NetApp™

Go further, faster®

Improving throughput for small disk requests with proximal I/O

Jiri Schindler

with Sandip Shete & Keith A. Smith

Advanced Technology Group

2/16/2011

v.1.3

Important Workload in Datacenters

- Serial reads after random writes
  - Writes are (small) logical updates
  - Serial reads of continuously changing data

- Database systems example
  - Data acquired through transactions
  - Data scans for business intelligence

- Datacenter/enterprise scale
  - Flash memory prohibitively expensive
  - Use disk drives for serial reads

Inherent Tradeoff in Current FS Designs

- Optimized for one prevalent access pattern
  - either serial reads or random updates
- Update-in-place writes (e.g., FFS, ext2 etc.)
  - Preserve physical locality for serial reads
  - Updates inefficient, especially with parity RAID
- LFS-style log-append writes
  - Fragment on-media layout for serial reads

Want a file system where both random writes and serial reads are efficient

Our Approach – Briefly

- Stage random updates in NV memory buffer
- Destage (bulk write-allocate) to disk
  - Use proximal disk I/O for efficient writes
    - Can retire multiple I/Os per revolution
- Write allocation with no overwrites
  - Old versions used for snapshots etc.
  - Maintains desired serial-read efficiency
    - Write to free locations near related data

RAID 1-like write performance on parity RAID
Minimal serial read degradation on aged FS

Remainder of the Talk Outline

- System concepts
- Experimental results
- Concluding remarks

SYSTEM CONCEPTS

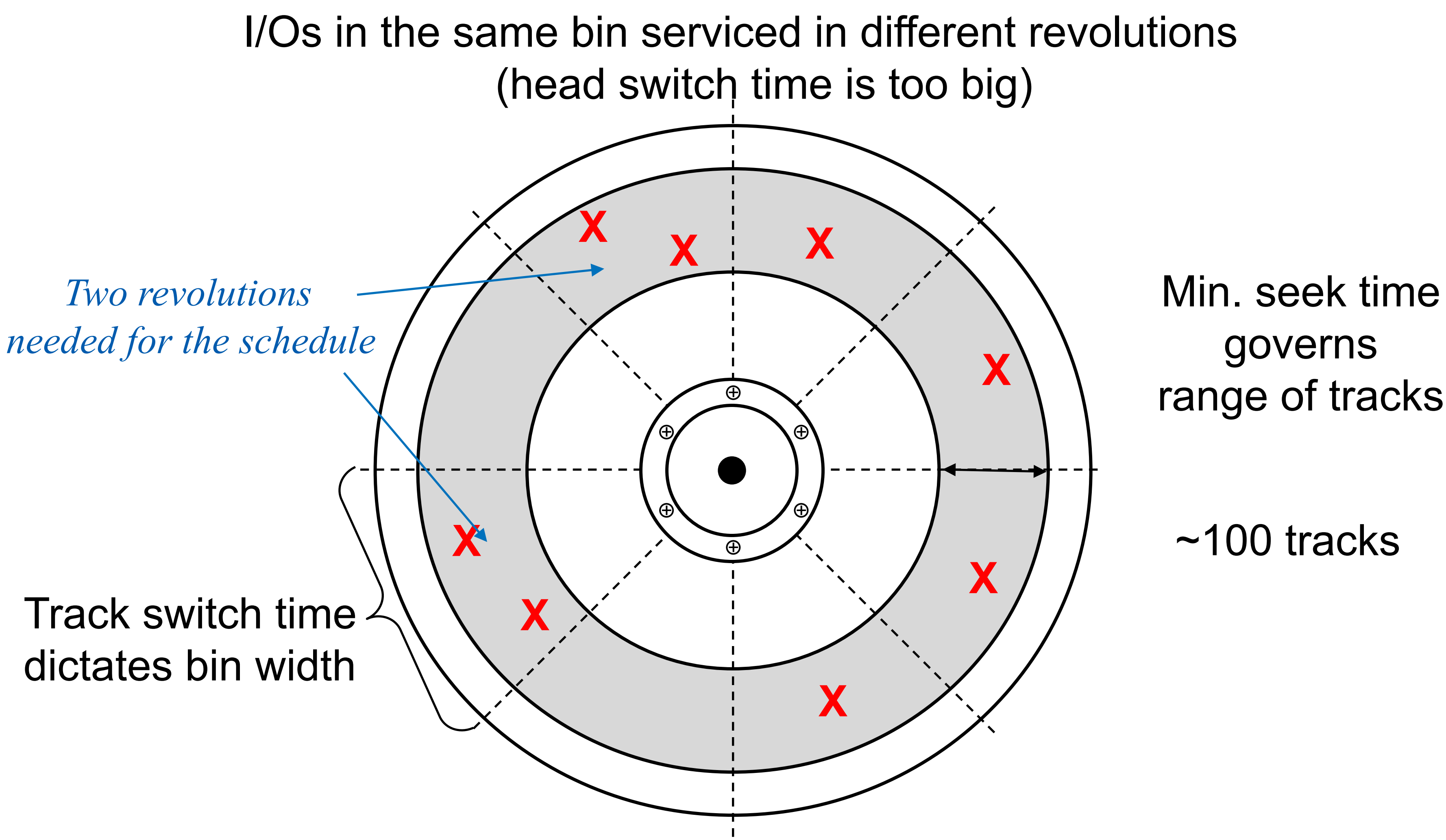
Efficient Proximal Disk I/O – Concept

- Disk technology trends
  - Minimal seek/head switch time <1ms
    - Same repositioning cost within ~100 of tracks
  - Increasing aerial bit density is in our favor
- Can service multiple small I/Os per rev.
  - Up to 10 locations within a span ~100K blocks
  - Large degree of freedom for write allocation

- Near-line (SATA) 7200 RPM disk
    - 8.3 ms per revolution
    - 0.8 ms head switch time

Proximal Disk I/O Details

- Assume 8 I/Os per revolution



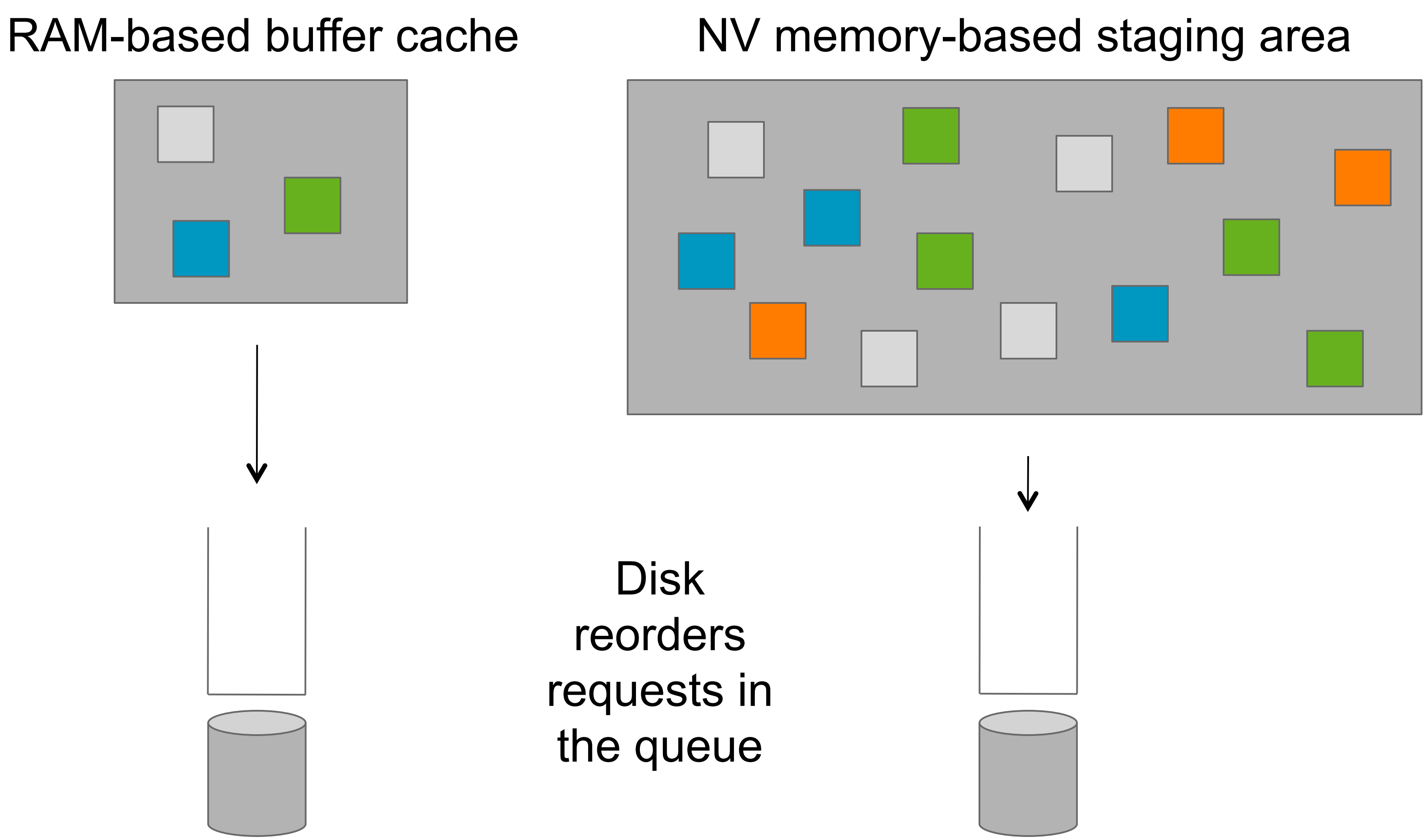
Achieving Enough I/O Density

- Assume random updates
  - Single 1 TB disk holds ¼ billion blocks
  - Proximal I/O needs 6 to 8 blocks/ ~100K blocks
- Stage random updates in NV memory
  - Destage to disk when required density achieved
- Trends in out favors
  - Not all I/Os are not user I/Os
  - “Degree of randomness” workload-dependent

Stage area sized at 1% of the working set sufficient

Putting it All Together

Traditional approach | Our approach with proximal I/O



Features & Some Details

- Small (over)writes absorbed by Flash memory
  - Metadata updated immediately
    - Updates create “holes” in on-disk layout
  - New versions of data accessed from Flash in parallel while streaming other data from disk

- Destaging
  - Batch blocks belonging to the same extent
  - Metadata reads amortized during destage
    - read once access many times

EXPERIMENTAL RESULTS

Experimental Setup

- File system data layout engine (DLE) prototype
  - Extent Create/Read/Write/Unlink ops
  - Basic implementation of allocation algorithms

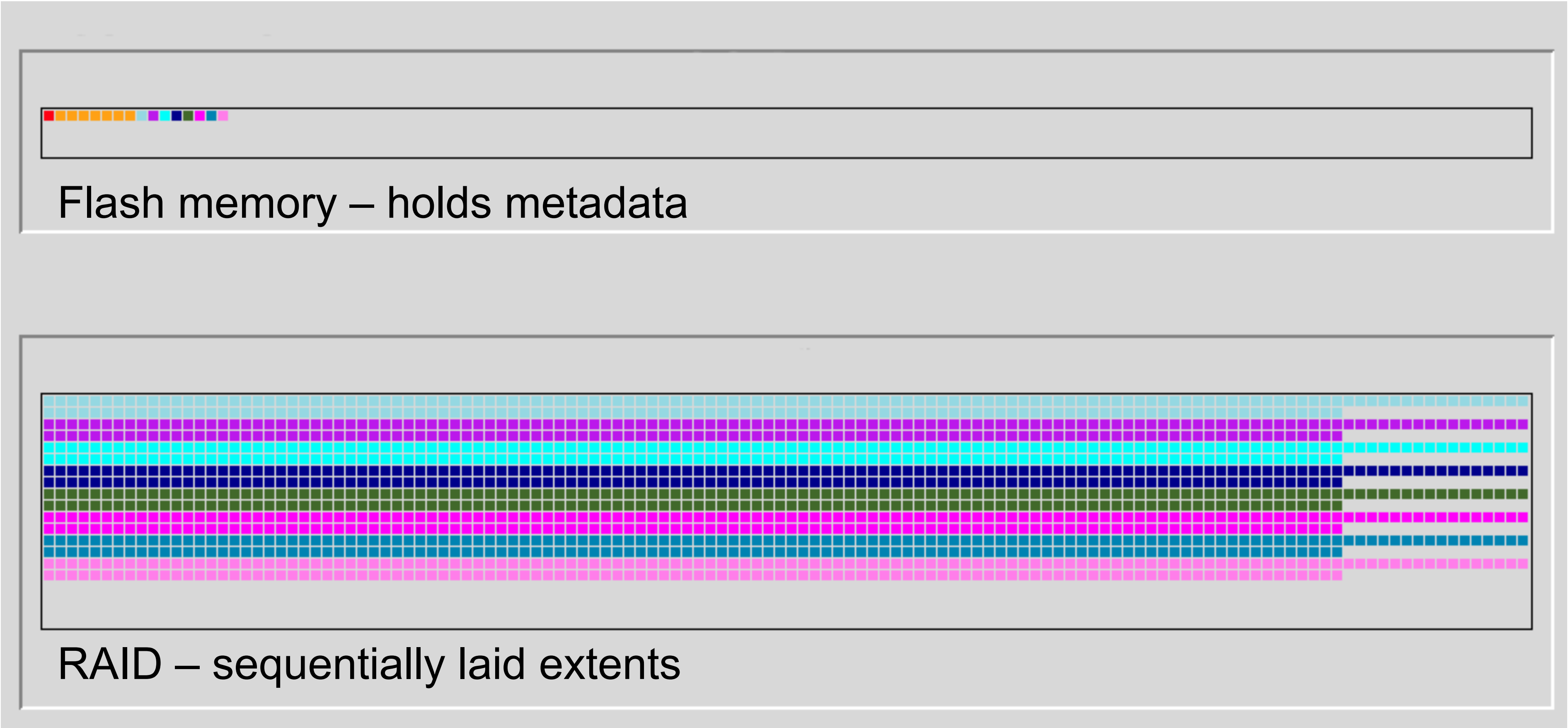
- Data Storage
  - NetApp DS4243 Shelf with 1TB SATA disks
  - Staging area on FLASH-based SSDs
  - NetApp RAID user-level emulator
    - Each FS 4KB block has checksum/context info
      - One block for every *n* user-blocks

Basic Workload Description

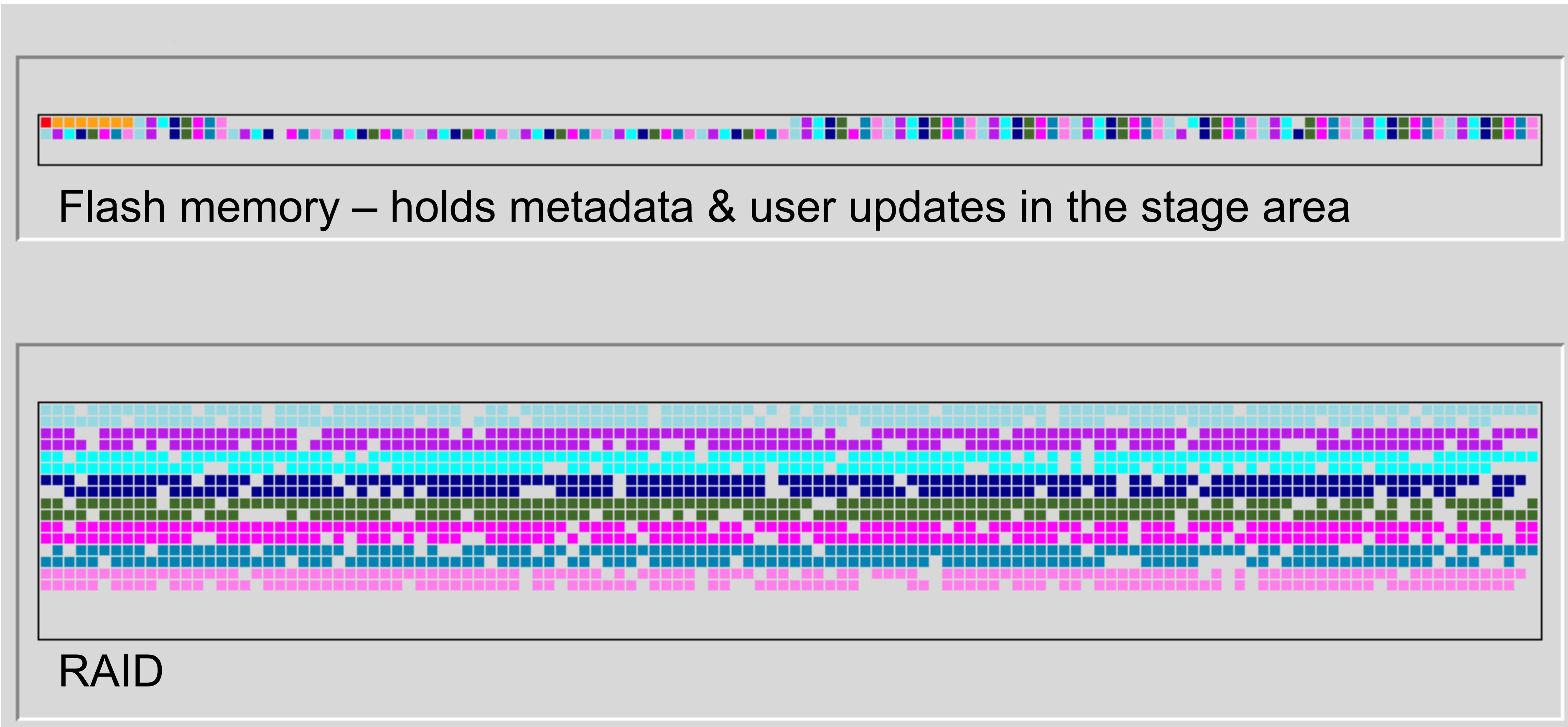
- Write sequentially large (16MB) extents
- Many random small updates
  - "age" initial on-disk layout
  - destage I/Os from Flash to disk

- Adverse scenario: 90% full FS
  - Limits allocation choices
- Vary staging area size
  - n% of disks' capacity

New File System Layout



Aged File System Layout



Small Random Updates

4+1 RAID4 (WD1002FBYS-05ASM 7200 RPM SATA disks), NetApp DS4243 shelf

	User writes per batch	Batch resp. time	Service time per user write	I/Os per revolution	I/O amplification
Baseline	1	16.1 ms	16.1 ms	2.0	8x
1% Stage	47.5	129.5 ms	2.7 ms	5.3	6.2x

BASE: update-in-place writes w/ checksum block
% Stage: Flash size relative to RAID capacity

Batch: one destage operation (blocks of one file/extent)
I/O amplification: # of disk I/Os for each user write

- 6x smaller per user write service time
- 5.3 I/Os serviced per revolution
  - With minimal reduction in write amplification

Serial Reads after Random Updates

Mean RAID group I/O request size: 819 blocks   Max I/O size per disk: 256 blocks (1024 KB)

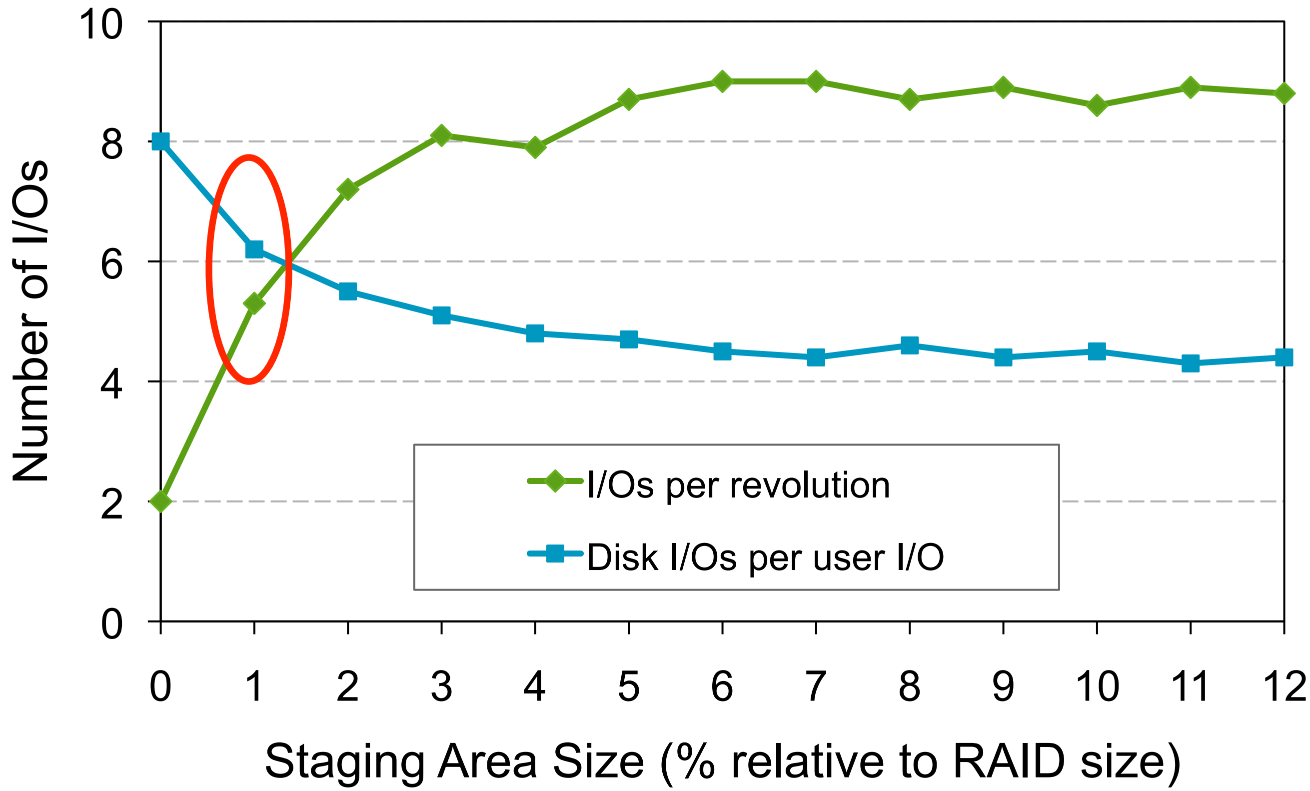
	Per-disk bandwidth		I/Os to RAID per request	Data Disks	
				Avg. I/Os per disk	Utilization (Busy time)
New FS	89.0 MB/s		1.2	11.7	85%
Aged FS	86.2 MB/s	**-3%**	26.3	20.7	82%
Aged LFS-style*	2.6 MB/s	**-97%**	509.9	210.2	85%

* no segment cleaning/background reallocation

- Only 3% BW degradation on aged layout

Flash Cost vs. Performance Trade off

- Increasing stage area size
  - Expressed as % of the RAID group size



- Diminishing return after stage area of 3%

CONCLUDING REMARKS

Summary

- Flash memory (~1% of HDD capacity)
  - Absorbs small random writes
  - Destage when enough data for efficient disk write
  - 3x more IOPS vs. equivalent disk-only system
- Making destage disk I/O efficient
  - Proximal I/O maximizes blocks accessed per seek
    - 5.3 I/Os per rev in the vicinity ~100,000 LBNs
  - RAID1-like I/O performance with RAID 4
- Minimize/eliminate background disk *grooming*
  - In-band reallocation during destage operation

Contact us

- At the poster session
- Email
  jiri.schindler@netapp.com
- Advanced Technology Group
  www.netapp.com/us/company/leadership/advanced-technology/
- Yes, we are hiring!
  http://www.netapp.com/careers