BLOCK MANAGEMENT IN SOLID-STATE DEVICES

Abhishek Rajimwale (University of Wisconsin-Madison)
Vijayan Prabhakaran (Microsoft Research)
John D Davis (Microsoft Research)

Existing storage stack

- Storage stack has remained static
  - Mechanical disk drives for decades
  - Narrow block interface existing for years (ATA, SCSI)
  - No information flow except block reads/writes
- File systems make assumptions about devices
  - Sequential access much faster than random access
  - Little or no background activity
- Assumptions true for disk drives
- **What if the underlying device changes ?**

SSD – A different beast

- SSDs differ from disks
  - No mechanical or moving parts
  - Contain multiple flash elements
  - Different internal architecture
  - Complex internal operations
- SSDs differ among themselves
  - Low, medium, and high end devices
  - Firmware, interconnections, mapping, striping, ganging
- **Will the existing file system assumptions hold ?**

Problem

- Several assumptions are no longer valid

Assumptions	Disks	SSDs
Sequential accesses much faster than random	✓	✗
No write amplification	✓	✗
Little background activity	✓	✗
Media does not wear down	✓	✗
Distant LBNs lead to longer access time	✓	✗

- Implications
  - **Need to modify storage stack for SSDs ?**

Results

- Modifications to tune storage stack for SSDs
  - Cope with violated assumptions
- Rich interface to convey more information to device
  - IO priorities
  - Free blocks
- More functionality in device
  - Low level block management
- Possible Solution
  - Object based storage (OSD)

Talk outline

- SSD benchmarking
- Case studies
  - Write amplification
  - Background activity
  - Device wear-down
- Object-based storage
- Related work
- Conclusion

SSD benchmarking

- Used a range of SSDs for experimentation
  - Engineering samples and pre-production samples
  - Used both SLC and MLC-based SSDs
  - Anonymized the SSDs as S1, S2, S3, S4
- Performed read/write experiments on
  - HDD: Seagate Barracuda 7200.11 drive
  - SSD samples

SSD benchmarking results

Device	Read (MB/s)			Write (MB/s)		
	Seq	Rand	Ratio	Seq	Rand	Ratio
HDD	86	0.6	143	86	1.3	66
$S1_{slc}$	205	18	11	169	53	3.1
$S2_{slc}$	40	4.4	9	32	0.1	328
$S3_{slc}$	72	29	2.4	75	0.5	151
$S4_{mlc}$	68	21	3.2	22	15	1.5

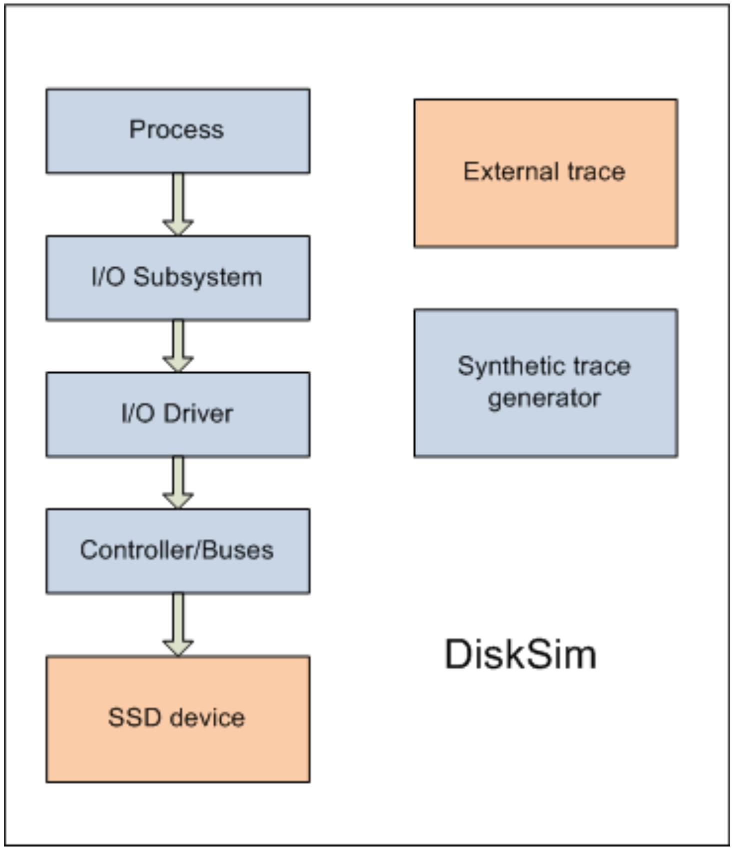
- Random-reads fast in SSDs
- Random-writes getting better with FTL techniques

Talk outline

- SSD benchmarking
- **Case studies (3 violated assumptions)**
  - Write amplification
  - Background activity
  - Device wear-down
- Object-based storage
- Related work
- Conclusion

Methodology

- Measurement on real SSDs
- File system traces from real machine
- DiskSim simulator (PDL)
  - Complete storage stack
  - Synthetic trace generator
  - External traces
- SSD module extension

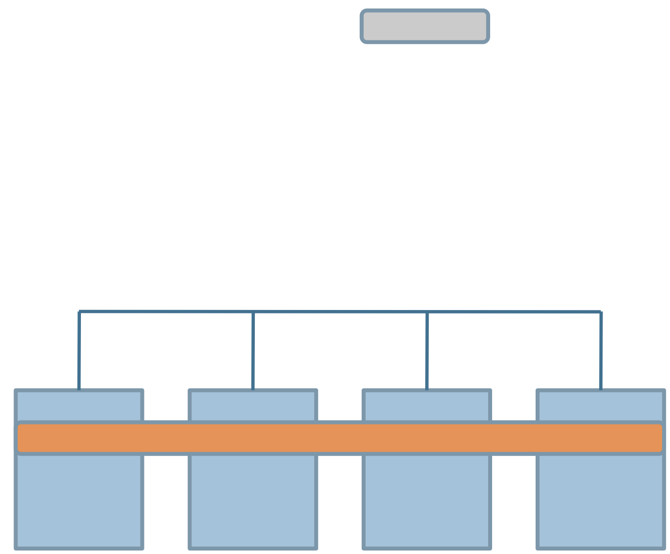


Talk outline

- SSD benchmarking
- Case studies
  - **Write amplification**
  - Background activity
  - Device wear-down
- Object-based storage
- Related work
- Conclusion

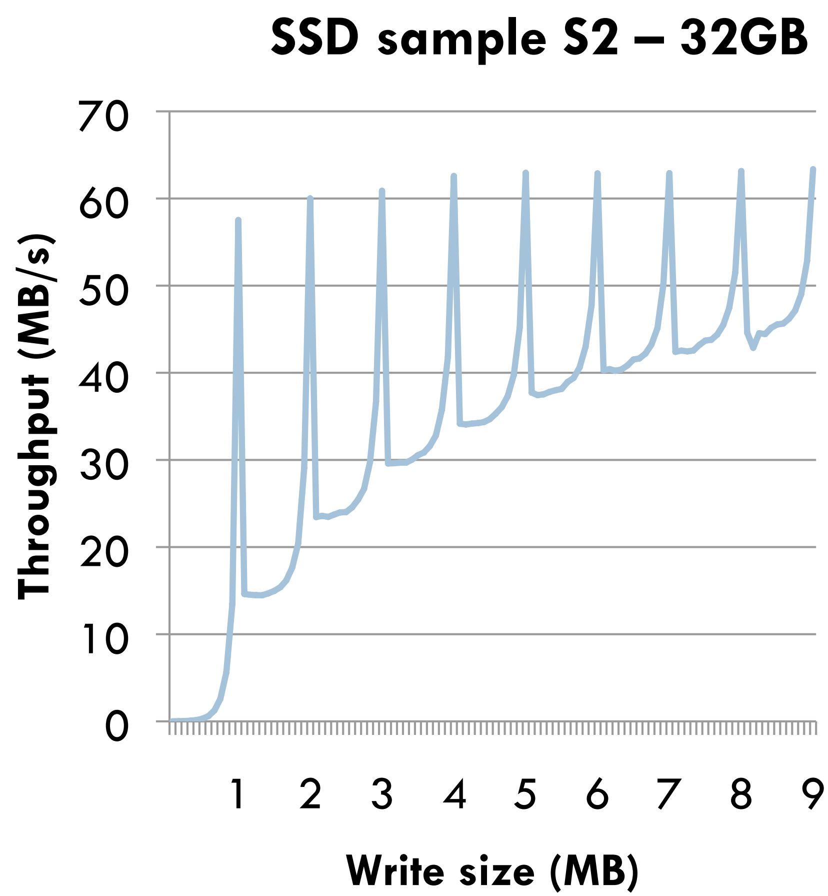
Write amplification

- Low-end and medium-range SSDs
- Reasons
  - Write size < stripe size
  - Physical page < logical page



Write amplification in real device

- Measurements taken on a real device
  - SSD sample S2 – 32GB (Low end SSD)
- Experiment: Issued continuous writes of varying sizes
- Writes are striped
  - Stripe size: 1 MB
- **Write amplification not seen in S1, S4**



Write amplification improvement

Violated assumption

- No write amplification

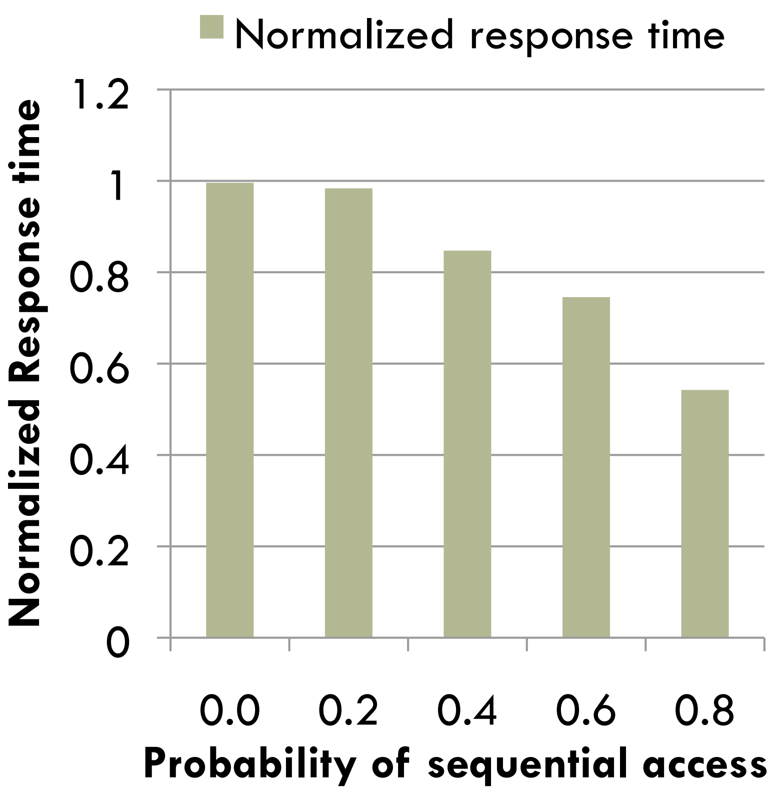
Proposed improvement

- Merge requests along stripe boundary in device

Case study implementation

- Implemented logic in simulator SSD module
- Run traces on modified simulator

Write amplification- Results



Synthetic trace

Benchmark	Improvement (%)
Postmark	1.15
TPCC	3.08
Exchange	4.89
IOzone	36.54

Real benchmark traces

Talk outline

- SSD benchmarking
- Case studies
  - Write amplification
  - **Background activity**
  - Device wear-down
- Object-based storage
- Related work
- Conclusion

Background activity

Violated Assumption

- Storage device passive - little or no background activity
- SSD does cleaning and wear-leveling

Problem

- Host can't control background activity
- Prevent effect of background operations on priority requests

Proposed Improvement: **Priority-aware cleaning**

- Inform device about priorities
- Device avoids background operations

Priority-aware cleaning - Implementation

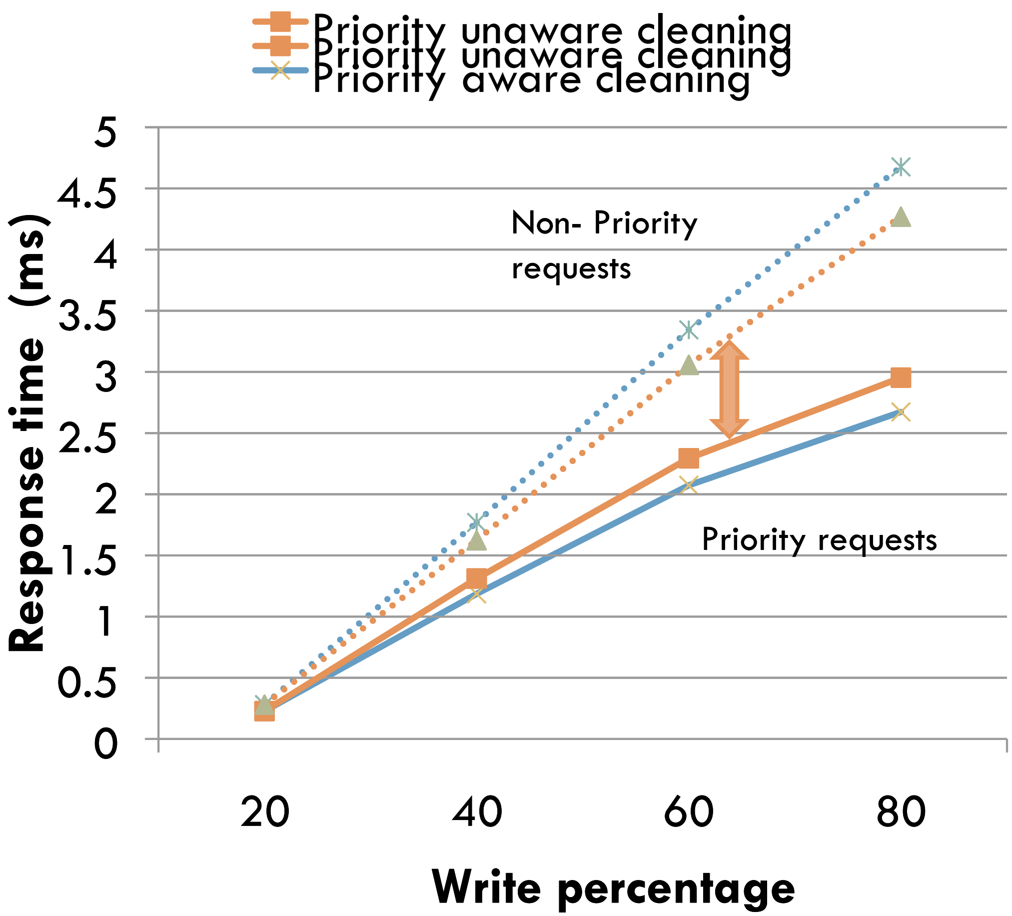
Methodology

- DiskSim supports priority request queuing
- Used synthetic trace generator
- Modified simulator SSD module

Improvement: **Priority-aware cleaning**

- Two cleaning thresholds
  - Low
  - Critical
- Outstanding priority requests
  - Clean only if below the critical watermark

Priority-aware cleaning - Results



- 10% improvement in re IO-Scheduling of prio requests
- Improvement at the cost rity traff

Talk outline

- SSD benchmarking
- Case studies
  - Write amplification
  - Background activity
  - **Device wear-down**
- Object-based storage
- Related work
- Conclusion

Device wear-down

Violated Assumption

- Media does not wear down
- SSD: Blocks have finite erase cycles

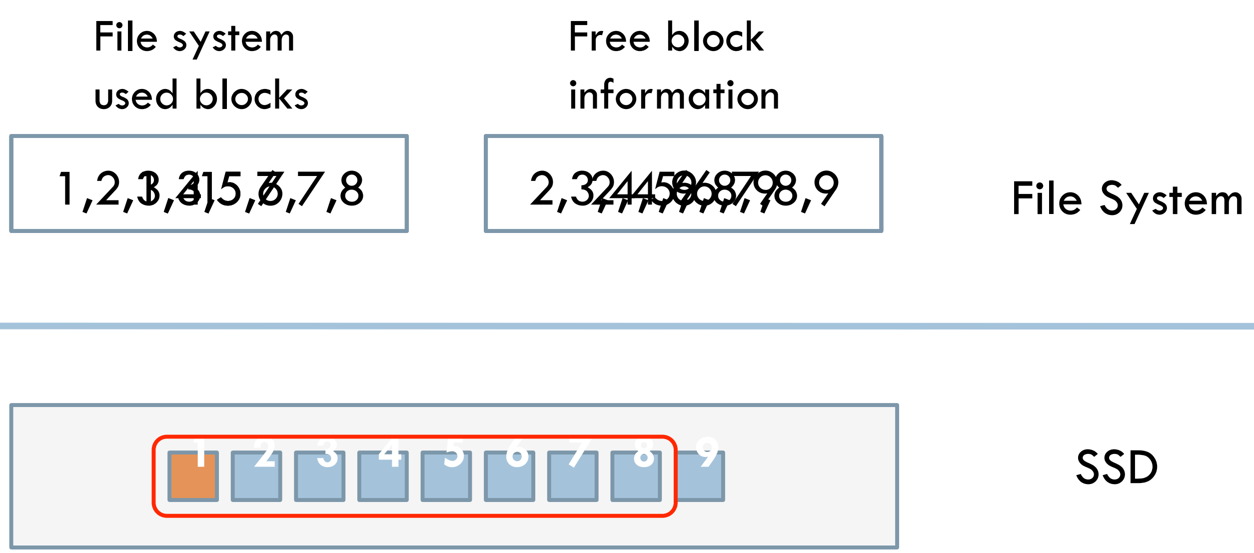
Problem

- Must reduce writes to blocks

Proposed Improvement: **Informed Cleaning**

- File system has free block information
- Inform device about block freeing
- Free blocks need not be copied in cleaning

Informed cleaning - Example



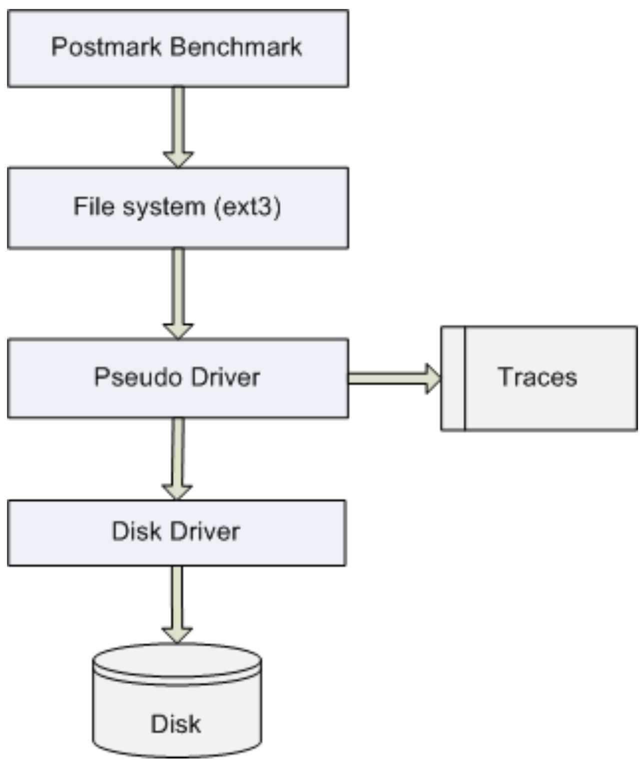
Informed cleaning - Implementation

Methodology

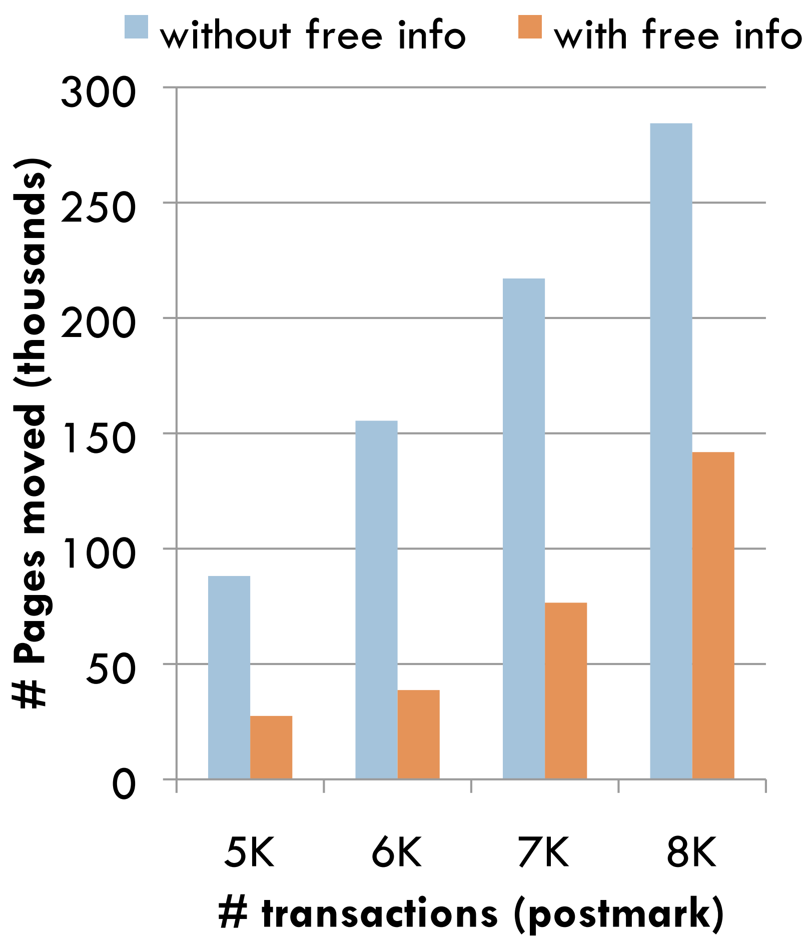
- Used postmark benchmark traces from real machine
- Intercepted block-free calls at pseudo driver below FS
- Generate real traces with free block information

Improvement: **Informed Cleaning**

- Modified simulator SSD module
  - Track freed blocks
  - Skip copying free blocks for reclamation



Informed cleaning - Results



- Cleaning efficiency
  - One-third pages moved
  - Cleaning efficiency improved by factor of 3
  - **Device lifetime improved**
- Cleaning time
  - 30 to 40 % improvement
  - **Response time improved**

Summary of improvements

- Write amplification
  - Need “stripe size” from device
- Background activity (Priority aware cleaning)
  - Need “IO priority” information from OS
- Device wear-down (Informed cleaning)
  - Need “free block” information from FS
- **Need richer interface**

Possible solution

- SSD has intricate knowledge of its internals
  - Amount of parallelism
    - Ganging ?
    - Shared bus and/or shared data ?
  - Logical to physical mapping
    - Super-paging ?
    - Striping ?
  - Internal background operations
    - When cleaning and wear-leveling ?
    - Separate unit for cleaning ?

Solution:

- Rich interface to convey higher level semantics
- Device handles block management

SSD as OSD

- OSD manages space for objects
  - Informed cleaning
  - Stripe aligned accesses
  - Logical to physical mapping
- OSD has object attributes
  - Wear-leveling using cold data information
  - Priority assigned to objects
- OSD handles low-level operations
  - Block management in SSD

Related work

- Design tradeoffs for SSDs
- MEMS-based storage devices and standard disk interfaces
- Operating system management of MEMS based storage devices
- Bridging the information gap in storage protocol stacks
- Non-Volatile Memory Host Controller Interface 1.0
- Object-based storage
- Track-aligned extents

Conclusion

- Revisited storage specific assumptions for SSDs
  - Several assumptions violated
- Proposed improvements to tune storage stack for SSDs
- Need for richer interface
- More functionality in devices
- One possible solution: OSD
  - Understands high-level semantics
  - Handles low-level operations

Questions