NVMFS: A Hybrid File System for Improving Random Write in SSDs

A.L. Narasimha Reddy
Sheng Qiu (Ph.D. Student)

Flash based SSD

- Widely used in current storage systems
  - Flash-SSD disk cache
  - Purely Flash-SSD storage
- Better random read than HDDs
- Decreased price ($/GB) and increased capacity

- *Small Random write still problematic !*

Outline

- Introduction
- Emerging nonvolatile memories
- Hybrid Storage System
- Evaluation
- Summary

Emerging NVMs (eNVMs)

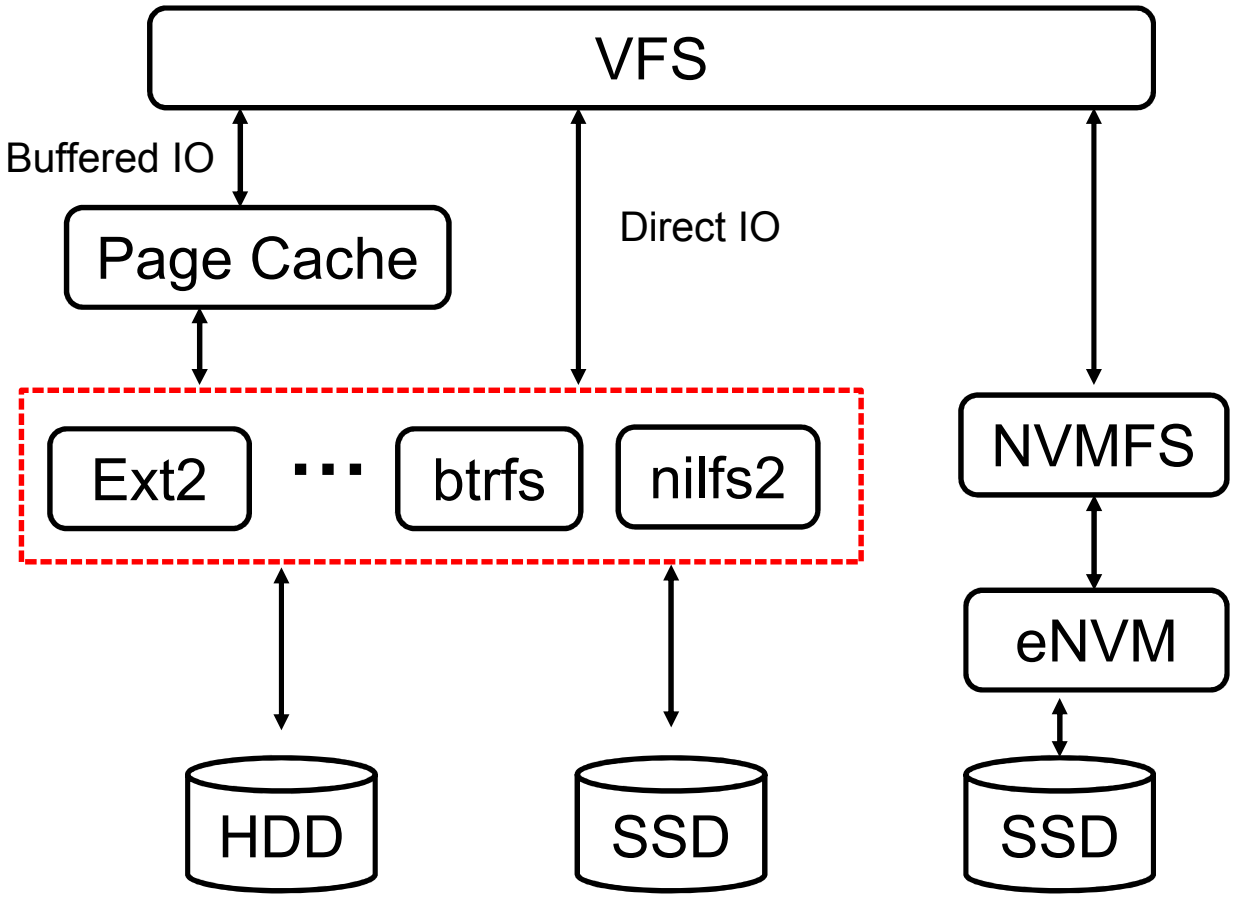
- eNVMs becoming widely available
  - PCM
  - DRAM + Flash combo DIMMs
- Small random writes caused by FS and Databases pushing data frequently from DRAM to storage
- What can we do with eNVMs cost-effectively?
  - Can we absorb periodic flushes of data?
  - Can we shape the workload seen by the storage system?

Outline

- Introduction
- Existing Disk Storage System
- Hybrid Storage System
- Evaluation
- Summary

Our solution

- Hybrid Storage --- eNVM + Flash-SSD
- NVMFS manages both devices

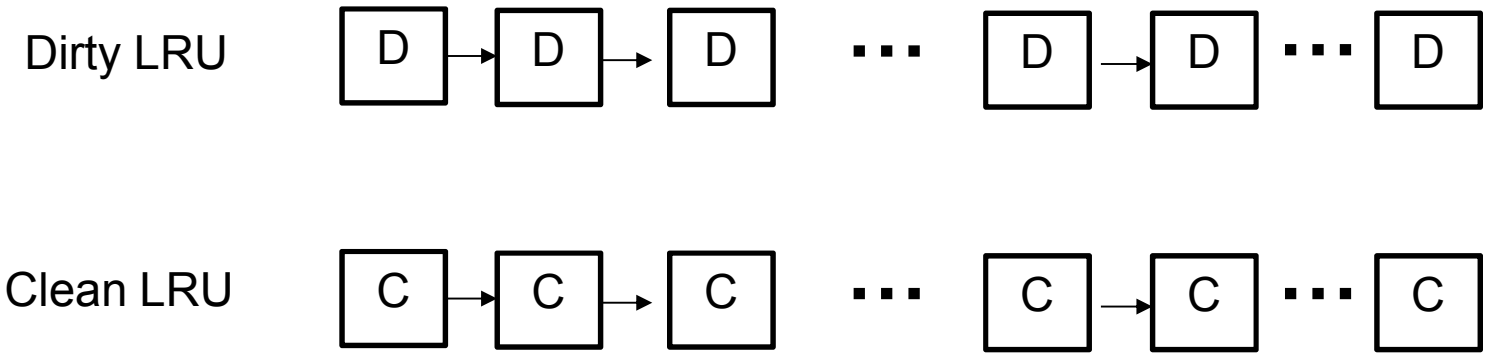


NVMFS Design

- Employ caching and migration within one system
  - Cache clean data on eNVM for read
  - Cache newly allocated data on eNVM for write
  - Migrate dirty data to eNVM for write
- Keep two possible FS locations for data
  - Either on eNVM or SSD or both (cached clean data)
  - Employ different FS address for different device
- Have more flexibility on how to write to SSD
- Writes to SSD always in full erase block size
  - Make writes efficient

What data to keep on eNVM?

- Dirty data
  - Limited to dirty_max%
- Hot clean data
- New allocation replaces LRU page in clean list

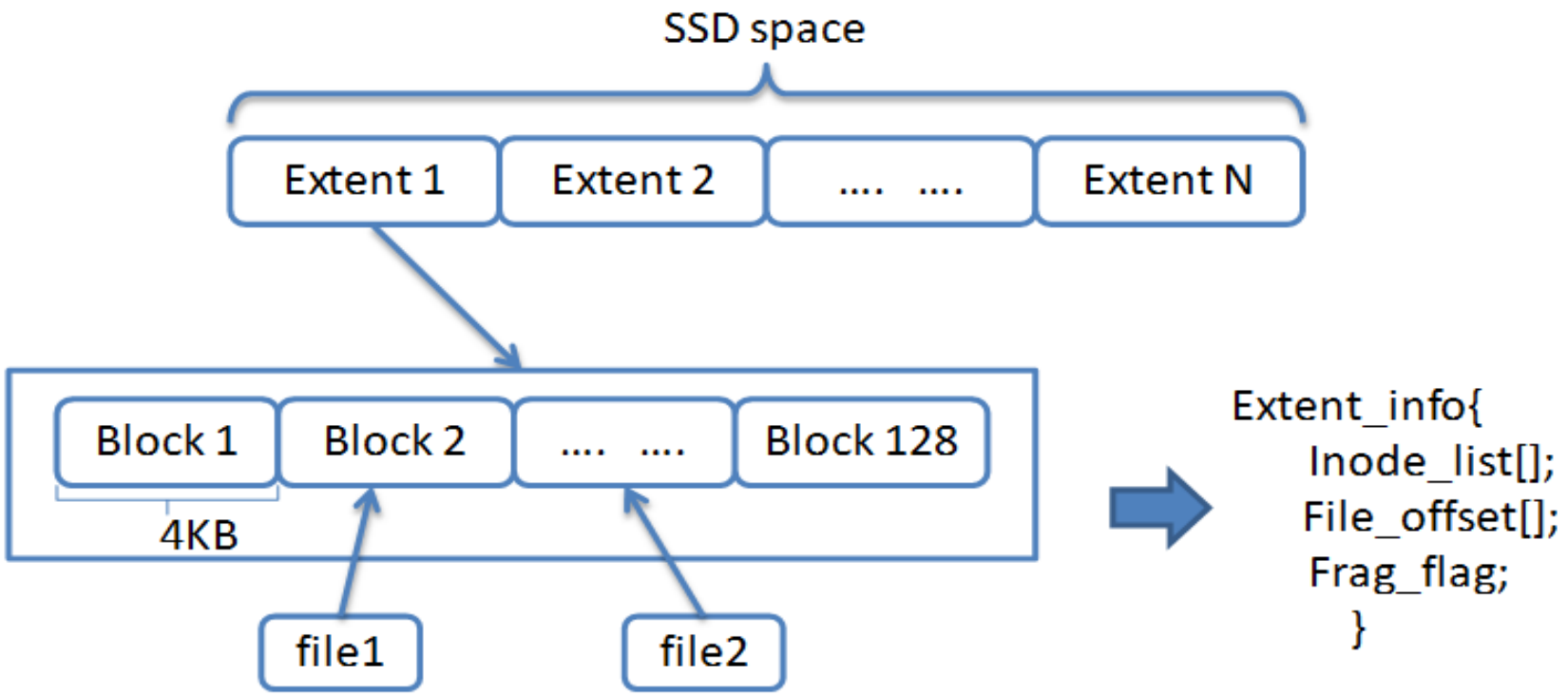


Writes to SSD

- When?
  - Dirty data ratio > watermark%
- What data to write?
  - LRU pages from dirty LRU list
- How to write on SSD?
  - Allocate one extent (512KB) per round
  - Writes in full erase block size

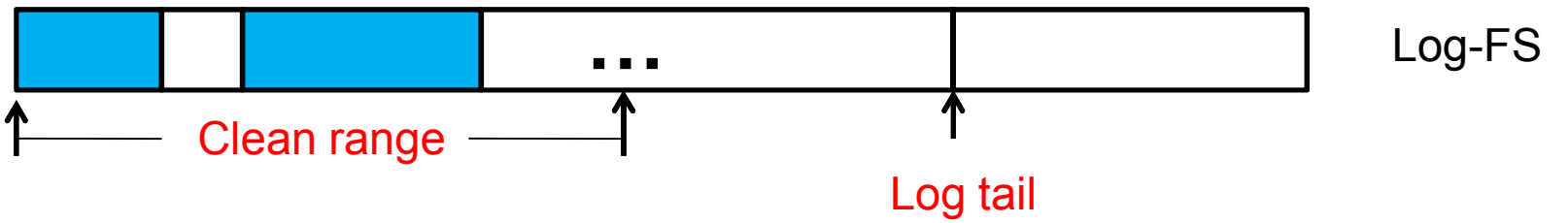
Space management on SSD

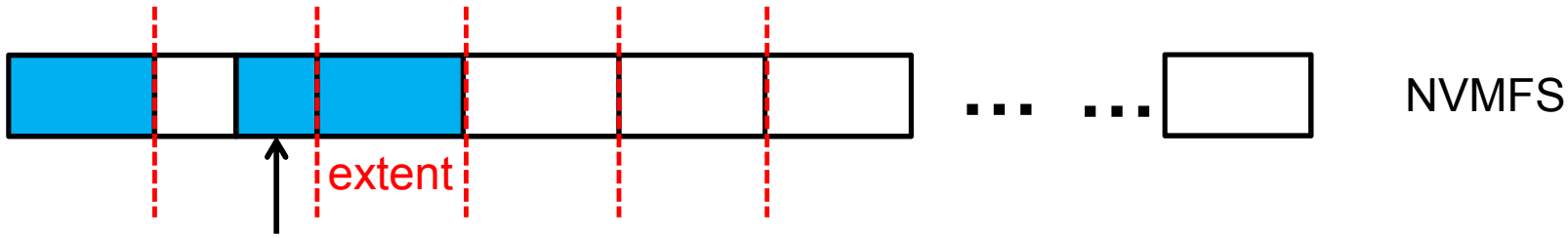
- Allocation/write unit is one '**extent**' (512KB).
- Read unit can be as small as 4KB.
- Recycle fragmented extents on below conditions
  - Valid_space_size / allocated_extents_size > frag_ratio%
  - Free_extent_num < min_num



Diff with log-FS

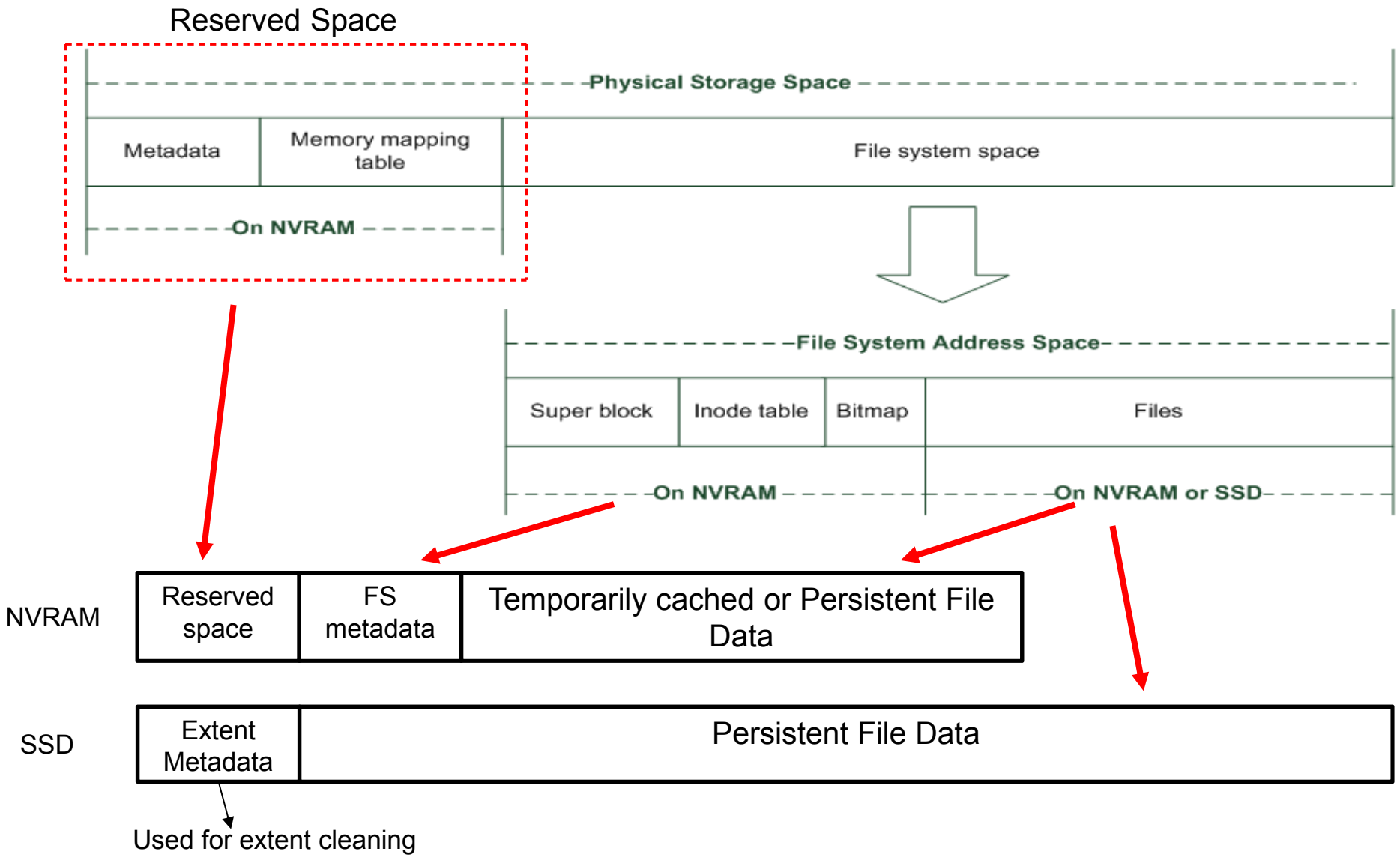
- Better data grouping
- Different cleaning strategy
  - Log-FS has to clean up a range of sequential space for during log-cleaning for future allocations
  - NVMFS only need clean up each individual extent





Only migrate valid data within partially used extent, if extent is fully used, no need to migrate them

File system layout



Important Data Structures

- Page_info_table
  - Each entry is a page_info structure
  - Entry size is total number of eNVM pages
- Page_info
  - The inode this page belongs to
  - The LBA on SSD if it's a cached clean page
  - The fileblock number this page belongs to
  - The embedded LRU link list
- SSD extent_info
  - The inode each valid FS block belongs to
  - The fileblock number each valid FS block belongs to

Outline

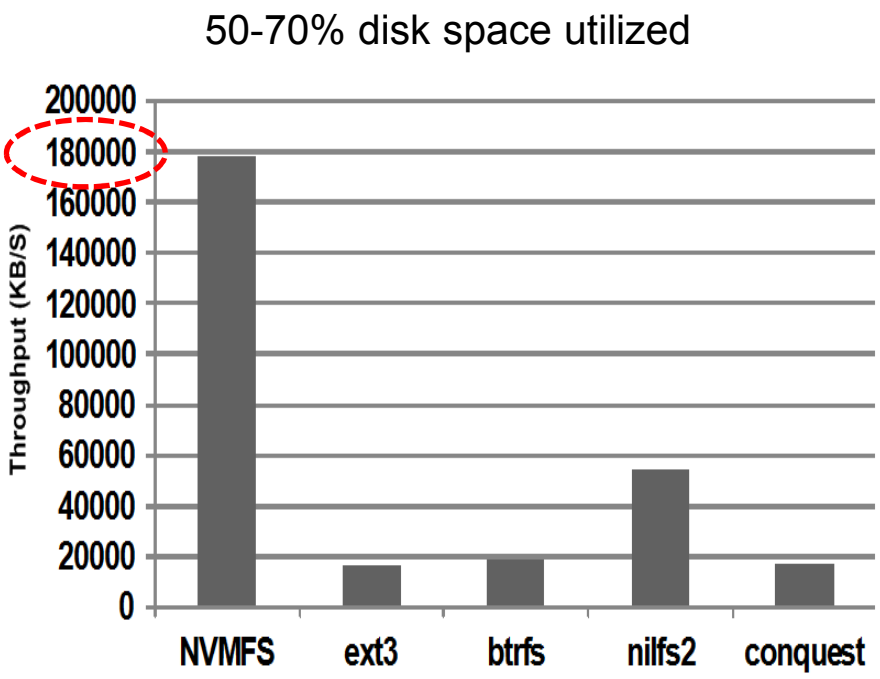
- Introduction
- Existing Disk Storage System
- Hybrid Storage System
- Evaluation
- Summary

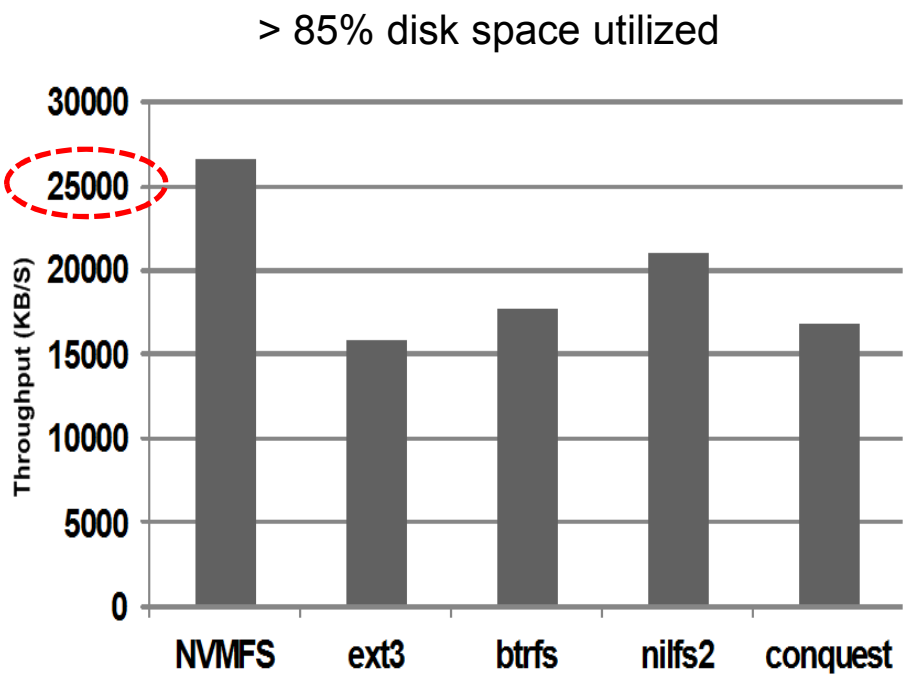
Evaluation

- Implemented within Linux Kernel 2.6.33
  - About 3000+ lines of codes
  - Touched MM, FS
- Hybrid Storage
  - 4GB DRAM mapped as eNVM (OS differentiates it with regular DRAM)
  - 64GB Intel X25-E SSD
- Compared with other file systems
  - Ext3, btrfs, nilfs2 and Conquest
- Benchmarks
  - Fio, Postmark, Filebench and IOZONE
  - Perform random 4KB IOs on multiple files or one large file

Results

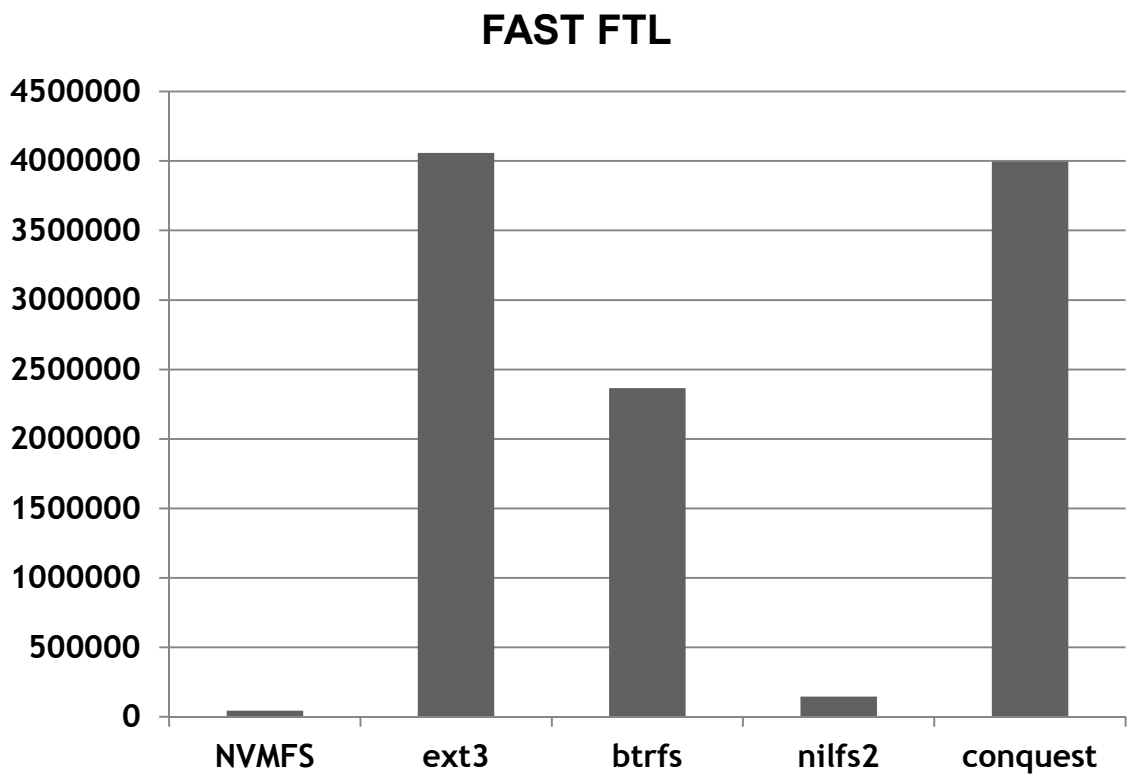
- IO throughput (IOZONE)





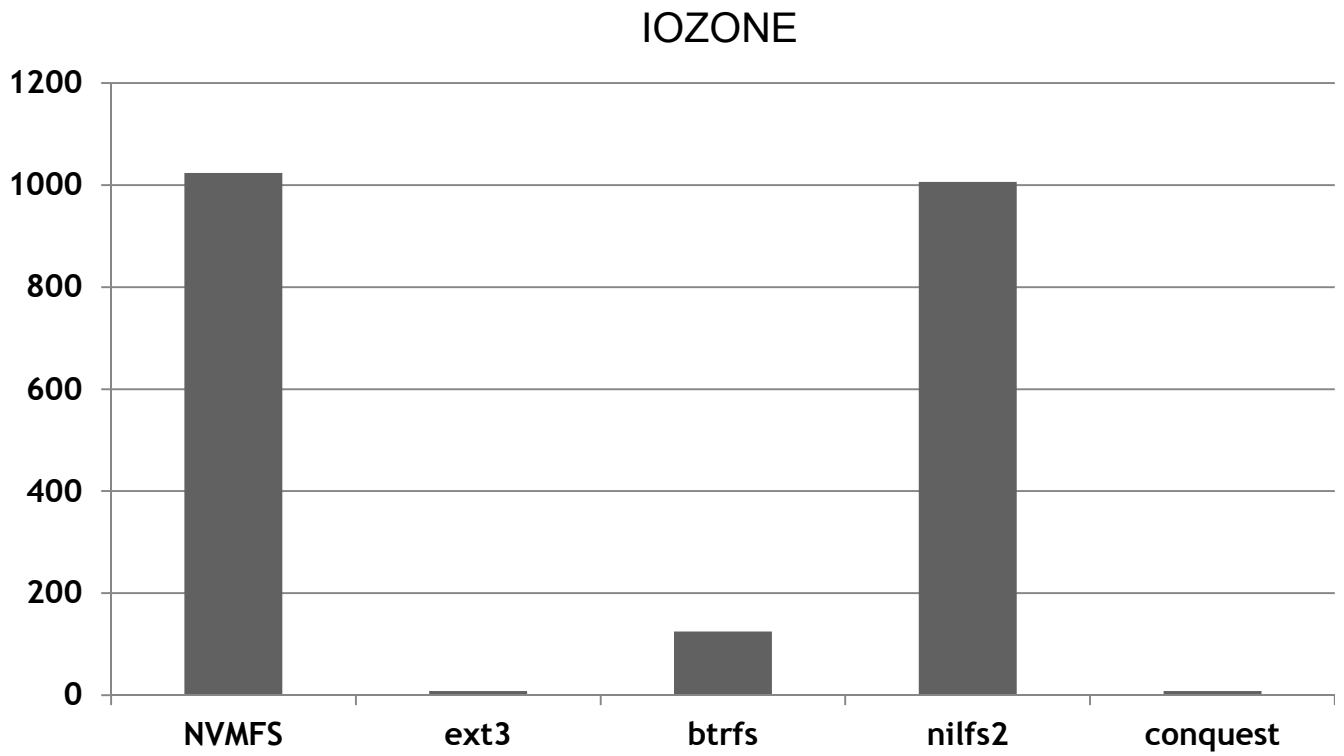
Results (cont'd)

- GC efficiency (IOZONE)
  - Y axis -- Number of erase operations



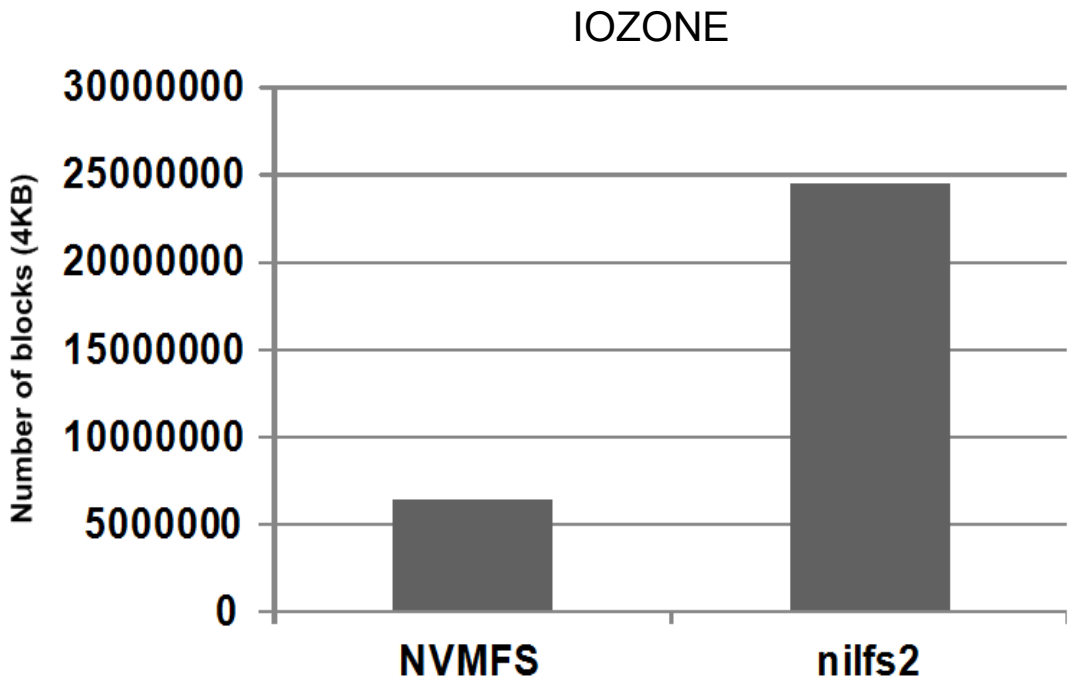
Why NVMFS performs better at GC?

- Transform small writes to large ones
- Y axis --- the average request size (sectors)



Why NVMFS performs better at GC? (cont'd)

- Better data grouping within SSD erase block
  - Fewer valid blocks to recycle during extent/segment cleaning



Outline

- Introduction
- Existing Disk Storage System
- Hybrid Storage System
- Evaluation
- Summary

Summary

- Flash-SSD's random write is still problematic in terms of both performance and lifetime
- Emerging NVMs can be integrated into existing storage stack to improve SSD's random write
- The proposed NVMFS demonstrated how to utilize eNVMs to improve SSD's performance and GC
- In future we want to explore more on optimizing Flash-SSD based storage system