http://clyde.itu.dk
A project funded by the Danish Council for Independent Research

CLyDE Project:
Lessons Learned and Perspectives

Philippe Bonnet – phbo@itu.dk
IT University of Copenhagen

Joint work with Luc Bouganim (INRIA), Niv Dayan (ITU), Matias Bjørling (ITU), Jesper Madsen (ITU), Martin Svendsen (ITU)

In Collaboration with Jens Axboe (Facebook), David Nellans (Nvidia), Zvonimir Bandic (HGST), Qingbo Wang (HGST), Aviad Zuck (Tel Aviv Univ.),
Michael Wei, Steve Swanson (UCSD), Dennis Shasha (NYU)

Database Systems / Computer Systems Landscape

Appliance Small Cluster (1000s)	Relational DBMS (transactional)
Rack-scale (10000+)	Key-value store In-memory Database

THE DATACENTER AS A COMPUTER

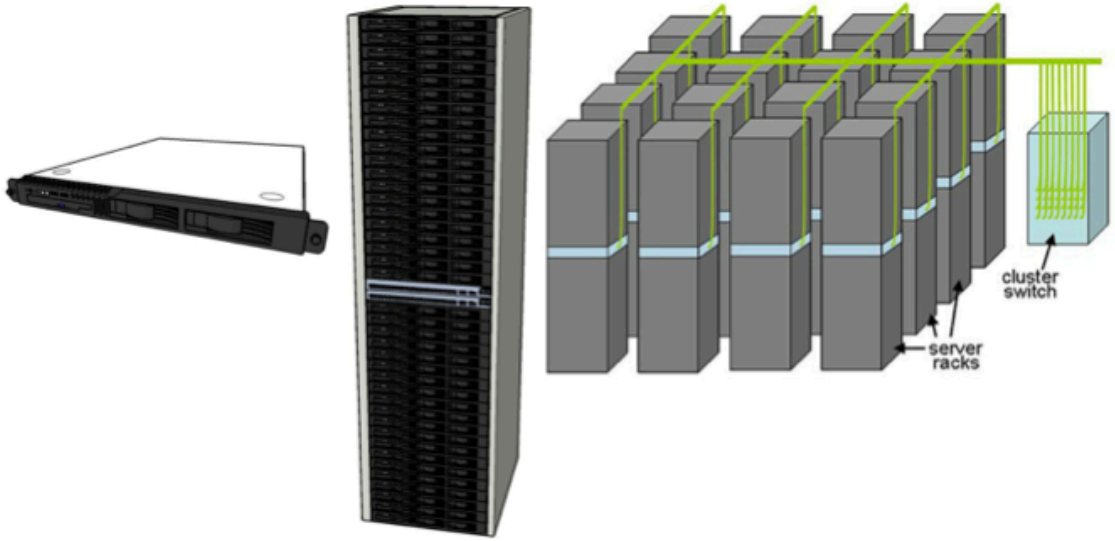
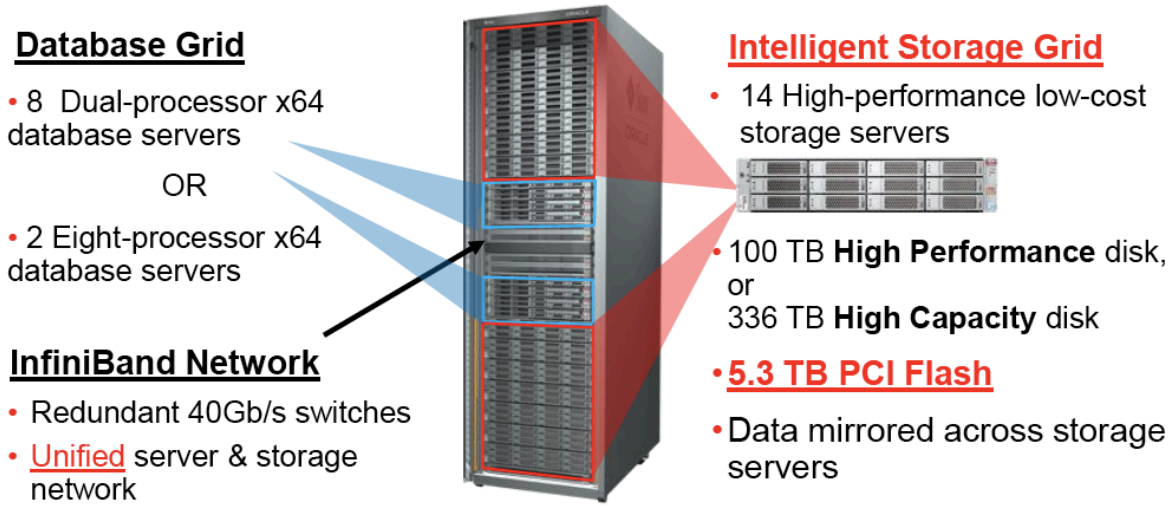


FIGURE 1.1: Typical elements in warehouse-scale systems: 1U server (left), 7′ rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/router (right).

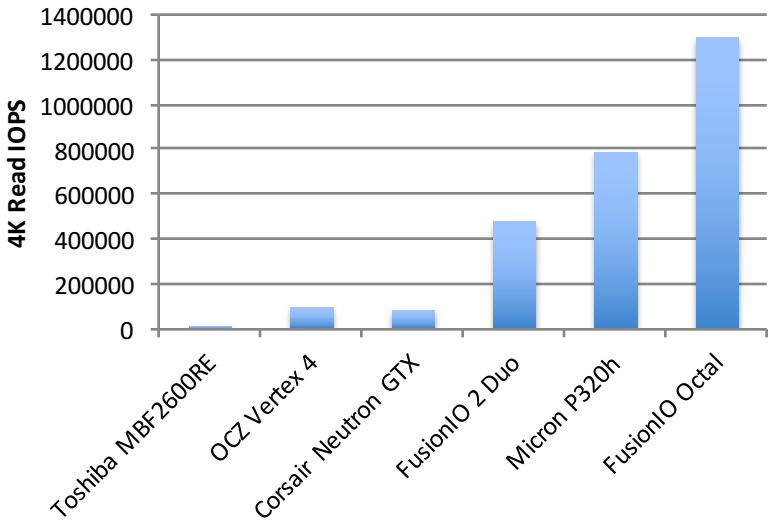
Source: http://www.morganclaypool.com/doi/abs/10.2200/S00193ED1V01Y200905CAC006



Source: http://www.oracle.com/us/products/database/exadata/overview/index.html

The Advent of SSDs

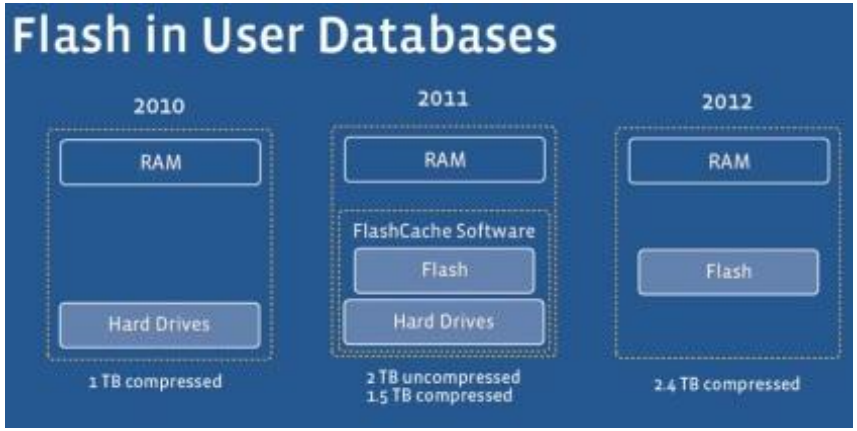
Balanced Systems



High throughput/Low latency reduces the gap between RAM and IO performance

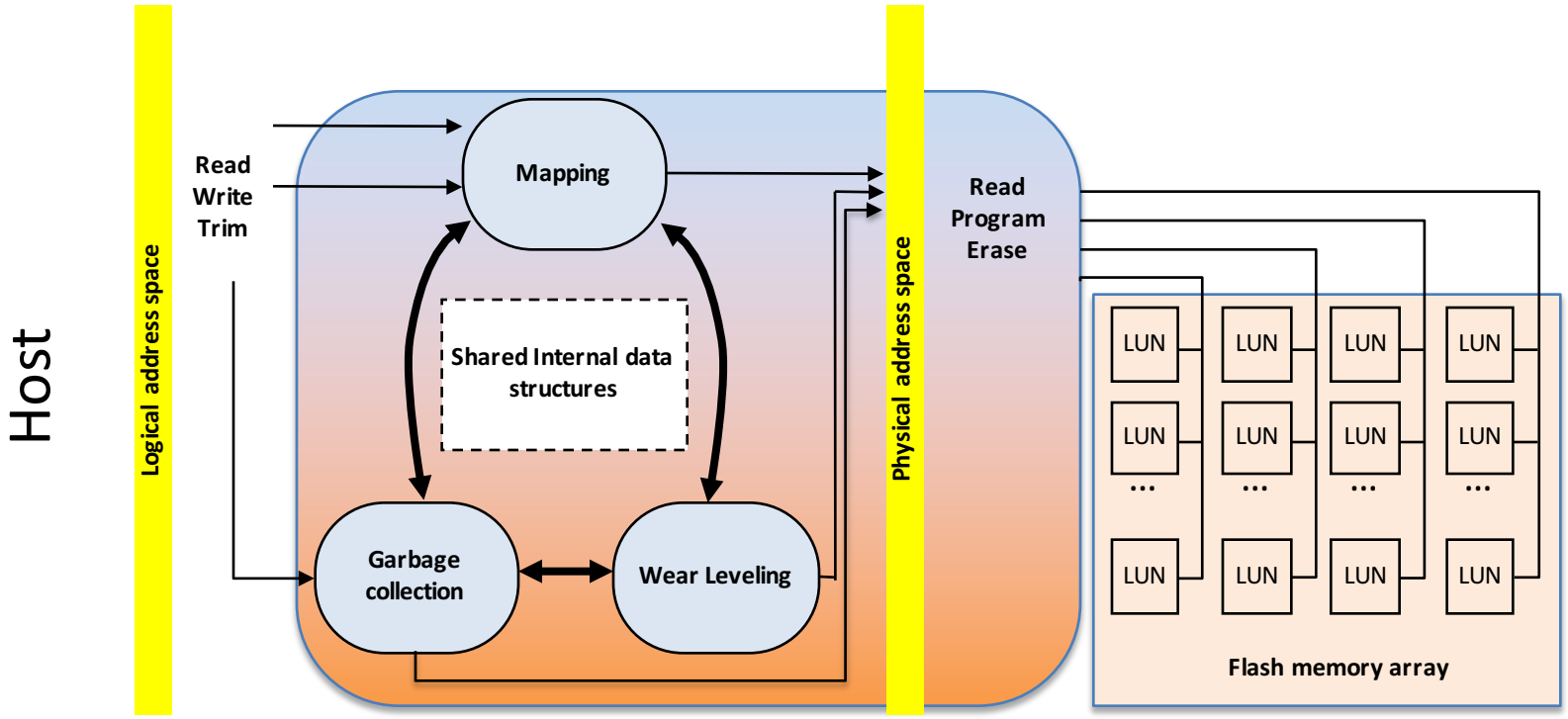
Energy Efficient Storage

Fewer Watts/TB drives higher data density on the cloud (e.g., OCZ vertex 4: 5W/TB; Toshiba MBF: 22W/TB)



Jason Taylor, Flash Memory Summit, 08/13

Solid State Drives (SSDs)

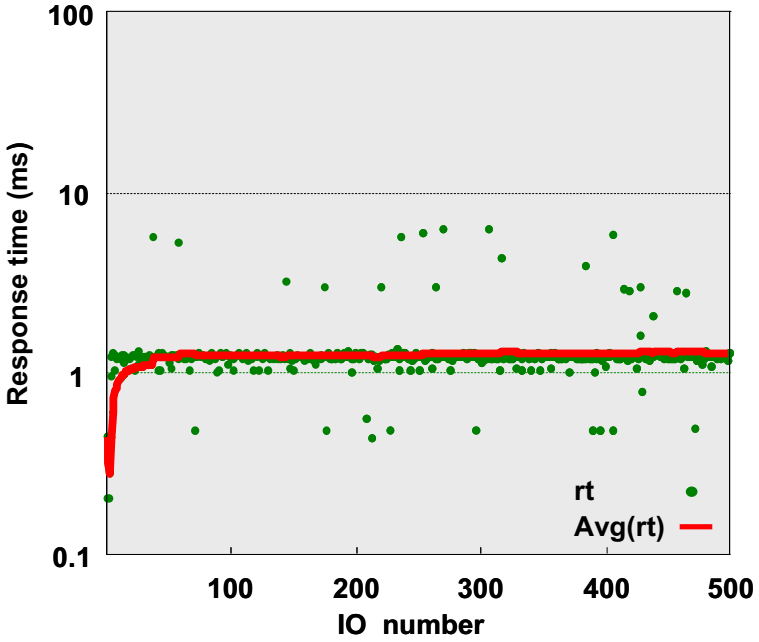


Flash Translation Layer (FTL)
Implemented on SSD controller or SSD driver

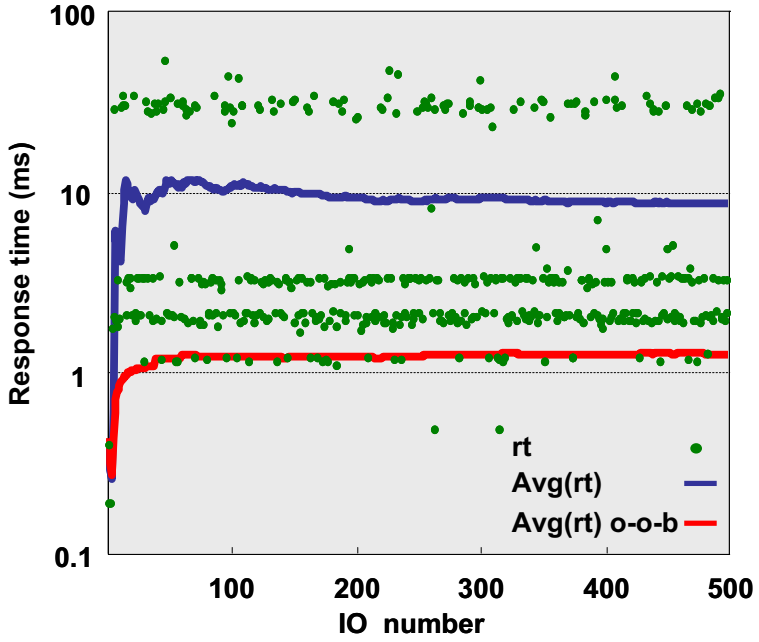
CIDR 2009 Best Paper

Methodology (1): Device state

- Measuring Samsung SSD RW performance
  - Out-of-the-box … and after filling the device!!! (similar behavior on Intel SSD)



Random Writes – Samsung SSD
Out of the box



Random Writes – Samsung SSD
After filling the device

The Challenges

AWS I2.8xlarge instance: 32 vCPU, 244GiB RAM, 8x800 GB SSD, $6.82/h

SSD I/O Performance

To ensure the best IOPS performance from your I2 instance, we recommend that you use the most recent version of the Amazon Linux AMI, or another Linux AMI with kernel version 3.8 or later. If you use a Linux AMI with kernel version 3.8 or later and utilize all the SSD-based instance store volumes available to the instance, you can get at least the minimum random IOPS (4,096 byte block size) listed in the following table. Otherwise, you'll get lower IOPS performance than what is shown in the table.

Instance Size	Read IOPS	First Write IOPS
i2.xlarge	35,000	35,000
i2.2xlarge	75,000	75,000
i2.4xlarge	175,000	155,000
i2.8xlarge	365,000	315,000

As you fill the SSD-based instance storage for your instance, the number of write IOPS that you can achieve decreases. This is due to the extra work the SSD controller must do to find available space, rewrite existing data, and erase unused space so that it can be rewritten. This process of garbage collection results in internal write amplification to the SSD, expressed as the ratio of SSD write operations to user write operations. This decrease in performance is even larger if the write operations are not in multiples of 4,096 bytes or not aligned to a 4,096-byte boundary. If you write a smaller amount of bytes or bytes that are not aligned, the SSD controller must read the surrounding data and store the result in a new location. This pattern results in significantly increased write amplification, increased latency, and dramatically reduced I/O performance.

SSD controllers can use several strategies to reduce the impact of write amplification. One such strategy is to reserve space in the SSD instance storage so that the controller can more efficiently manage the space available for write operations. This is called *over-provisioning*. The SSD-based instance store volumes provided to an I2 instance don't have any space reserved for over-provisioning. To reduce write amplification, you should leave 10% of the volume unpartitioned so that the SSD controller can use it for over-provisioning. This decreases the storage that you can use, but increases performance.

You can also use the TRIM command to notify the SSD controller whenever you no longer need data that you've written. This provides the controller with more free space, which can reduce write amplification and increase performance. For more information about using TRIM commands, see the documentation for the operating system for your instance.

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/i2-instances.html

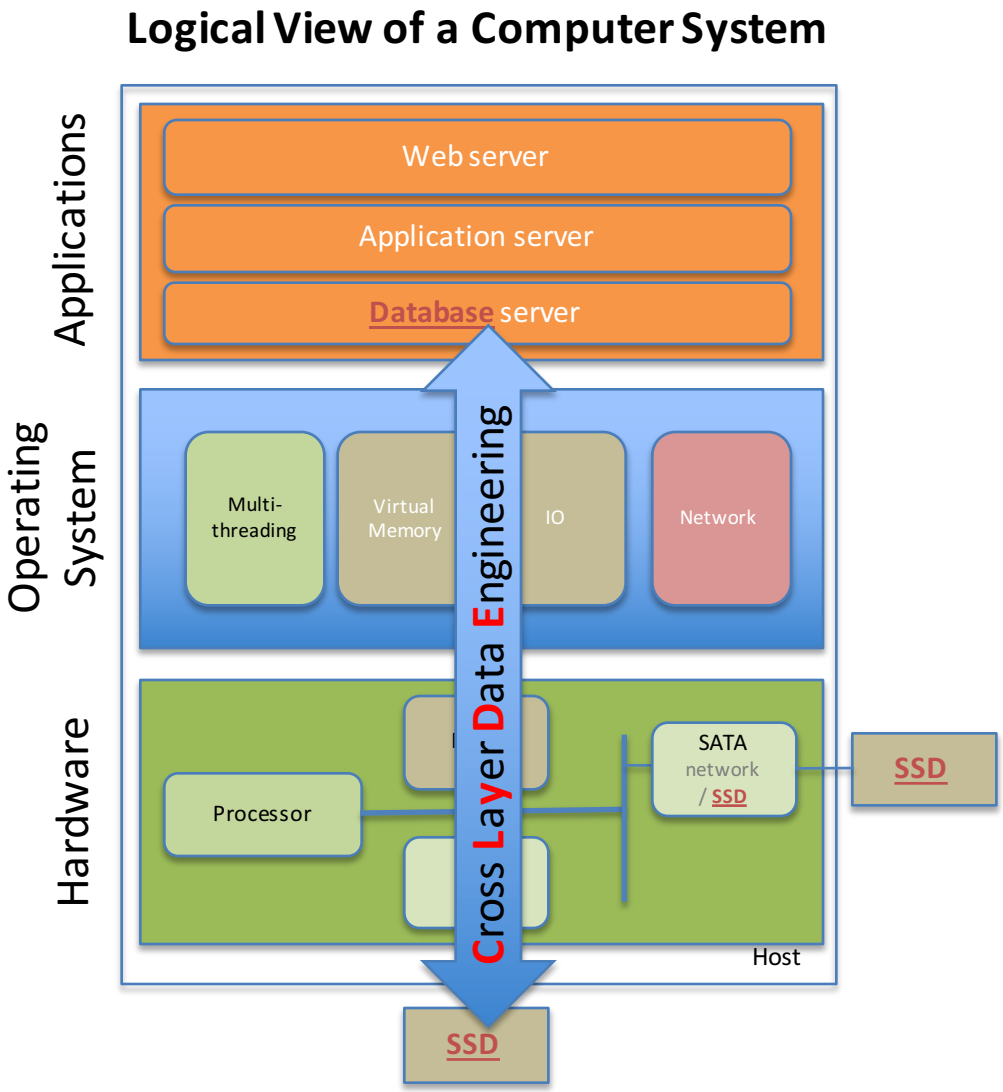
Cloud provider
How to design and characterize the IO service?

System designer
What is the impact of SSDs on software design, in particular on database systems?

Focus of CLyDE

Database Administrator
How to tune for performance?

The CLyDE Hypothesis



"This project is based on the insight that the strict layering that has been so successful for designing database systems on top of magnetic disks is no longer applicable to flash devices. In other words, the complexity of flash devices cannot be abstracted away if it results in unpredictable and suboptimal performance."

SSD moving from memory to communication abstractions.

The CLyDE Goals

2011

2014

"The goal of this project is to explore how *flash devices, operating system and database system* can be designed together to improve overall performance. Such a **co-design** is particularly important for the next generation database appliances (e.g., Oracle's Exadata or Neteeza's TwinFin), or cloud-based relational database systems (e.g., Microsoft SQL Azure) for which well-suited flash components must be specified. More generally, our goal is to influence the evolution of flash devices and commodity database systems for the benefit of data intensive applications. "

EMC

Samsung

Micron

Hitachi

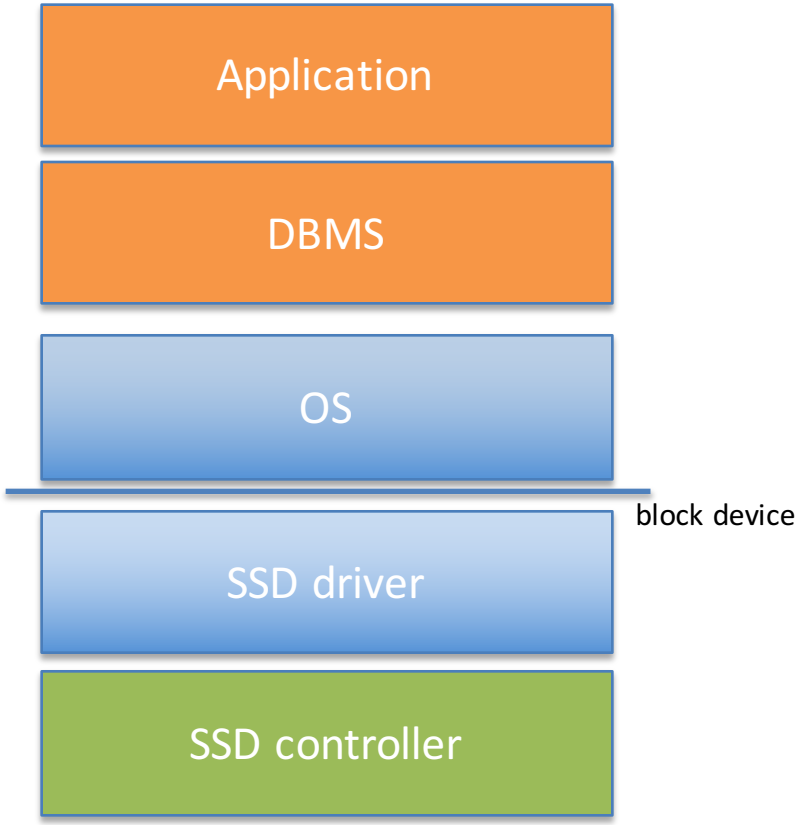
Fusion-IO

Rel. Work: UCSD, MIT, CMU, Princeton, Baidu, U.Wisconsin.

How to design for SSDs?

Business as usual

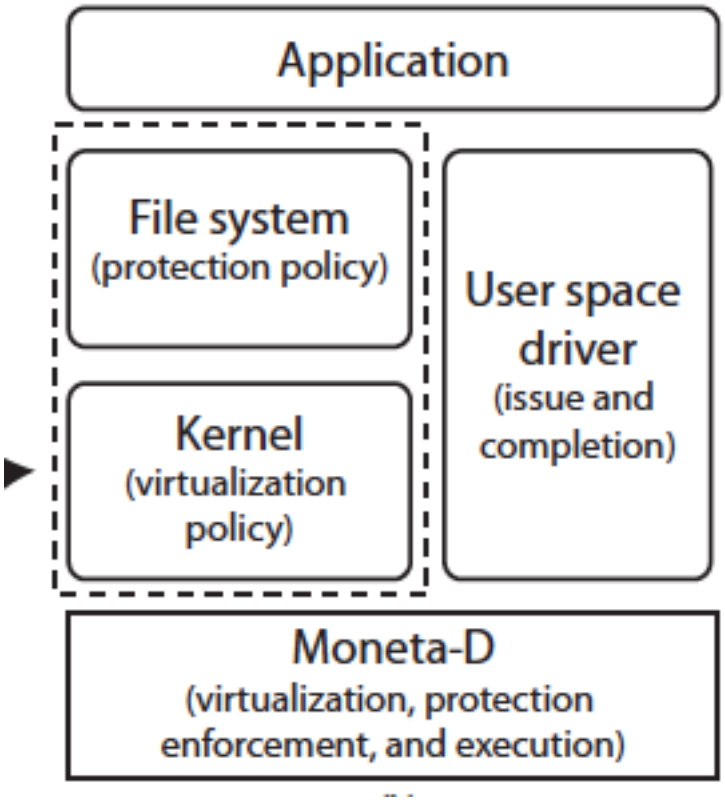
- Layered design
- SSD as block device
- SSD as a black box
- Performance model for SSD to drive design decisions



Lessons learnt [CIDR'09][Sigmod'10][DEB'10]:

- Performance varies across SSDs, in time for a SSD (with firmware updates, depending on IO history)
- Performance varies for a given IO pattern with target size, with concurrency, with submission rate

How to design for SSDs?



Moneta-D [asplos12]
Caulfield et al., UCSD

New approaches

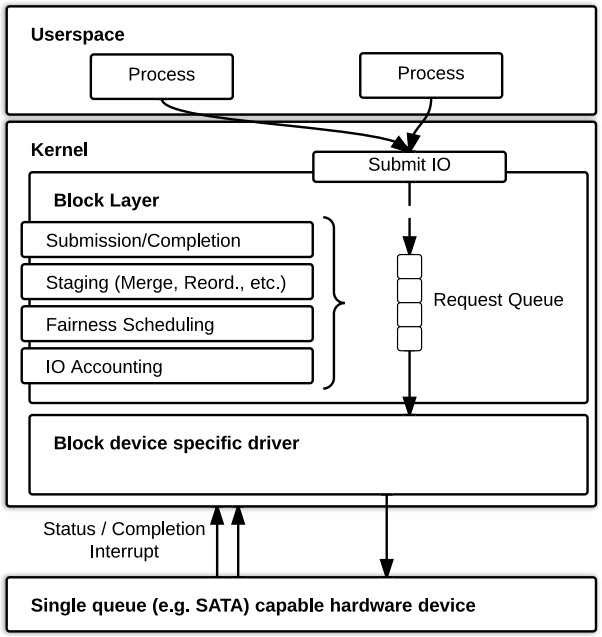
- Layers revisited:
  - OS bypassed (e.g., Moneta-D [asplos12])
  - Avoid duplication of work (e.g., from ARIES to MARS [SOSP13])
- Cross-layer optimizations
  - Trim command [HPCA11]
  - Extending fadvise [NVM12]
- New SSD interface:
  - Communication abstraction to replace memory abstraction
  - More complex address space

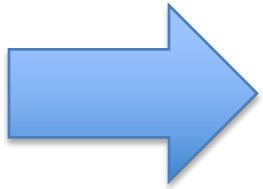
CLyDE Project

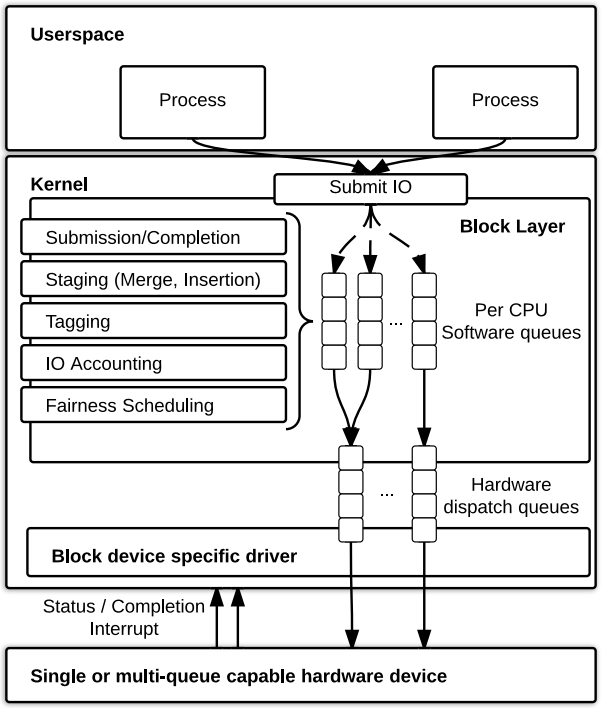
- Exploring the design space for Host-SSD co-design:
  - EagleTree [Niv Dayan]
    - Simulated SSD/OS/apps for broad exploration of design space (discrete event simulation)
    - Insights about GC, DB indexing
    - http://github.com/ClydeProjects/EagleTree
  - LightNVM [Matias Bjørling]
    - Host-side SSD management to experiment with actual OS/apps (wall time clock)
    - Linux support for Open-channel SSDs
    - http://github.com/ MatiasBjorling/LightNVM

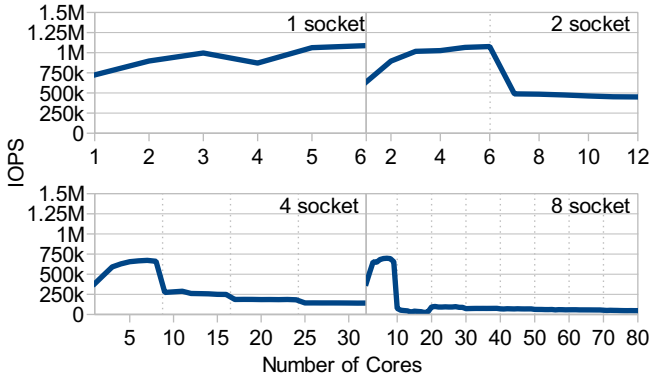
Systor 2013

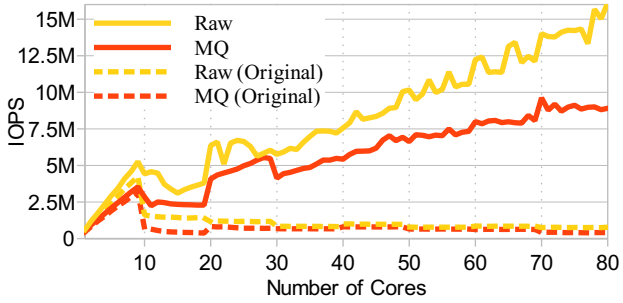
Linux Block Layer: Introducing Multiqueue SSD Access on Multi-core Systems











Matias Bjørling, Jens Axboe, David Nellans, Philippe Bonnet

Joe Williams
@williamsjoe

Follow

Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems (coming in linux 3.13) kernel.dk/blk-mq.pdf

Reply Retweet Favorite More

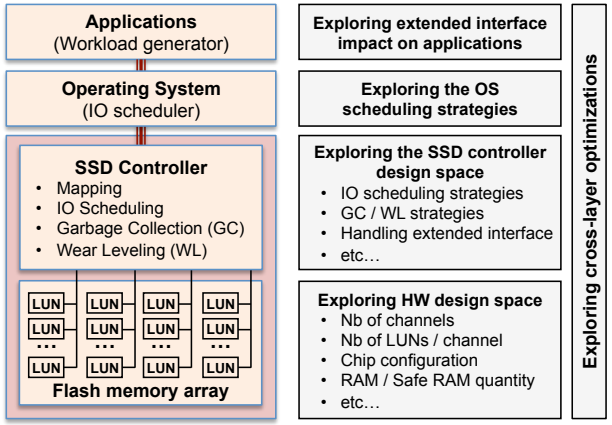
RETWEETS 87 FAVORITES 72

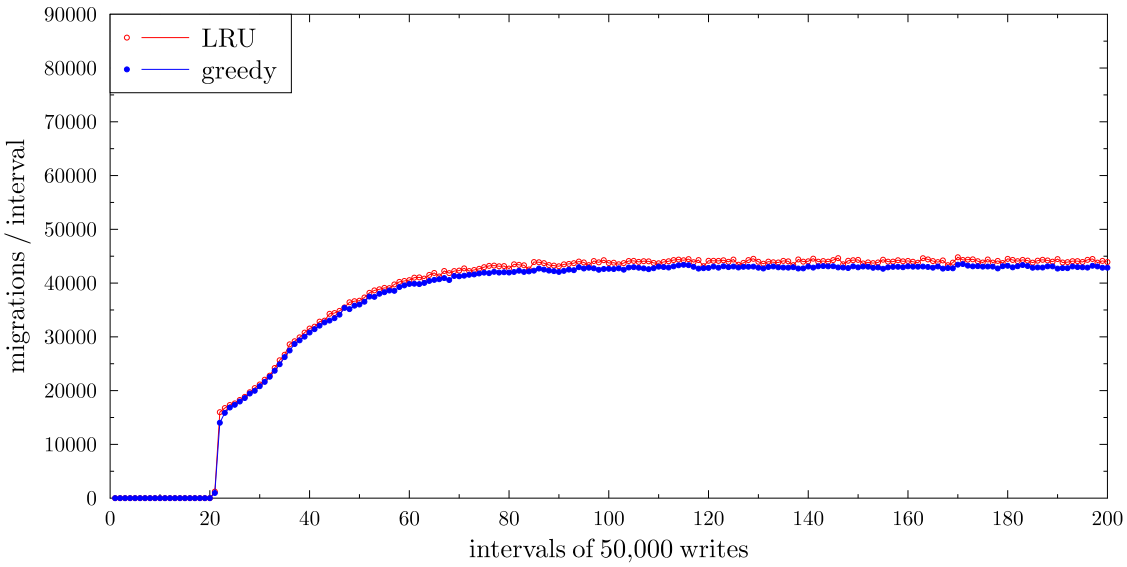
8:40 AM - 16 Nov 2013

https://github.com/ClydeProjects/EagleTree

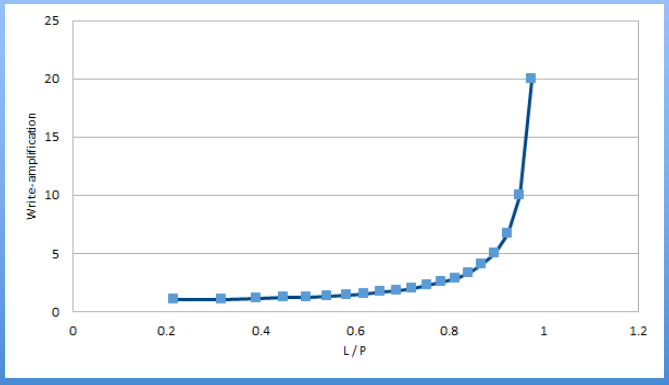
New Insights about GC policies

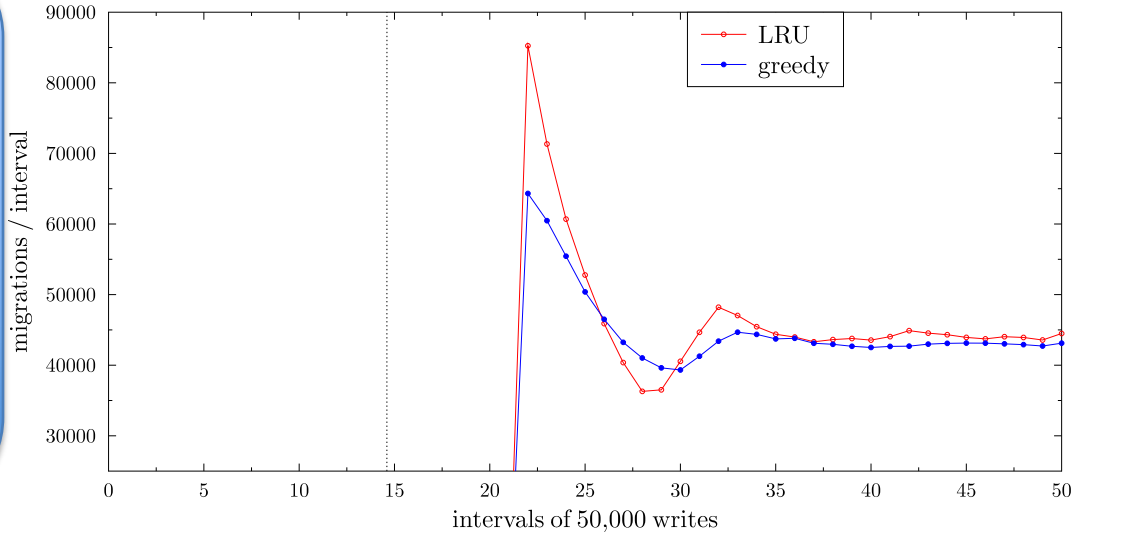
EagleTree





Closed-form equation for write-amplification

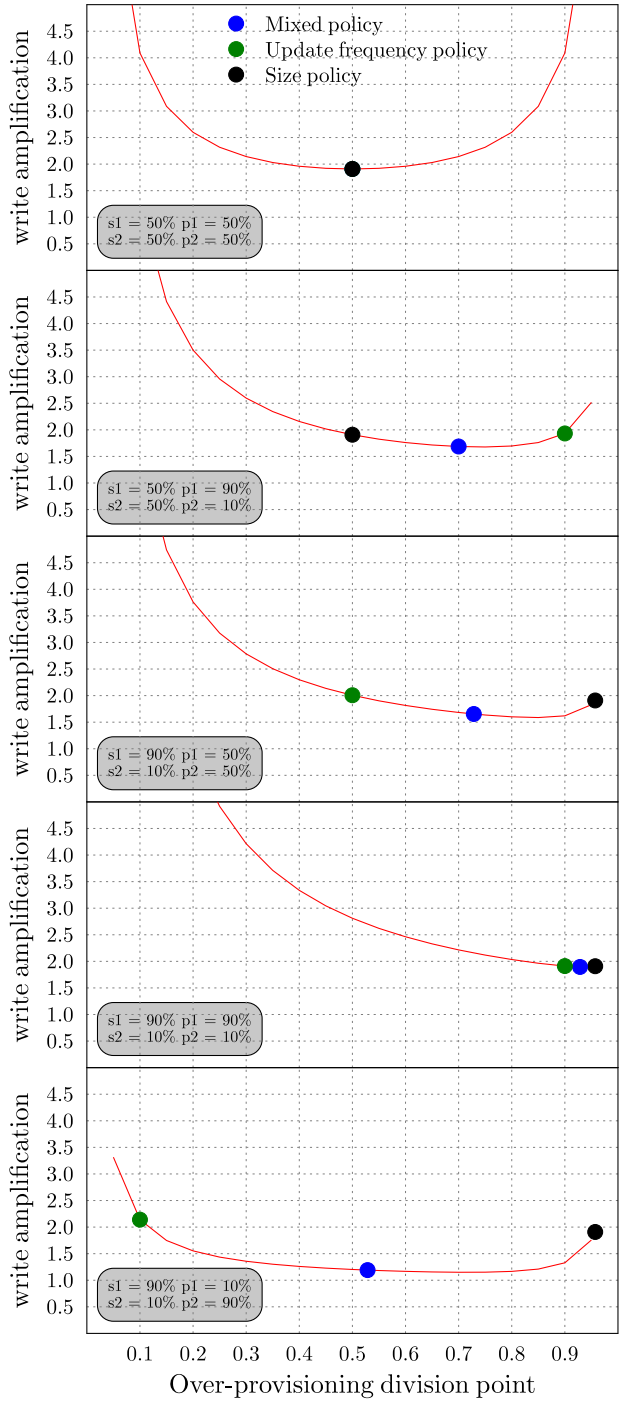


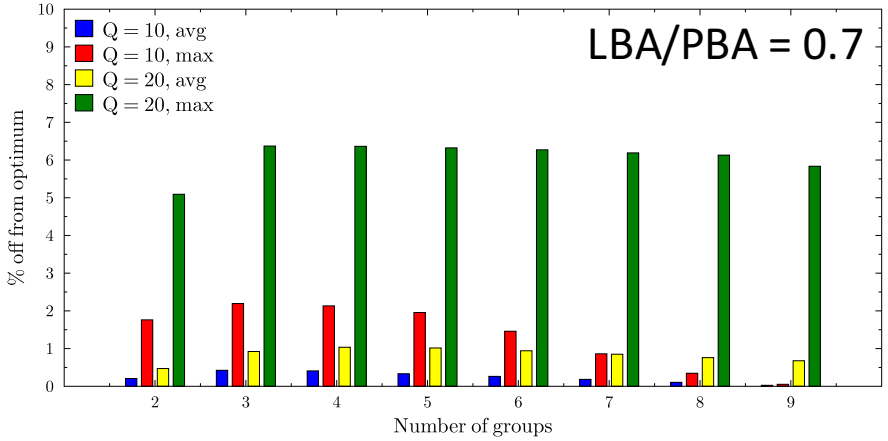


More Insights about GC Policies

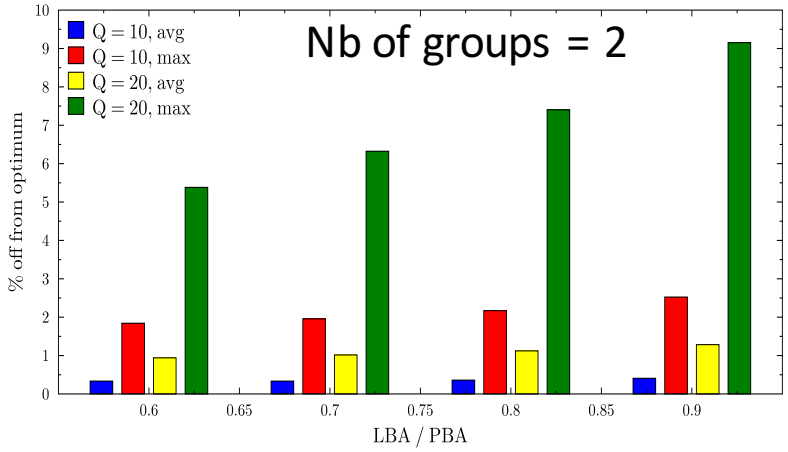
- K-modal workload:
  - Data grouped based on update frequency
  - Previous work:
    - Each flash block is dedicated to a single group
  - Questions:
    1. How to partition over-provisioned blocks across groups?
    2. How to deal with changing update frequencies?

Partitioning Over-provisioned Blocks





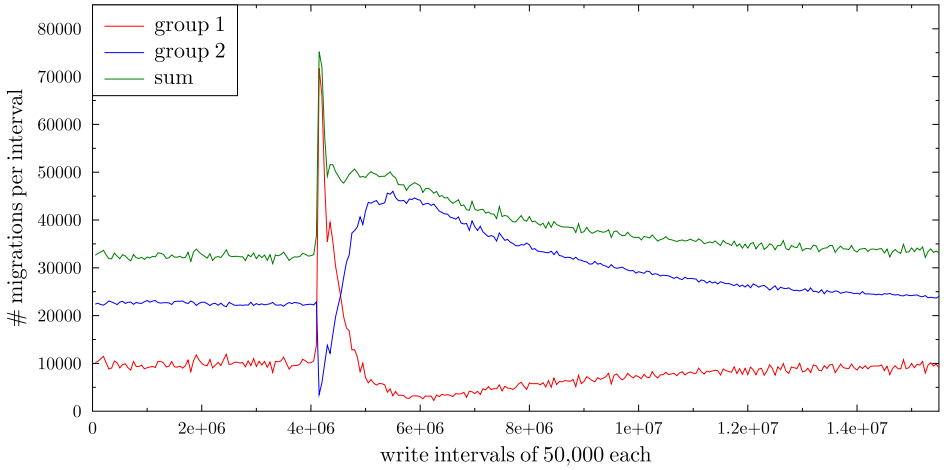
Q: number of distinct update frequencies in the workload

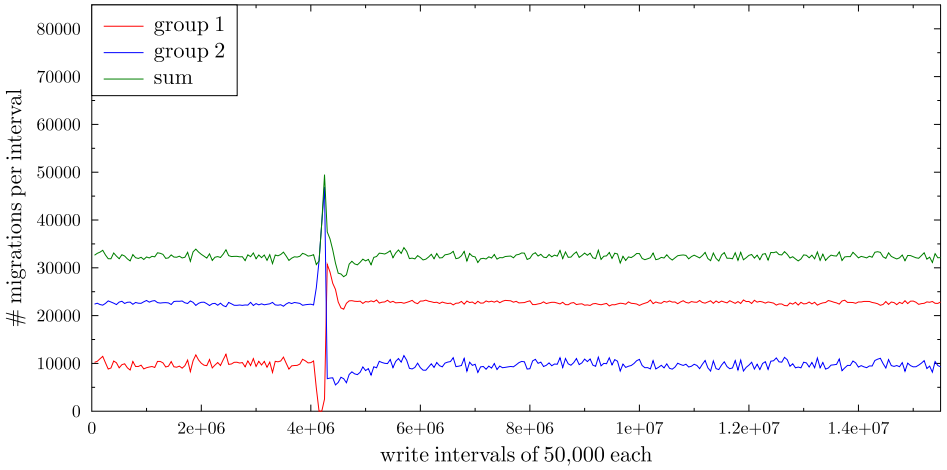


Adapting to Changing Workloads

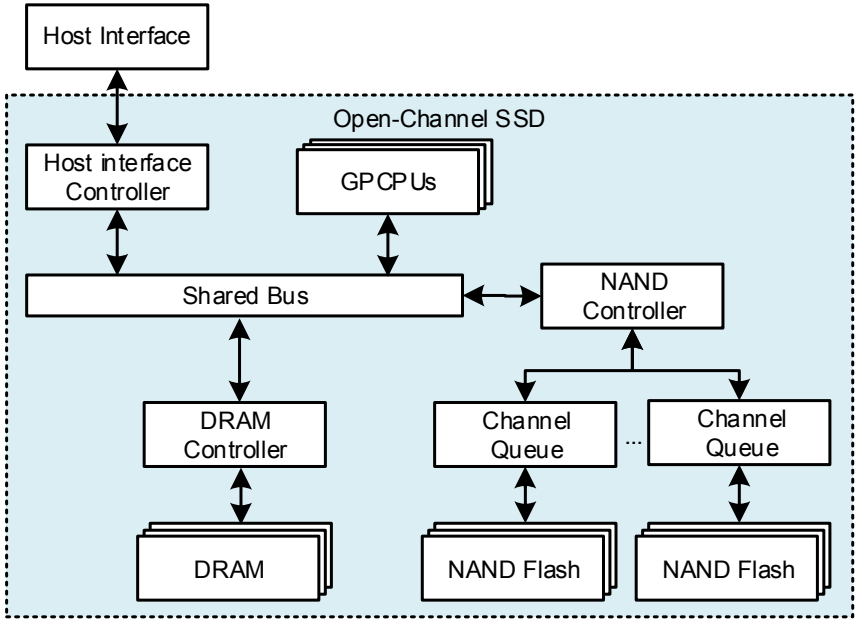
WOLF:

- dynamically measures update frequency
- adapts the number of groups
  - Too few/many groups harm GC efficiency
- triggers garbage-collection aggressively to re-distribute over-provisioned space across groups (from cooling to heating groups)



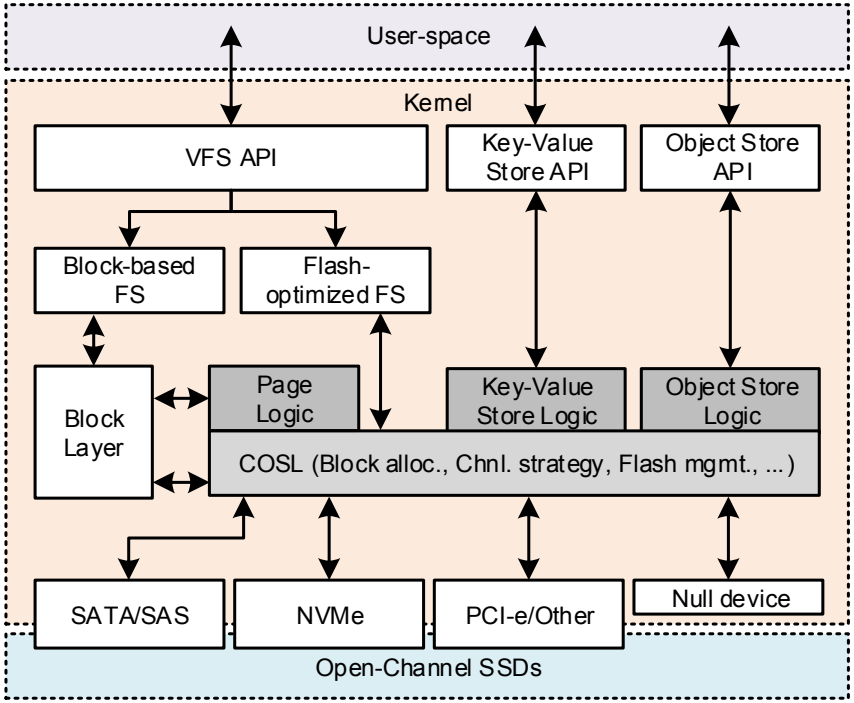


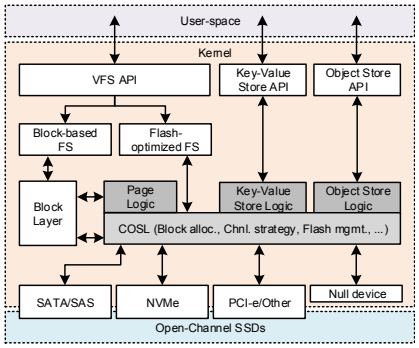
Linux Abstractions for Open-Channel SSDs



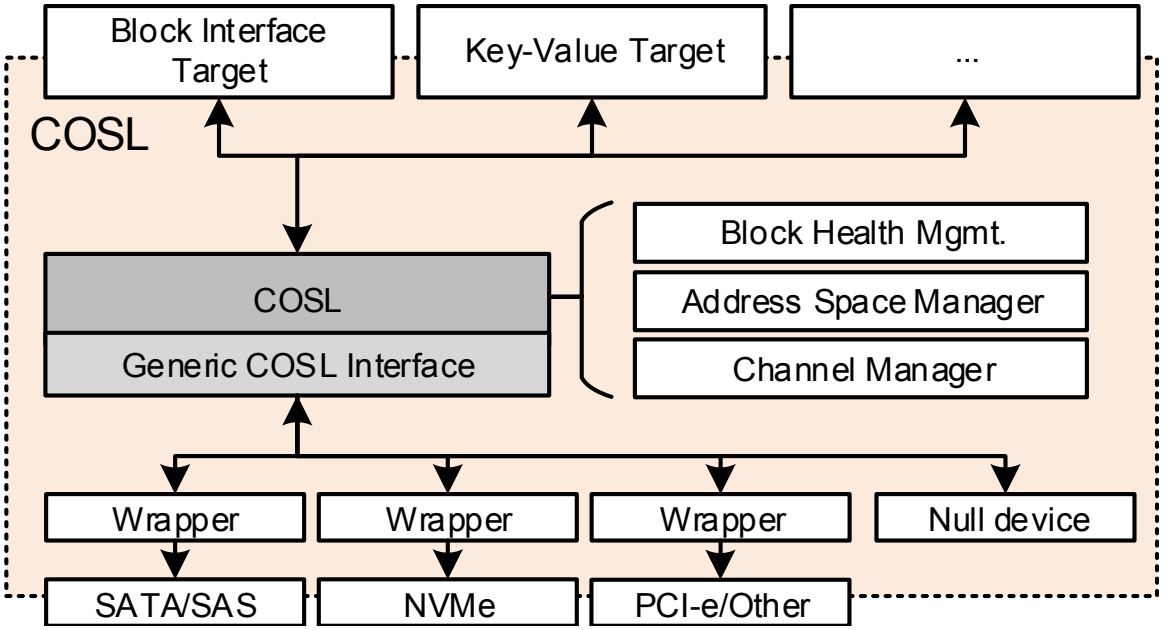
- Extensible
  - Single level of mapping between Applications and physical storage
- Modular
  - SSD Management components should be replaceable
- Low overhead
  - Host-side SSD management should not get in the way of performance, still provide consistency, durability

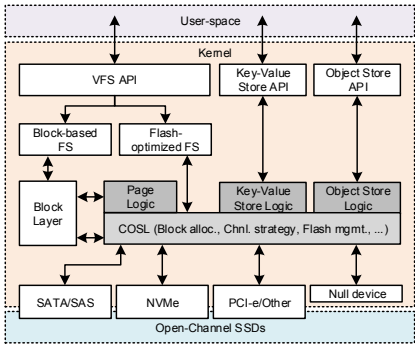
LightNVM Design (1/3)



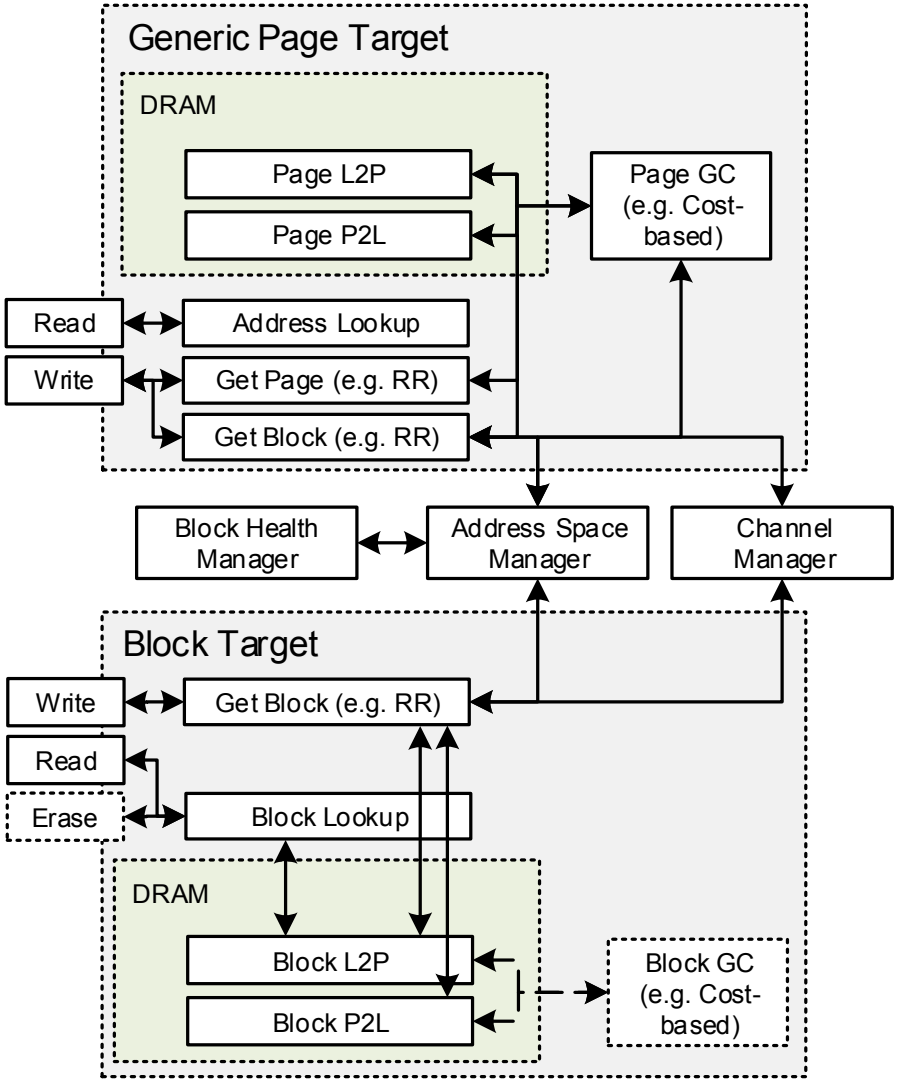


LightNVM Design (2/3)

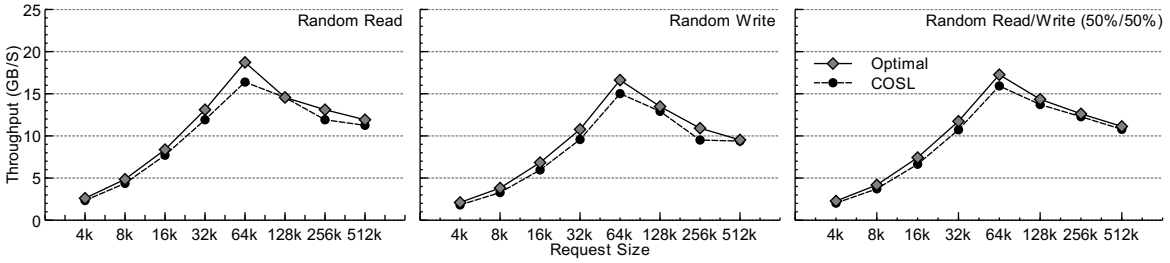


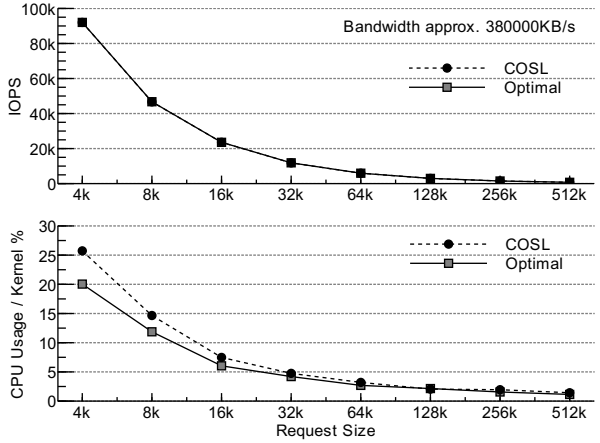


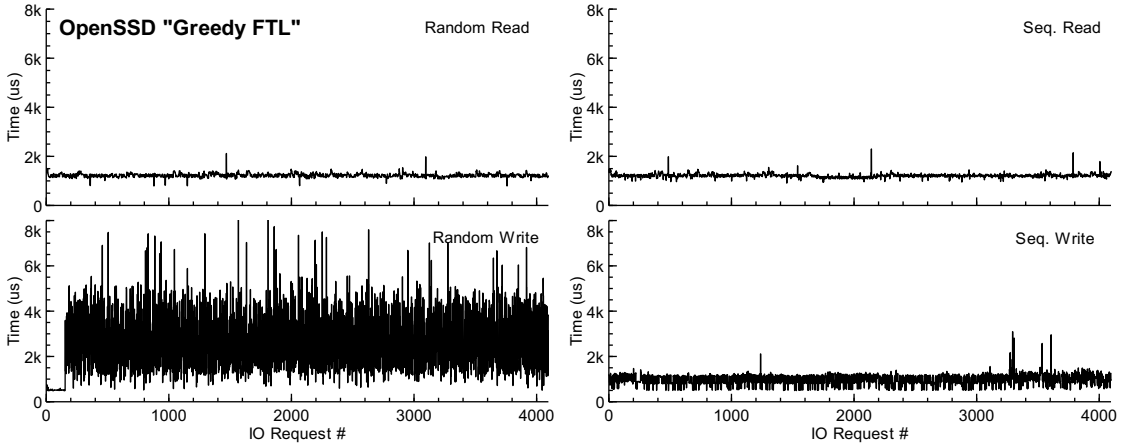
LightNVM Design (3/3)

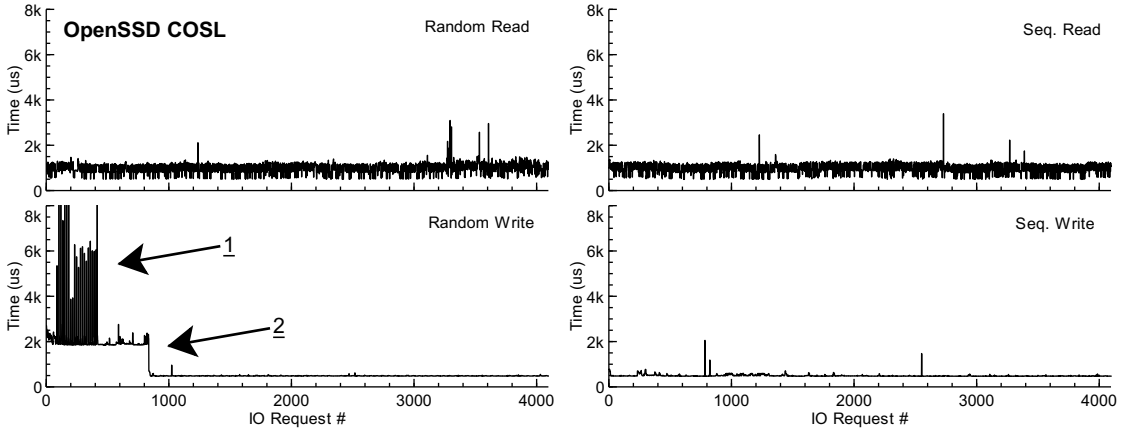


LightNVM Performance









LightNVM: Next Steps

- More research ...
  - New Targets (B Forest, ...)
  - New Open-channel SSDs (Lightstor, ...)
  - SSD Management variants (WOLF)
- Open-source contributions
  - Linux
    - Linux Plumber Workshop on Open-channel SSDs
    - Lower RAM occupation, better performance
  - Lightstor foundation
    - Joint work with GS Madhususan at IIT Chennai
    - Contribution to RapidIO (ITU is now a member)
- http://github.com/ MatiasBjorling/LightNVM

B-Forest

- Tree Logarithmic method for database indexing
  - Writes grouped in chunks, updated at the same time
  - Multiway merges to leverage SSD parallelism
  - Bloom filters to speed up lookups

A variant of the block target in LightNVM

WOLF

- Garbage Collection
  - Flash blocks partitionned in groups based on update frequency
  - Fixed overprovisioning per group (and local GC per group) dominates global over-provisioning (and global GC)
  - WOLF block manager adapts to changing update frequencies
    - Aggressive GC for cooling groups to re-distribute over-provisioning across groups.

A variant of the LightNVM Adress Space Manager and page-based GC

Dissemination

Publications

- Niv Dayan, Martin Kjær Svendsen, Matias Bjørling, Philippe Bonnet, Luc Bouganim. *EagleTree: Exploring the Design Space of SSD-Based Algorithms*. VLDB 2012.
- Matias Bjørling, Philippe Bonnet, Luc Bouganim, Niv Dayan. *The Necessary Death of the Block Device Interface.* CIDR 2013.
- Matias Bjørling, Jens Axboe, David Nellans, Philippe Bonnet. Linux Block IO: Introducing Multi-Queue SSD Access on Multicore Systems. Systor 2013.
- Michael Wei, Matias Bjørling, Philippe Bonnet, Steven Swanson. I/O Speculation in the Microsecond Era. Usenix ATC 2014.

Submitted:

- Niv Dayan, Philippe Bonnet, Dennis Shasha. *Blooming Logarithmic B Forests: A Fast Access Method for Solid State Drives.*
- Matias Bjørling, Jesper Madsen, Philippe Bonnet. LightNVM: Operating System Abstractions for Open-Channel Solid State Drives.
- Niv Dayan, Luc Bouganim, Philippe Bonnet. *WOLF: Improving SSD Performance by Exploiting Update Frequencies for Dynamically Changing Workloads*

Talks

- Academic
  - Conferences: VLDB (8/12, 8/13), CIDR (1/13), Systor (7/13)
  - Visits: EPFL (2/14), INRIA Montpellier (5/14)
  - Stay abroad: UCSD, NYU
- Industrial
  - Linux Venues (3/14, 10/14)
  - Hitachi (4/14), Micron (4/14), EMC (9/14)

Strengthening Danish Research

- We are collaborating/competing with US universities
  - US: PhD takes 5 years
  - DK: PhD + Postdoc
- We are collaborating with international companies/consortium
  - Open-source contributions (Linux)
  - Keeping jobs in DK: internship, consulting anchored around research projects

New Challenges

- Emerging Byte-addressable Non Volatile Memories (e.g., PCM)
- SSDs are entering the microsecond era.
- SSDs and persistent memories require a profound redesign of system software.

- With Persistent Memories, persistence is no longer tied to secondary storage.
  - How can the OS handle Persistent Memories:
    (1) By extending virtual memory
    (2) By providing a block device interface for such memories
    (3) By designing new file systems tailored to their characteristics

Energy-Proportional Transaction Management

- A proposal: back to the good-old single-level store:
  - durability – how/when is data manipulated in the virtual address space made durable?
  - concurrency – how to design data structures that can perform efficiently when accessed concurrently
  - security – how to enforce access control but also integrity for a given portion of memory or storage?

- An approach:
  - LightNVM common services extended to Persistent Memories
  - Transactional VMM as LightNVM target

- Now with energy proportionality as a design goal
  - Requires new programming models!