Sawmill: a logging disk array file system

Ken Shirriff

shirriff@cs.berkeley.edu

Sprite Project

University of California at Berkeley

Summary

Disk arrays provide high bandwidth.
How to get it to clients?

Sawmill file system uses:

- Disk array with high-bandwidth disk controller
- Log-structured file system (LFS)

Design features:

- New log layout technique
- Data path directly through controller
- Streaming instead of caching

Results:

- Reads at 21 MB/s, writes at 15 MB/s on Sun 4
- LFS good with RAID: small writes much faster

Outline

Background

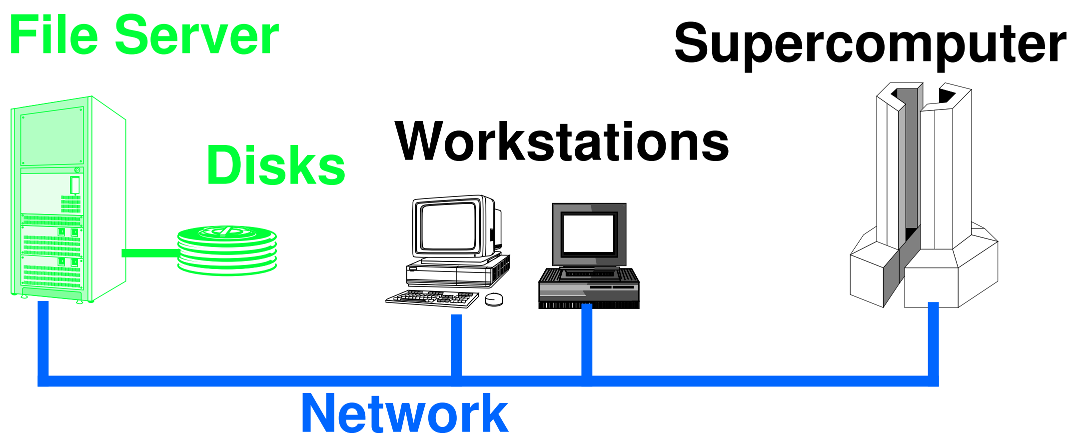
The Sawmill file system

Performance

Conclusions

Network file systems

File server provides storage for clients



Applications:

- Academic/office environment
- Diskless supercomputers

Goals:

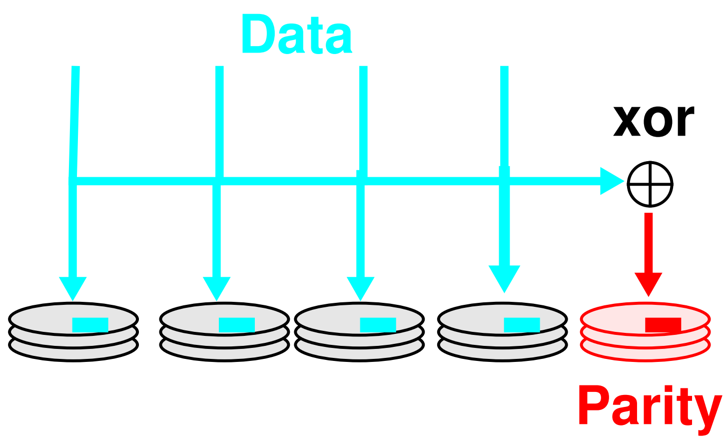
- High performance: bandwidth and latency
- reliability
- low cost

RAID

Redundant Array of Inexpensive Disks

Data is striped across disks

- Parallelism for speed
- Redundancy for reliability



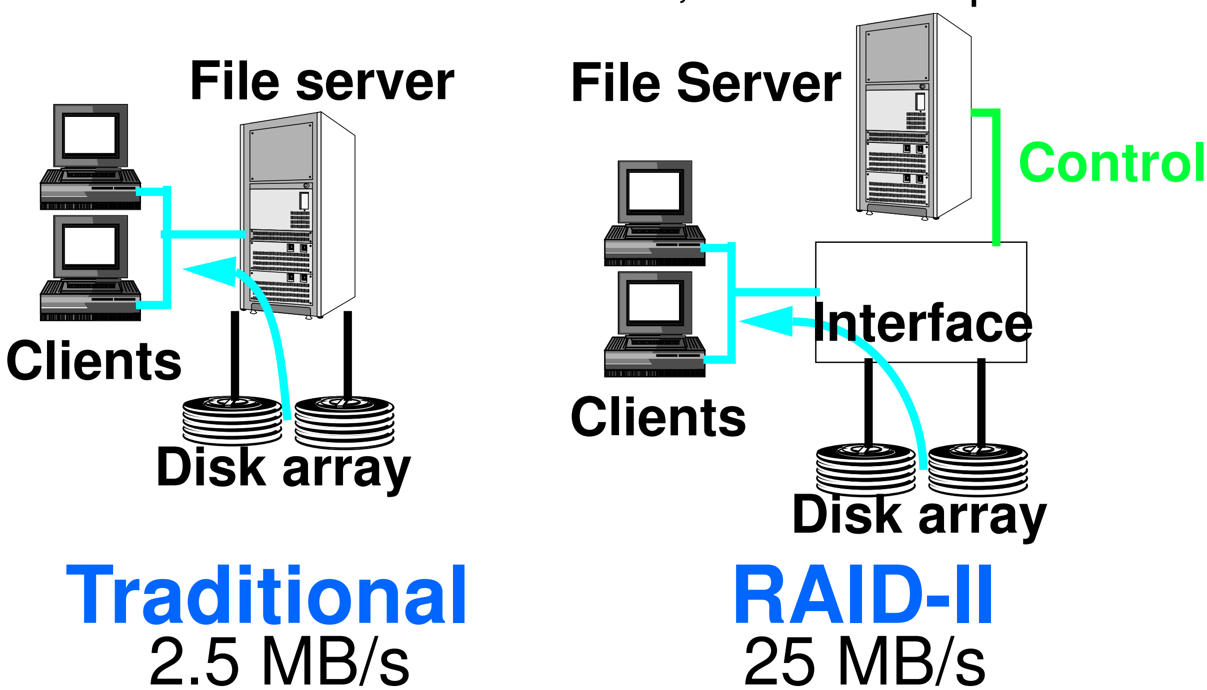
Small writes expensive: parity update

- Factor of 4 more expensive

Data path architecture

Bottleneck: File server data movement

Solution: Controller with fast, direct data paths



RAID-II:

- Data moves directly between disks and network
- Data does not go through server
- Data path optimized for high speed

Future of storage systems

RAID:

- Disk array for bandwidth, reliability

High bandwidth data path:

- File server not on data path

Fast network:

- FDDI, HIPPI, ATM

Why a new file system?

RAID:

- Current file systems: small transfers
- Small writes particularly bad

New controller:

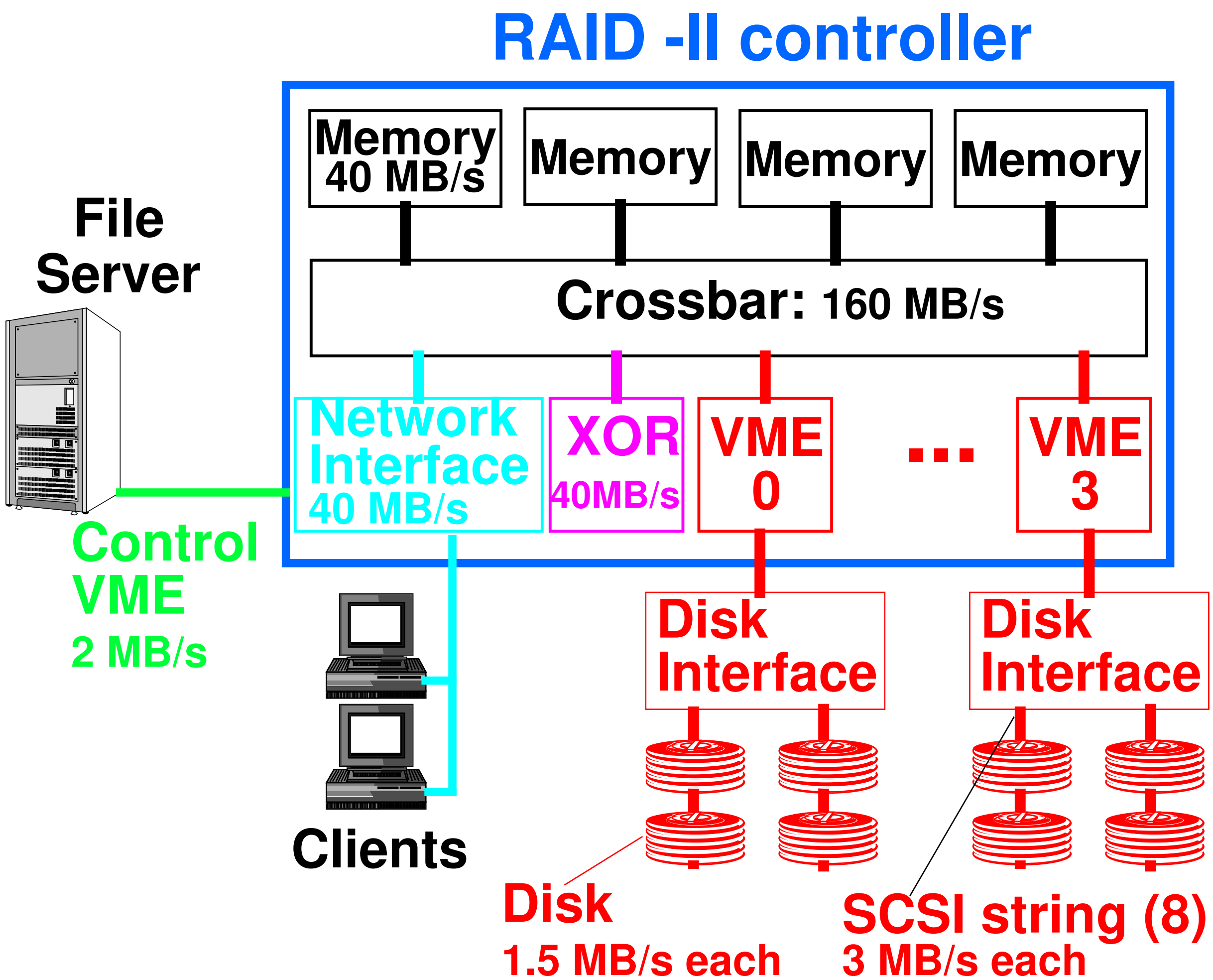
- Memory in controller data path: how to use?
- Metadata: needs server data path

Much higher data rates:

- File system overheads must be minimized
- Cache basis: high per-block overhead

RAID-II

Sawmill runs on RAID-II storage system

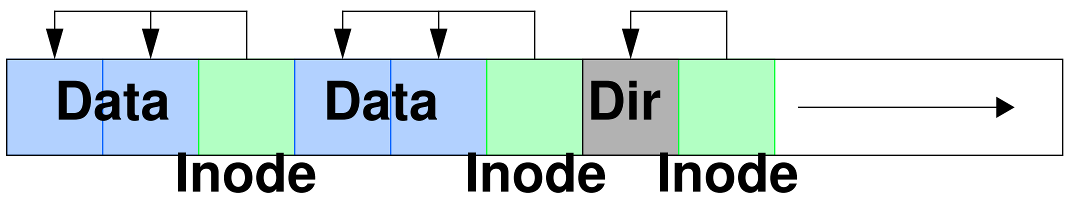


- High bandwidth path: disks ↔ network

Log-structured file system

LFS: data stored only in sequential append-only log

- No random disk writes



- New blocks appended with new inode
- Inode (descriptor) holds address of block

LFS is a good match for RAID:

- Log can be written in large units
- No expensive in-place parity update

Cleaning: garbage collection of free space

Outline

Background

The Sawmill file system

Performance

Conclusions

The Sawmill file system

Goals:

- Test a log-structured file system on RAID
- Use fast data path

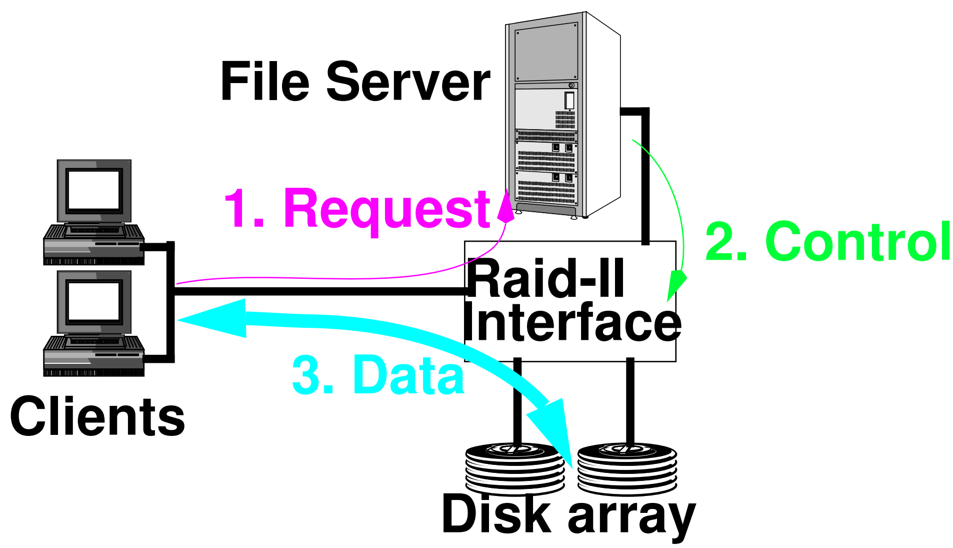
Key implementation ideas of Sawmill:

- Large disk transfers through batching
- Streaming instead of caching
- New log layout technique
- Metadata caching in server

Division of responsibility

File server: processes requests, provides file system abstractions

RAID-II: moves data between disks and clients



1. File server receives request

2. File server starts disk and network operations

3. RAID-II moves data between disks and network

Streaming vs. caching

How to use 32 MB memory on RAID-II controller?

File system block cache:

- 32 MB memory / 20MB/s data rate = 1.6 seconds
  Cached data unlikely to be reused

- Significant per-block overhead

Streaming:

- Treat large requests as stream: low overhead

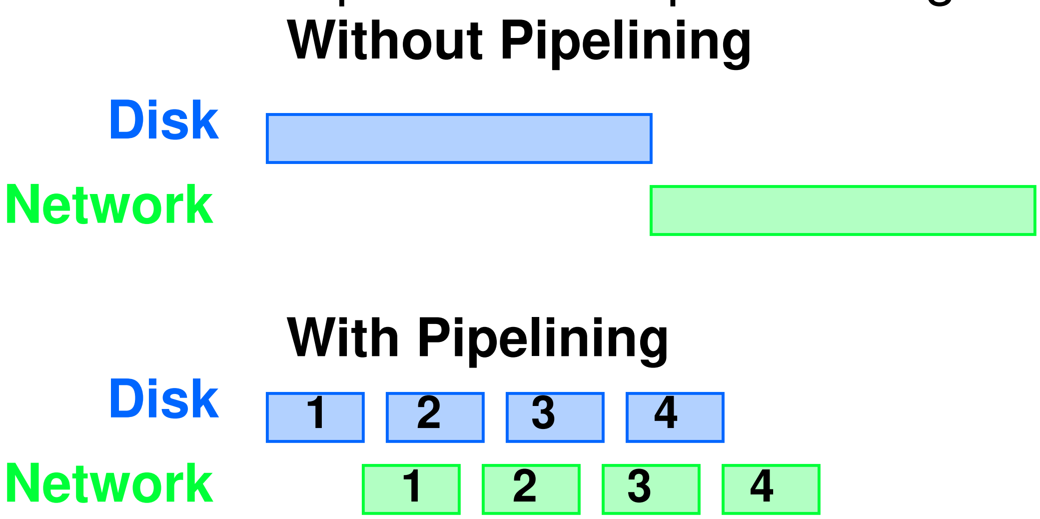
- Client caches provide caching advantages

Read pipelining

Idea: want to keep disks in use with large requests.

Overlap network sends with disk reads.

- Large requests: broken into segments
- For small requests: need prefetching.

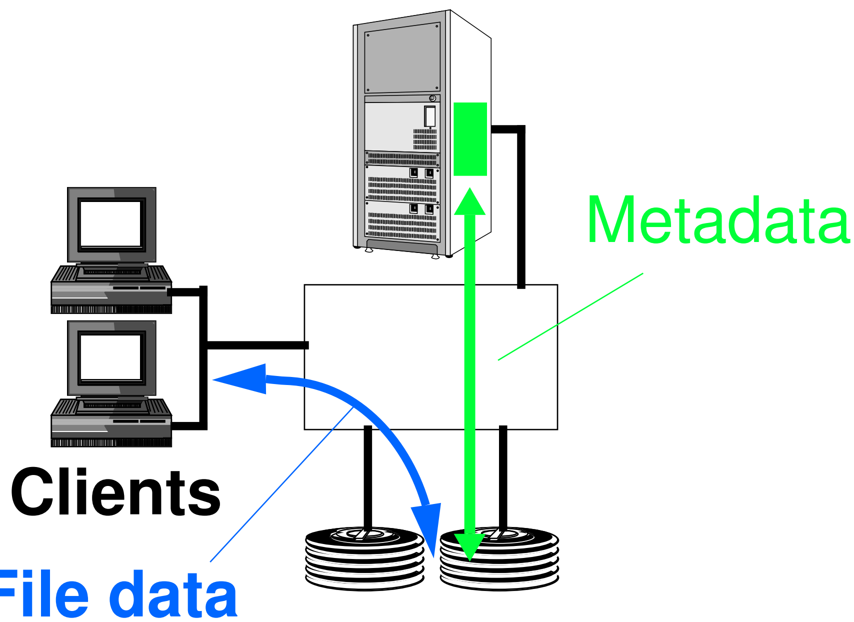


Metadata

Metadata: information needed by file system
• Directories, block addresses on disk, etc.

File system reads and writes metadata.
• Metadata is cached in server

• Consistency of cached data



Server cache

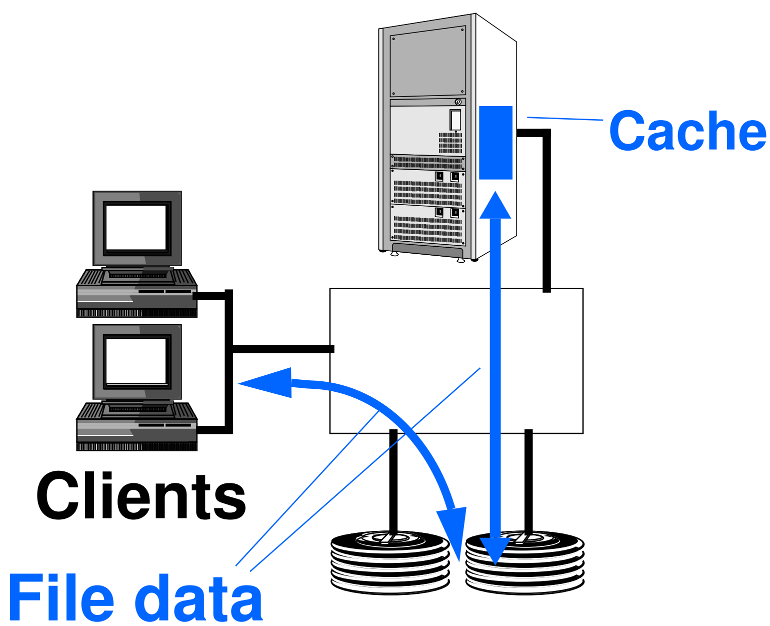
Reasons to keep data in server memory:
Metadata used in server memory.

Gateway to other networks

The file server may read files off the RAID-II disks.

Implemented server cache:
Standard file cache in the file server.

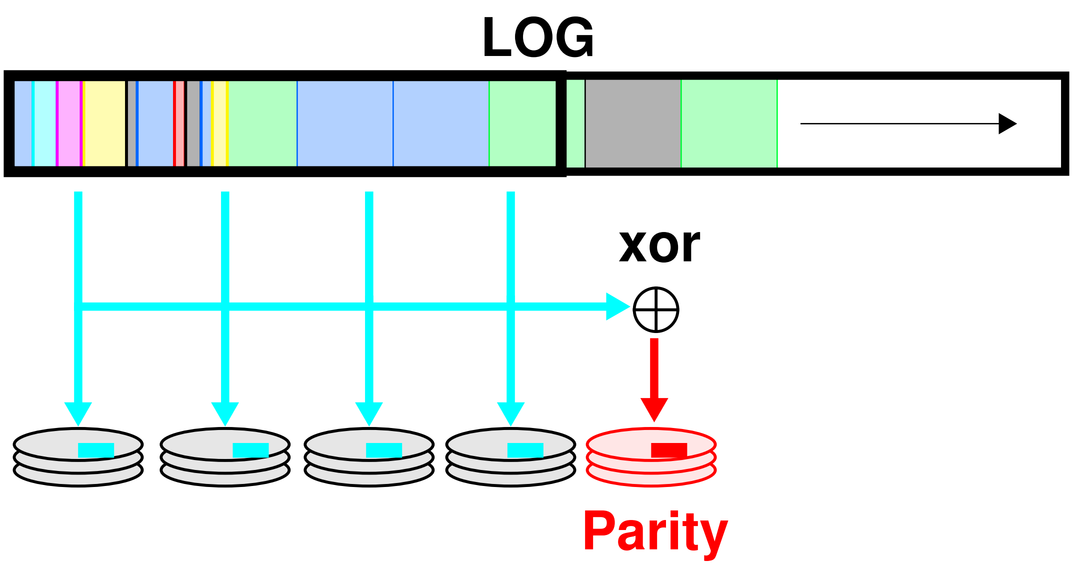
Cache integrated with Sawmill: consistency



Alternatives: No cache, treat as client

Log-structured file system

- Data written to append-only log
- Log is only copy of data
- Cleaner does garbage collection



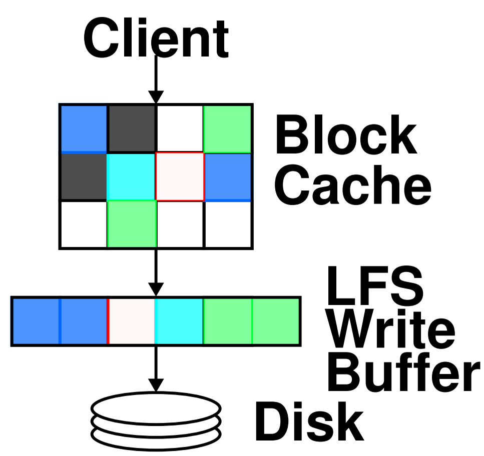
LFS is good with RAID:

- Small writes are collected into log
- No random writes
- No expensive parity updates

Write layout

- Data blocks placed into log segment
- Full segment written to disk

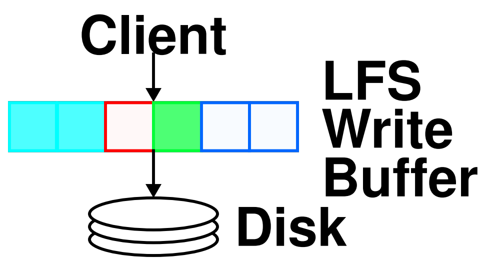
Previous work: Blocks written into cache first.



Problem:
- Two operations per block: high overhead

On-the-fly layout

- Incoming writes assigned a position in the log.
- Blocks received into log directly from network.
- Full log segment written to disk.



Problems:
- Can’t reorganize data
- Stale data may be written unnecessarily
- However, clients can do caching

Advantages:
- 1 operation per request, not 2 per block
- Lower per-block cost from batching
- Lower latency to disk

Summary

Sawmill high-bandwidth file system:

- Uses LFS for efficiency with RAID
- Takes advantage of RAID-II data path

Implementation:

- Large disk transfers through batching
- Streaming instead of caching
- New log layout technique

Outline

Background

The Sawmill file system

Performance

Conclusions

Measurements

Test setup:

- Sun 4/280 file server (8 SPECmarks)
- Array of 16 IBM 0661 disks
  320 MB each
  1.5 MB/s read bandwidth, 1.1 MB/s write
  12.5ms avg seek, 7ms rotational latency
- 8 SCSI strings on 4 Cougar controllers
- Data striped in 64 KB blocks (15+parity)

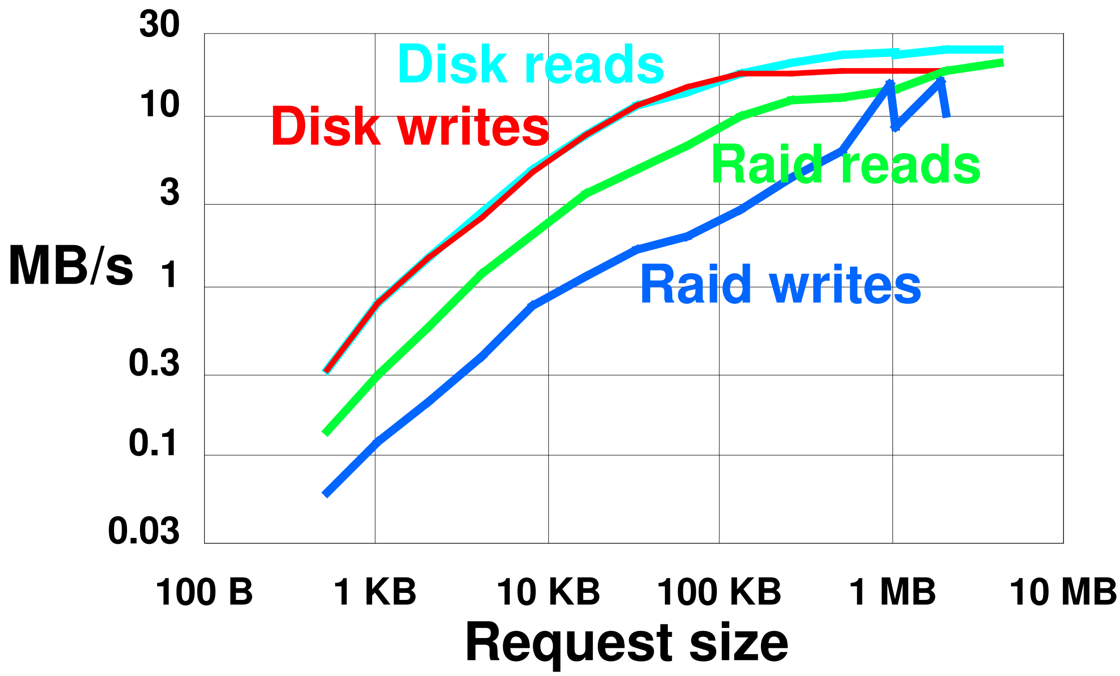
Benchmark:

- Fixed request size
- Random offsets into large file
- Transfers to controller memory, not network

Hardware performance

Measurements without file system:

Disk array: requests to 16 individual disks
RAID: 8 request streams to RAID device



Disk array: Reads to 24 MB/s, writes to 18 MB/s

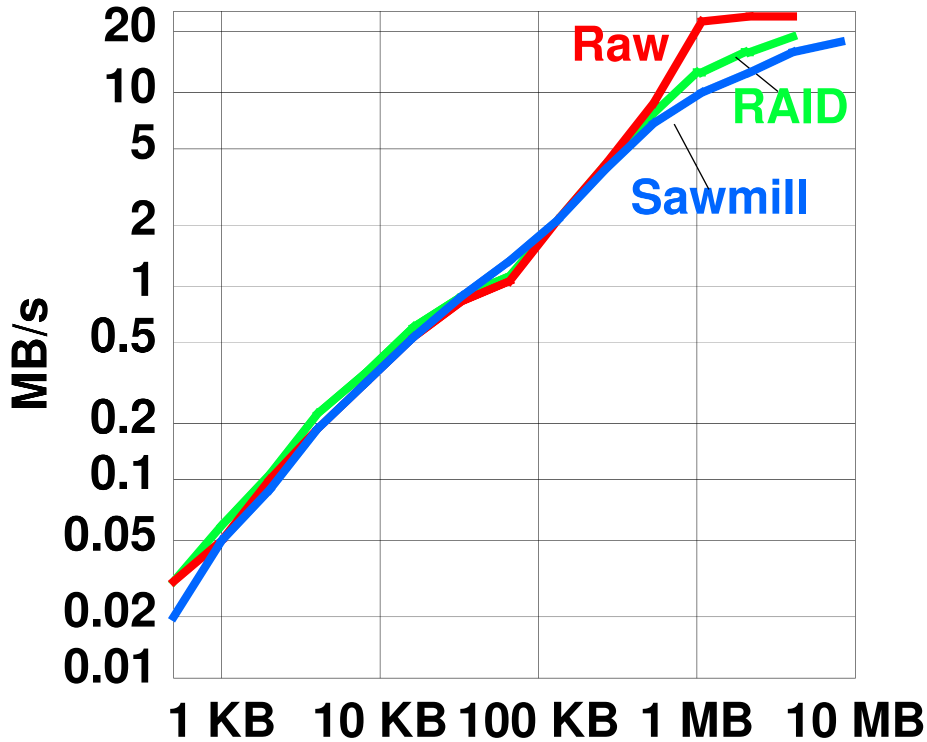
- Small requests positioning limited

RAID: Reads to 21 MB/s, writes to 16 MB/s

- small requests CPU limited

Single client reads

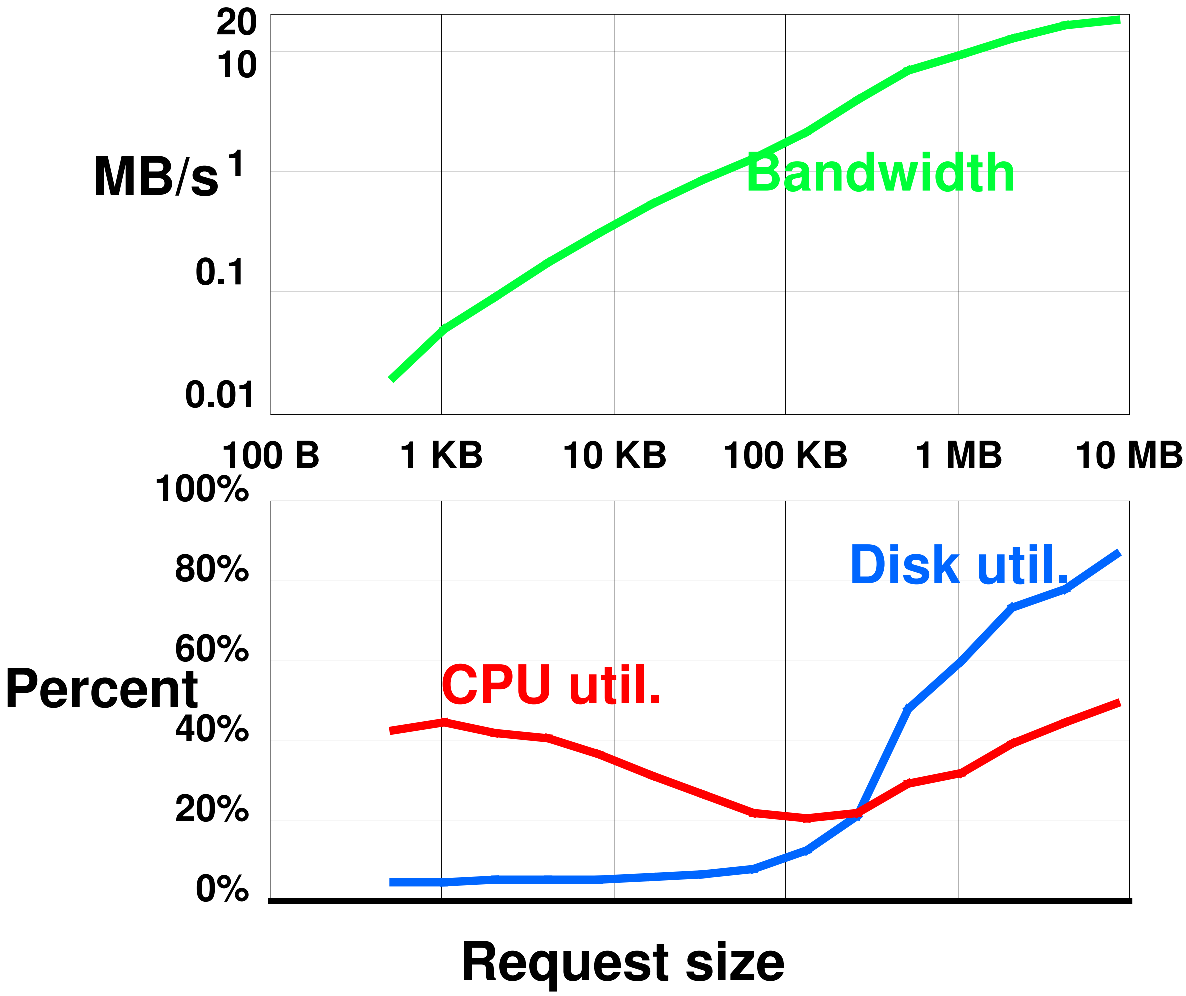
Single request stream: raw disk, RAID, Sawmill



- Small requests: positioning limited
- Large requests: overheads important
- Sawmill peak performance: 85% of raw

Multiple clients: Sawmill limited by CPU

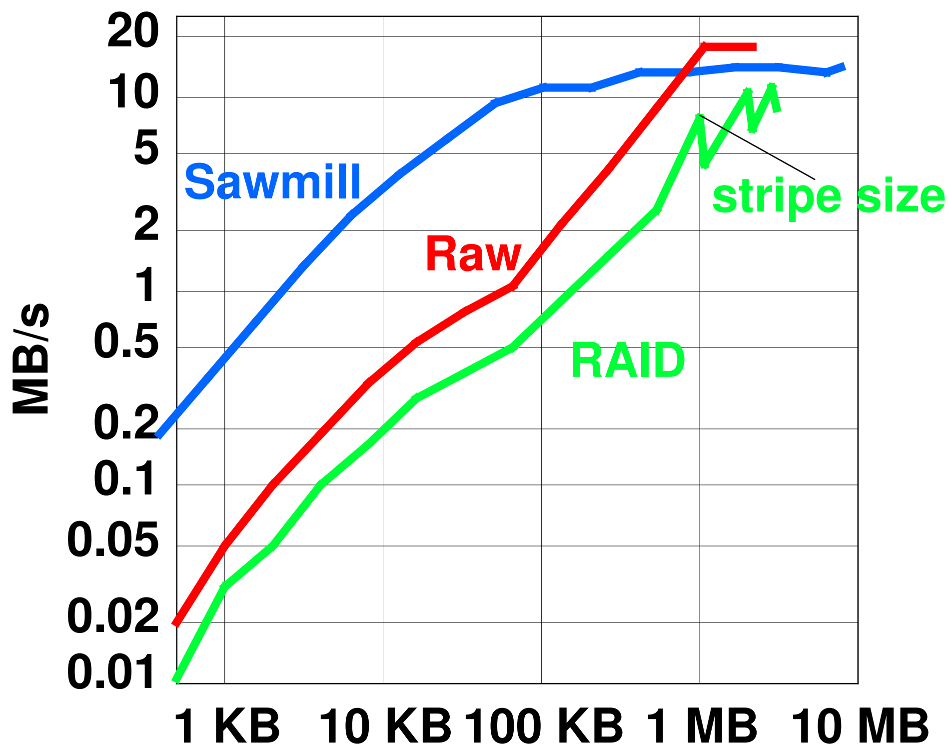
Analysis of reads



- Small reads are dominated by latency (positioning).
- Large reads approach raw disk bandwidth

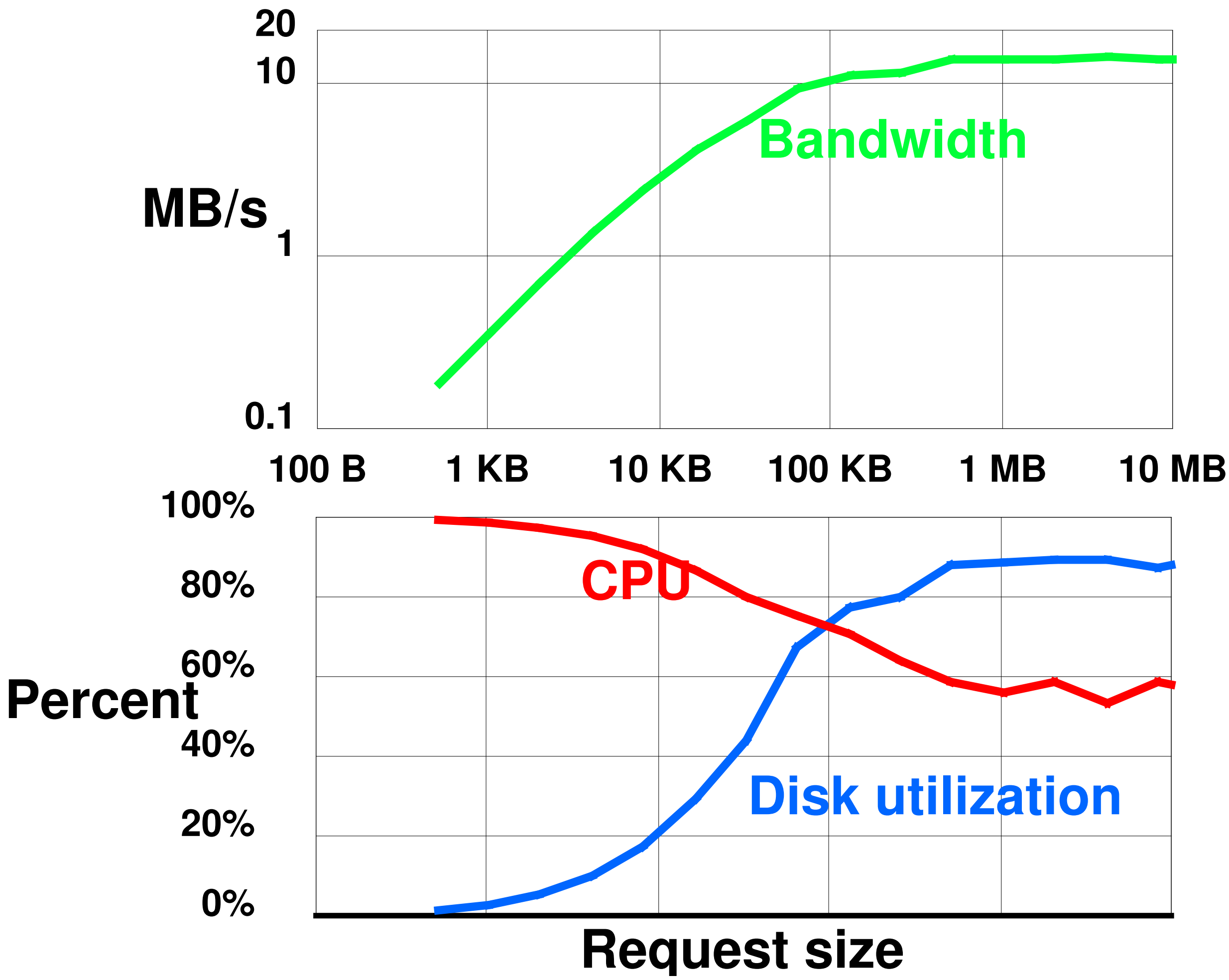
Single client writes

Single request stream: raw disk, RAID, Sawmill



- RAID slower than raw from parity
- Sawmill order of magnitude faster: LFS
- Small writes limited by CPU, not disk

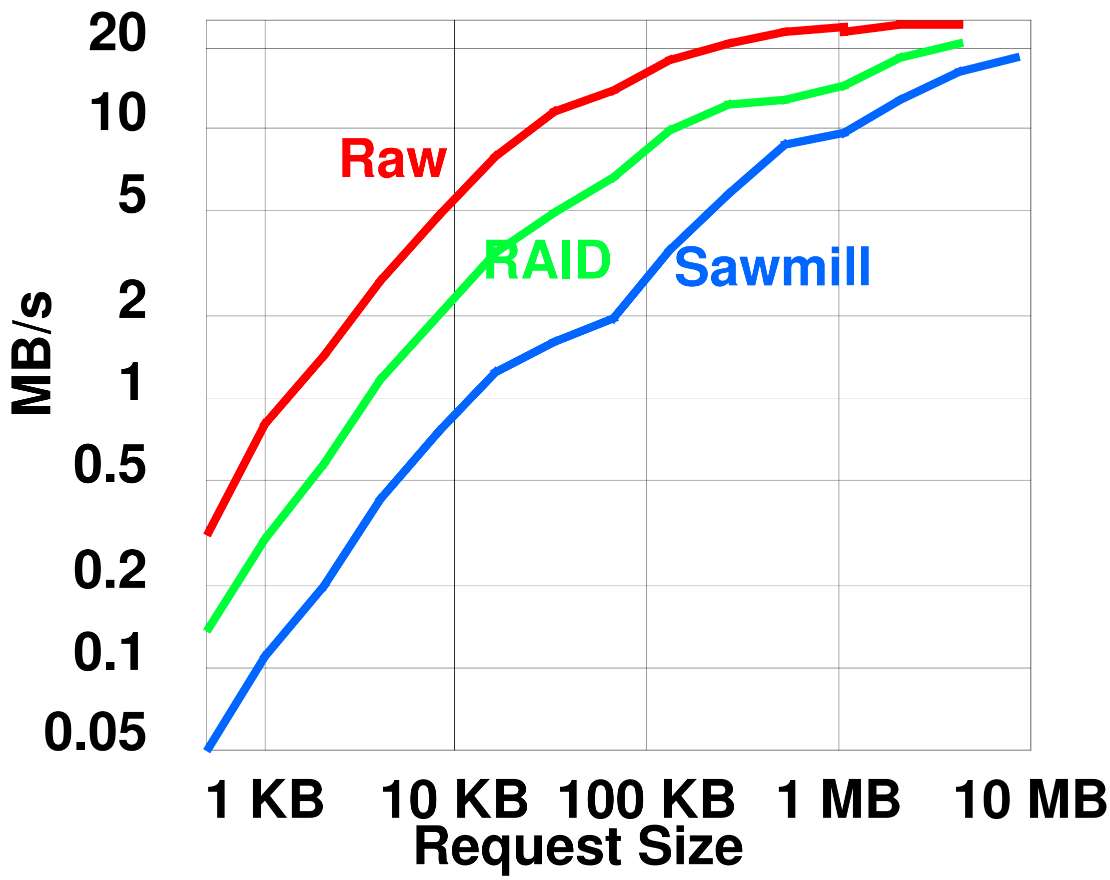
Analysis of writes



- Small writes are CPU limited; 360 ops/sec
- CPU usage remains fairly high.
- Large writes are disk limited

Concurrent reads

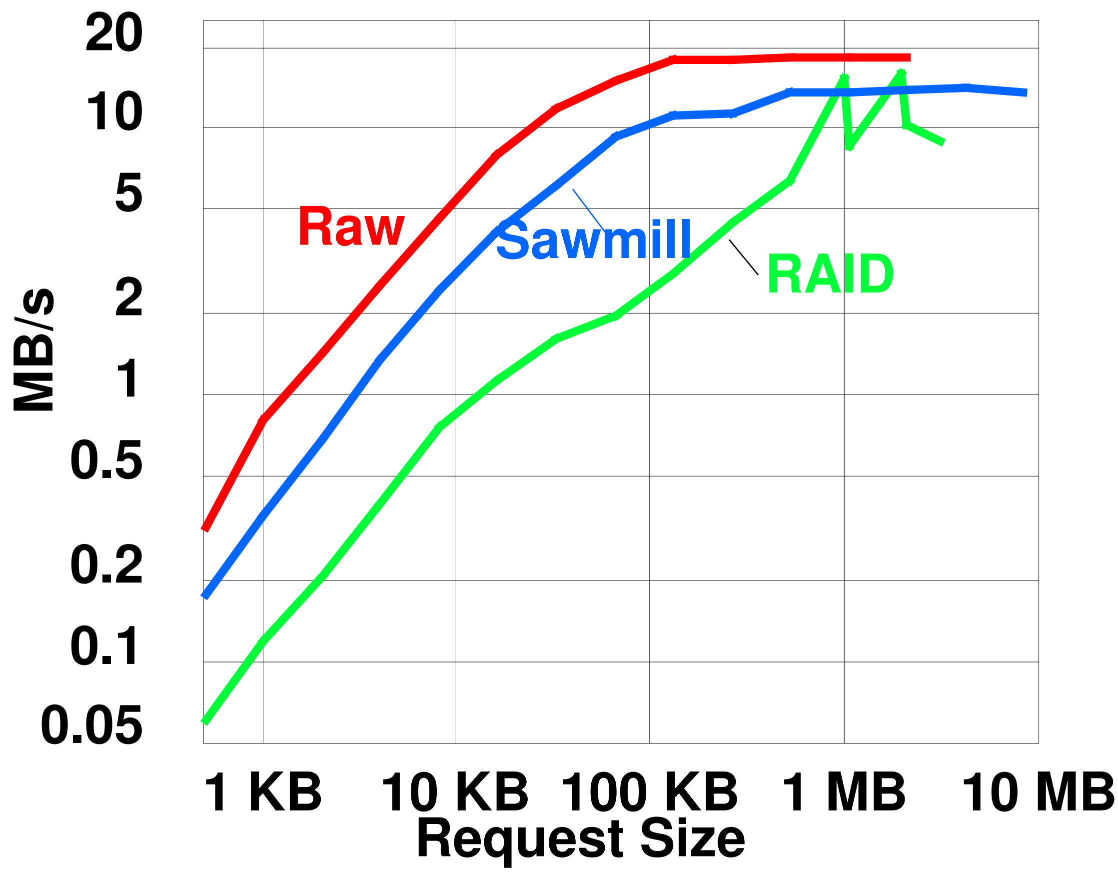
Concurrent requests: raw disk, RAID, Sawmill



- Concurrency limited by CPU
- Sawmill: only 3 concurrent requests

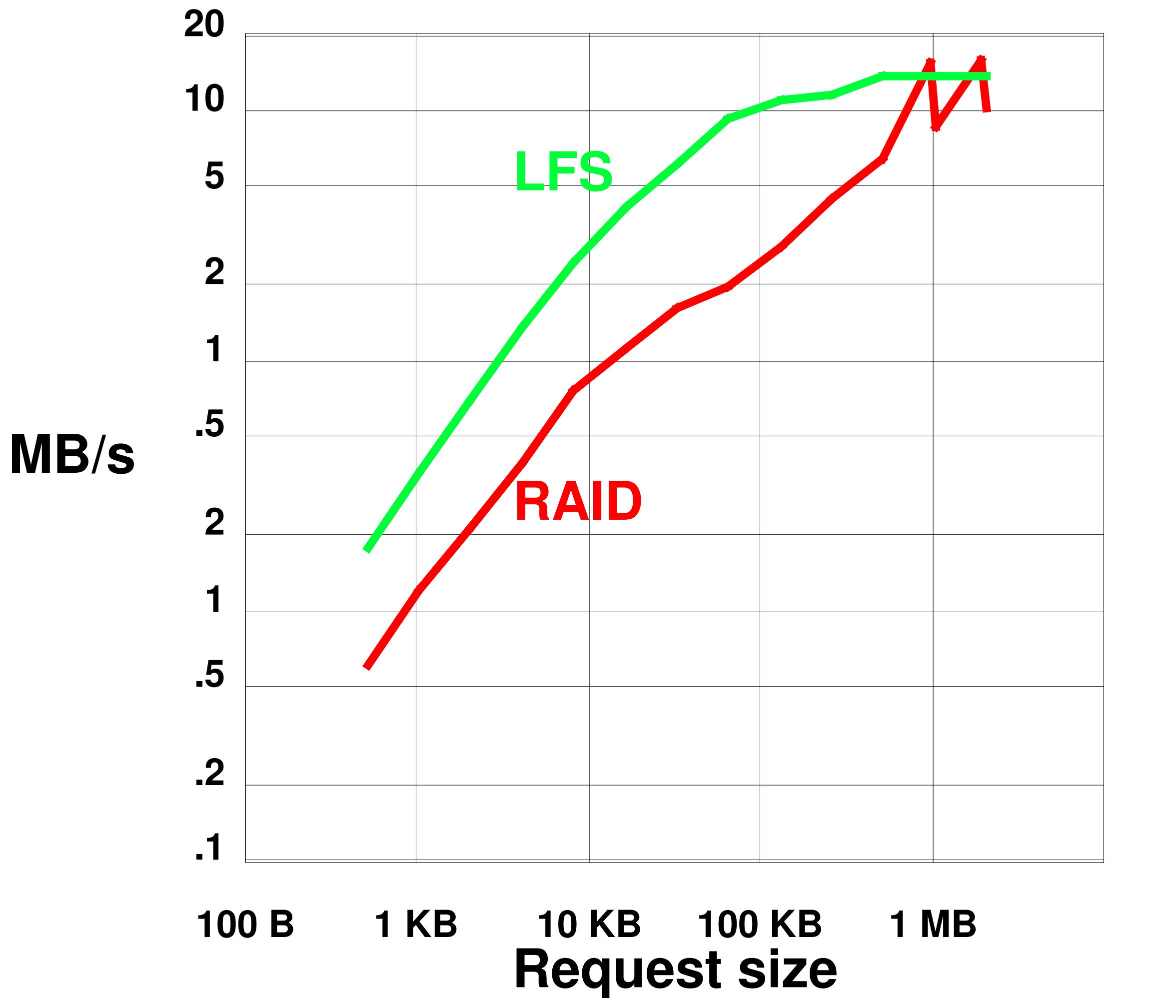
Concurrent writes

Concurrent requests: raw disk and RAID



- Sawmill inherently parallel
- Concurrency limited by CPU load
- Sawmill 3 times RAID performance

Writes: Benefit of LFS



- Small LFS write 3x faster than RAID device
- Speedup from grouping writes
- Small files are CPU limited, otherwise even faster.

Bottlenecks and scaling

- Large reads and writes limited by disk bandwidth.
- Small writes limited by CPU.
- Small reads limited by seek time.
- Concurrency limited by CPU.

Future trends:

- More disks: large requests improve
- Faster CPU: small writes and concurrency improve
- Small reads unlikely to improve:
  Need prefetching

Analysis

Performance of Sawmill file system shows:

- High performance with RAID-II architecture:
  20 MB/s reads, 15 MB/s writes
- Fast data path: 20 MB/s vs. 2.5 MB/s without
- LFS improves small write performance x3
- CPU load still high, dominates for small requests.
- Disk latency important factor

Related Work

High bandwidth storage systems

- Stripe across multiple servers:
  Zebra (Hartman), Swift (Cabrera)
- Mainframe server: Los Alamos, LINCS, MSS-II
- Multiprocessor: Auspex, DataMesh (Wilkes)

Avoiding RAID parity overhead:

- Parity logging (Gibson et al)
- Floating parity (Menon et al)

Grouping disk writes:

- Logical disk (de Jonge et al)
- Extent-like File System (McVoy et al)

Flexible file to block mapping:

- WAFL (Network Appliance)
- Loge (English et al)
- Mime (Chao et al)

Future work on Sawmill

Overhead of cleaning

- Garbage collection of freed data blocks.

- Cost depends on access patterns:
  Low for office workloads (Rosenblum).
  High for transaction processing (Seltzer).

Effects of prefetching

- Improve read performance

Performance under real workloads

Faster partial writes

- Partial writes necessary for `fsync`

- Log structure: can reduce parity overhead.

Contributions

Sawmill high-bandwidth file system:

- New log layout technique
- High bandwidth from workstation server
  20 MB/s with a Sun 4
- Manages separate control path
  Metadata caching
- Cooperating controller and server memory
  Streaming vs. caching
- Implementation of LFS on RAID
  Small writes 3 times faster

Conclusions

Sawmill provides high-bandwidth network access to the RAID-II disk array.

Sawmill performance:

- 21 MB/s reads, 15 MB/s writes

Lessons:

- LFS greatly improves small write performance
- Fast data path helps: 21 MB/s vs. 2.5 MB/s without
- CPU still important
- Small reads still a problem: limited by positioning