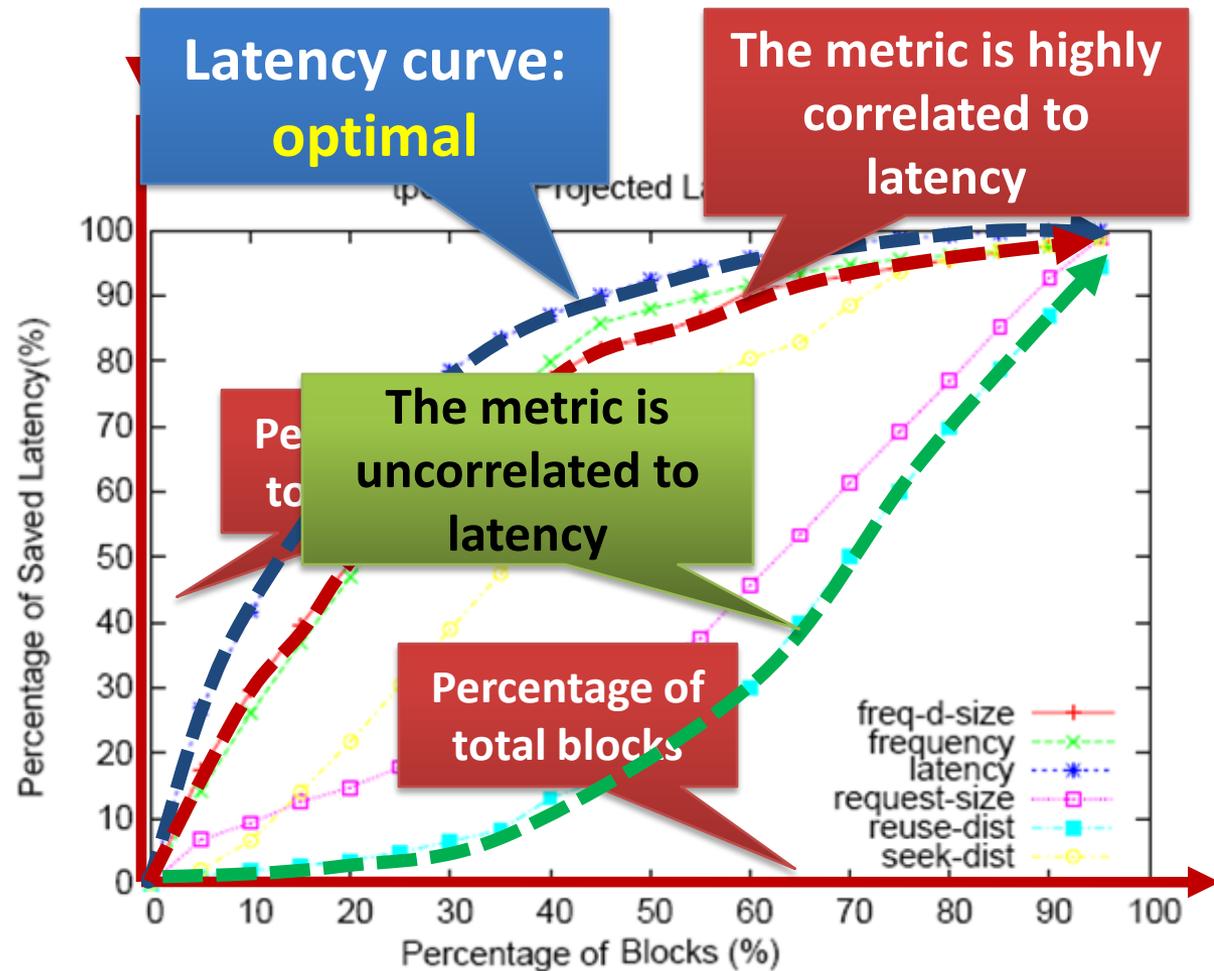
- A large data set
 - Low-priority data (e.g. movie files)

A prototype system at Intel® Labs for future storage system solution.

Identifying the high-cost data blocks

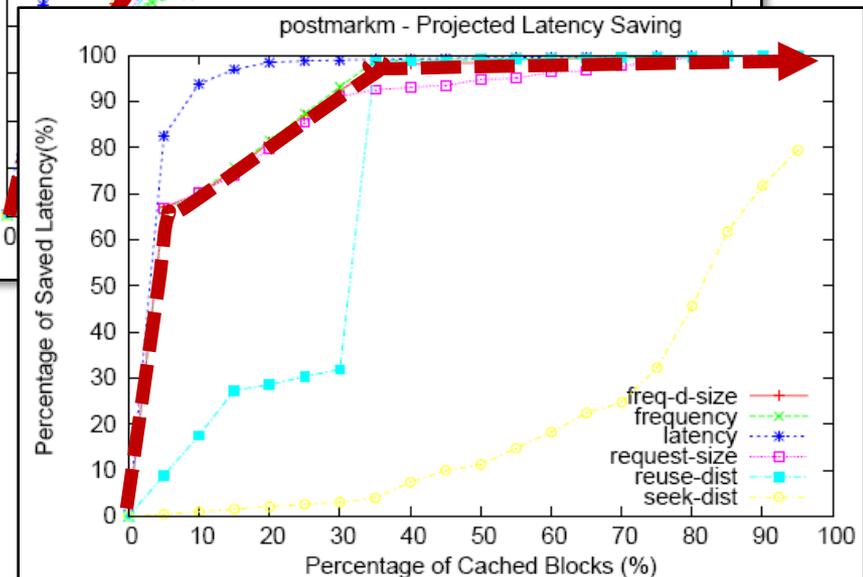
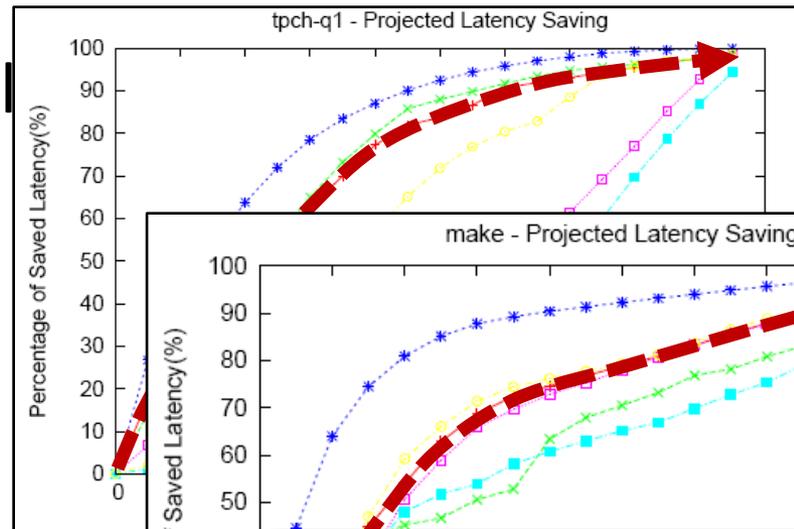
- A metric highly correlated to latency

- Latency (**optimal**)
- Frequency
- Request size
- Reuse distance
- Seek distance
- combinations



Identifying the high-cost data blocks

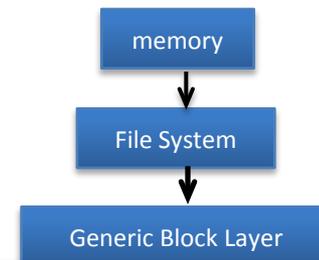
- A metric highly correlated with latency
 - Latency (**optimal**)
 - Frequency
 - Request size
 - Reuse distance
 - Seek distance
 - combinations
- Frequently used small blocks



*The best metric:
Frequency/Request size*

The prototype system of **Hystor***

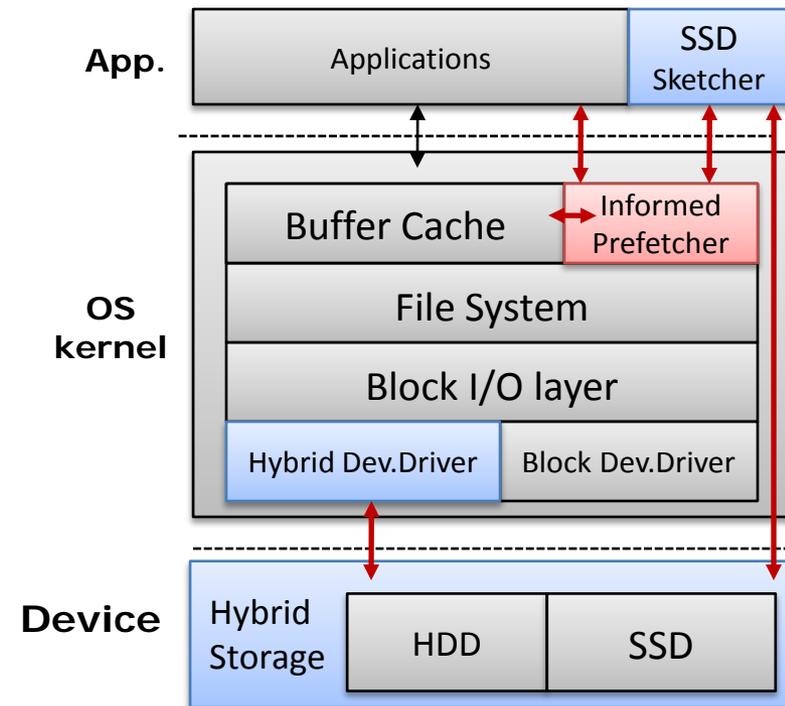
- Implementation
 - **Kernel module** in the kernel 2.6.25.8
 - Core code: 2,500 lines
 - Kernel-level monitor: 4,800 lines
 - 50+ lines in stock kernel
- A pseudo device driver
 - `/dev/mapper/hybrid`
- Inline Tracer
 - Intercepts I/O operations
- Monitor
 - Updates the block table
- Data Mover
 - Reorganizes data layout



Future plan: Prototyping a hardware hybrid storage system

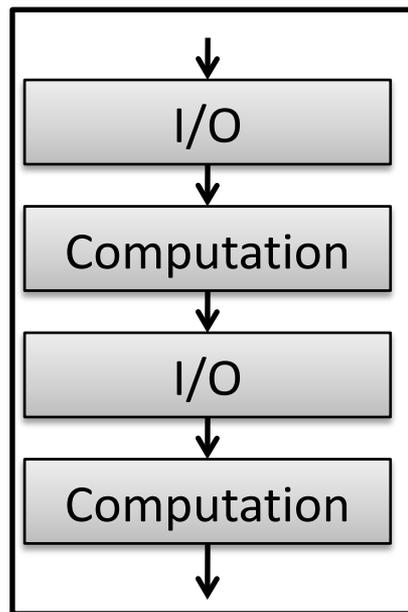
Outline

- Introduction
- Sketching SSD internals
- Hystor: A hybrid storage system
- **Exploiting Internal Parallelism**
- Conclusion
- Future Work

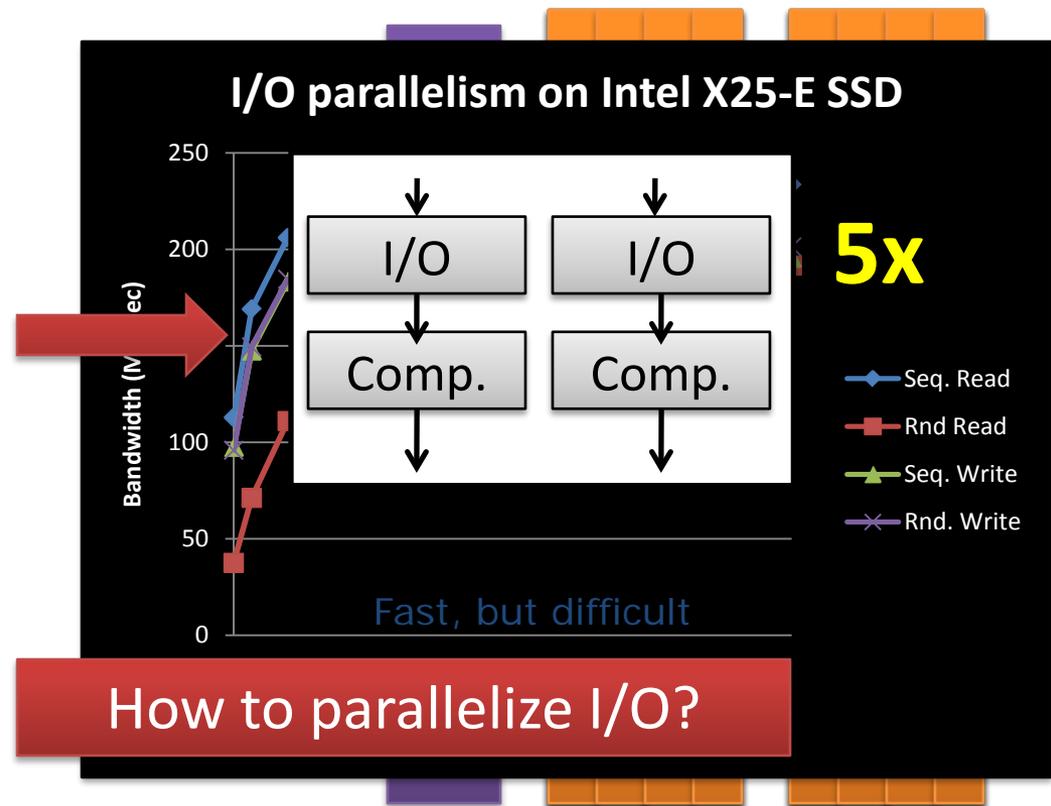


Internal parallelism – a hidden resource

- Internal Parallelism
 - An important hidden resource of SSD
 - I/O parallelism is the key to utilizing the idle resources



Serial I/O
slow



How to parallelize I/O operations?

- Initial attempt
 - Automatically generate parallelized I/O code
 - Heavily involved application redesign
 - Hardly be practical to rewrite all applications
- Alternative: **prefetching**
 - Leverage domain knowledge of applications
 - Automatically generate parallel prefetch I/O
 - Less speedup, more practical

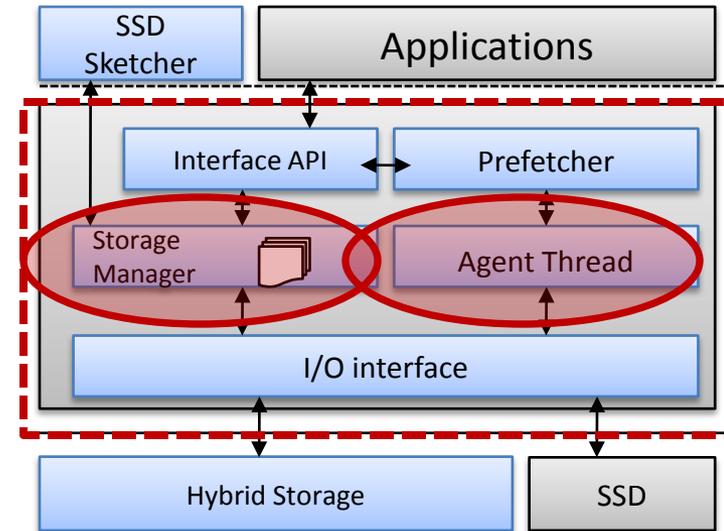
SSD-optimized Informed Prefetch (S.I.P.)

- Application

- Set a correlated data set
- Minimize changes to application

- Prefetcher

- An on-line kernel daemon thread
- Maintains correlated data information
- Leverages idle time to prefetch correlated data via parallel I/O
 - On-demand prefetching
 - On-access prefetching
- **Important:** maintain a proper concurrency level (**sketcher**)



Conclusion

- The emerging technology SSD arrives at the right time as we enter the data explosion era
- We have identified three major critical issues:
 - Affordability and limited capacity
 - Performance dynamics due to a non-transparent view of internal structures
 - Underutilizing rich and hidden storage resources
- To address these issues, we have designed and implemented a system framework with three major components
 - **SSD Sketcher** – detect internal structures (at the application level)
 - **Hystor** – a hybrid storage management system (in OS kernel and I/O device)
 - **S.I.P.** – an enhancement to help user exploit internal parallelism (in OS kernel)
 - High effectiveness is shown by extensive experiments
- Intel® Lab is prototyping it at both software and device level as a consideration for a future storage system product.



Related Publications

- SSD related papers
 - [**SIGMETRICS'09**] "*Understanding Intrinsic Characteristics and System Implications of Flash Memory based Solid State Drives*"
 - [**ISLPED'06**] "*SmartSaver: Turning Flash Drive into a Disk Energy Saver for Mobile Computers*"
- Other papers on memory and storage systems
 - [**USENIX'07**] "*DiskSeen: Exploiting Disk Layout and Access History to Enhance I/O Prefetch*"
 - [**FAST'05**] "*DULO: An Effective Buffer Cache Management Scheme to Exploit Both Temporal and Spatial Localities*"
 - [**USENIX'05**] "*CLOCK-Pro: An Effective Improvement of the CLOCK Replacement*"

Thank You !

Xiaodong Zhang

✉: zhang@cse.ohio-state.edu

<http://www.cse.ohio-state.edu/~zhang>