CSAR: Cluster Storage with Adaptive Redundancy

Manoj Pillai, Mario Lauria

Department of Computer and Information Science

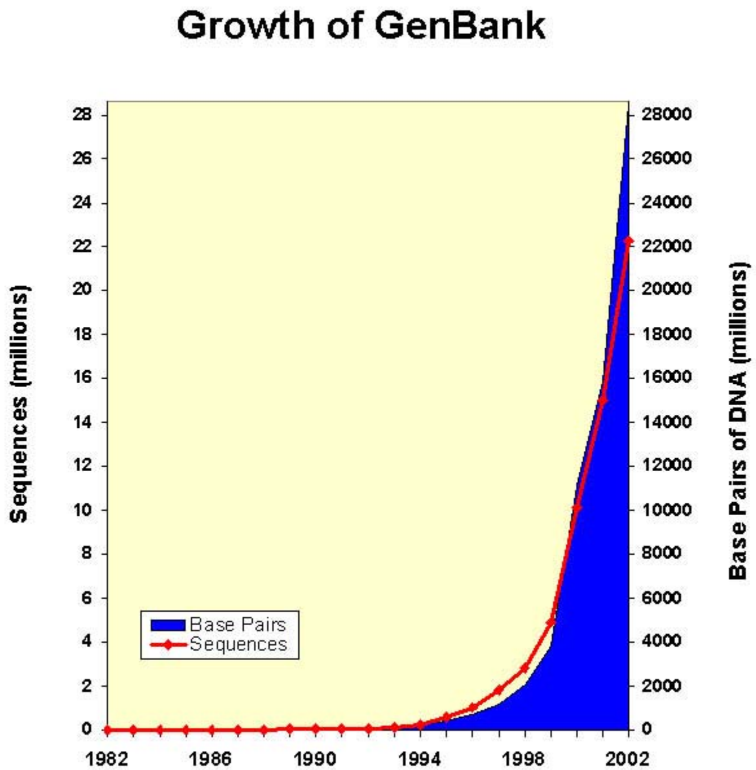
The Ohio State University

Outline of Presentation

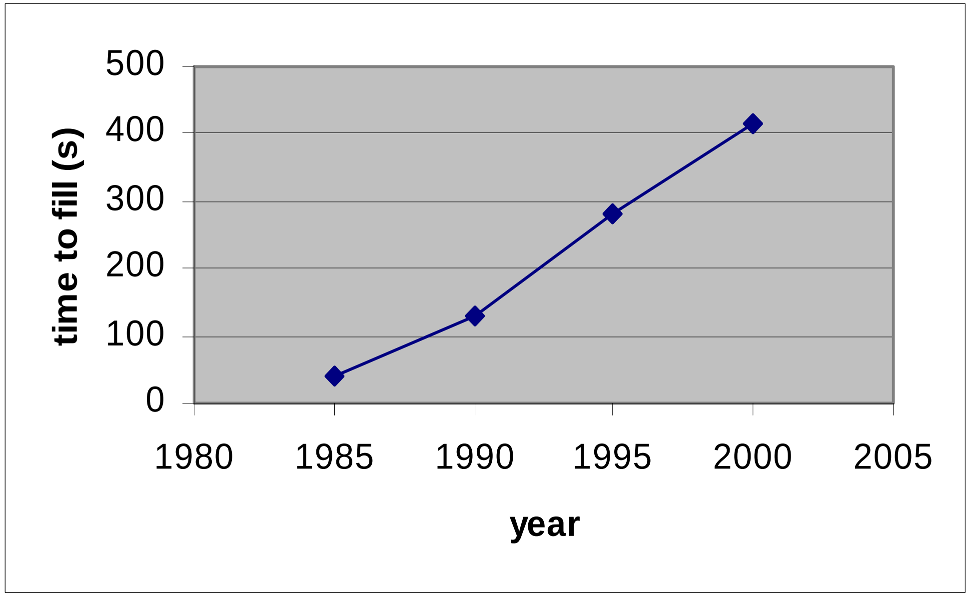
- Motivation
- Parallel Virtual File System (PVFS) and CSAR Overview
- Implementation of Redundancy schemes
- Performance Results
- Conclusions

Motivation: Application Trends

- The nature of HPC applications is changing from simply compute-intensive to compute *and* data-intensive.
- Data sets are growing faster that Moore's law
  - GenBank growth over the last few years: 1.7x/yr
  - Moore's Law: 1.6x/yr



Motivation: Technology Trends



- Storage has been growing faster than bandwidth
  - graph shows time to completely fill largest disk for each year (historical data from http://www.cs.utexas.edu/users/dahlin/techTrends/)
- Moving large data sets in and out of a storage server efficiently is the main challenge

Motivation

- These trends result in persistent mismatch between I/O and other aspects of cluster architecture
  - compare 10’s or 100’s of MB/s throughput of a typical NFS server vs. gigabit/sec communication, gigaflop processors, and gigabyte memories
- Striped file systems are a possible remedy
  - can leverage disk access parallelism in a cluster
  - but sensitivity to single disk failure is also their achille’s heel
- There is a critical need for **high-bandwidth** and **reliable** storage in commodity clusters

The main idea

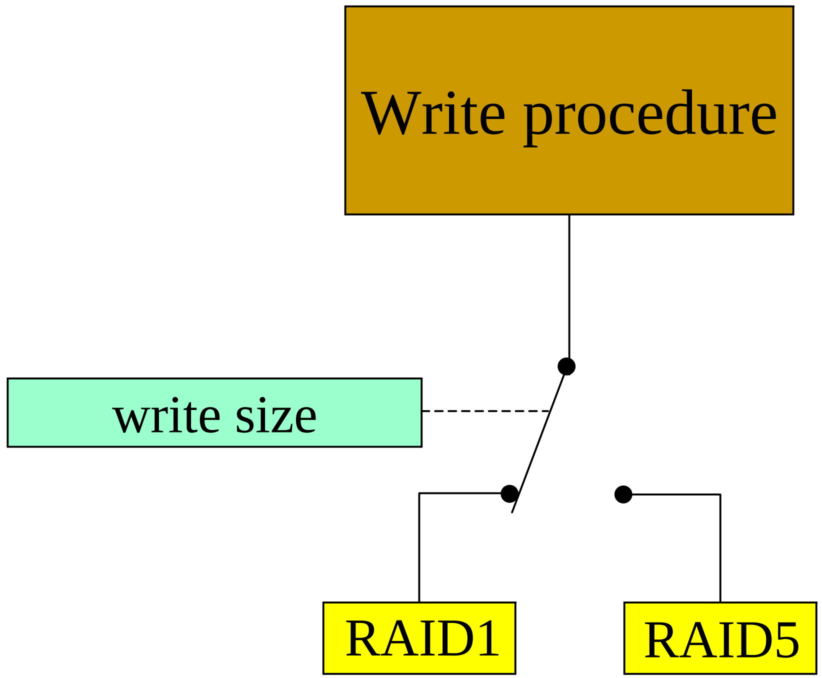
- It makes sense to augment a striped file system with some form of redundancy
  - striping + redundancy => performance + fault resilience
- However coming up with a form of redundancy that does not adversely affects performance is not a trivial task
- Our approach: select the most favorable redundancy scheme on the fly
  - compared to basic schemes, we get:
    - best of all worlds in terms of bandwidth
    - intermediate performance in terms of storage utilization

Problems with RAID1/RAID5

- RAID1: redundancy through mirroring
  - problem: has poor bandwidth for large transfers
- RAID5: redundancy through parity
  - problem: has high latency, low bandwidth for small writes
- Also, the distributed implementation of RAID5 adds a consistency issue
  - to maintain consistency, some form of synchronization is needed between writes to elements of the same stripe

Our Hybrid Scheme

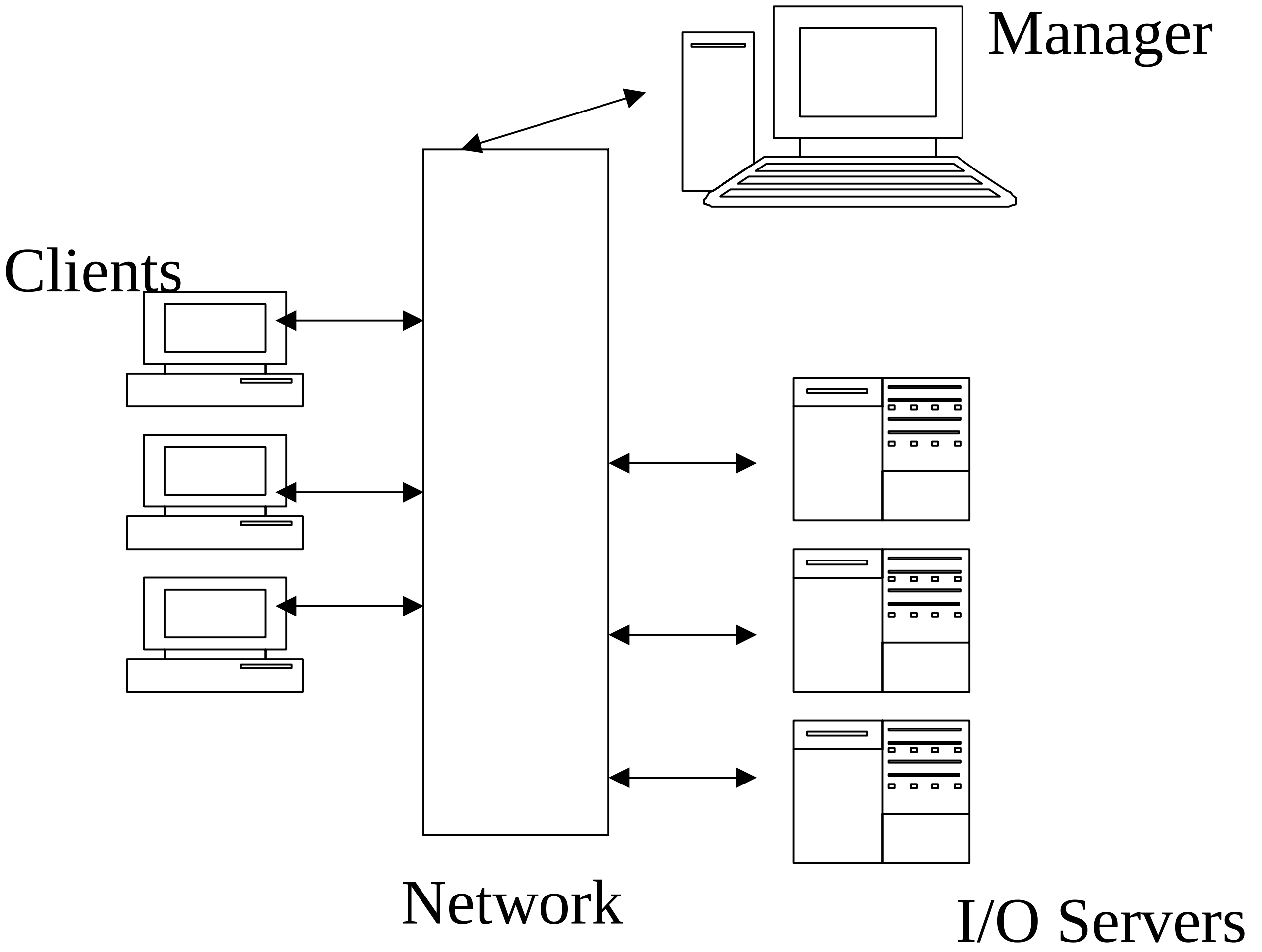
- Dynamically select the best redundancy scheme:
  - parity for full-stripe writes,
  - mirroring for partial-stripe writes
- Disk layout is similar to RAID5, but with an additional overflow file to hold 2nd copy of data for partial-stripe writes
  - a table listing overflow regions is maintained at each server



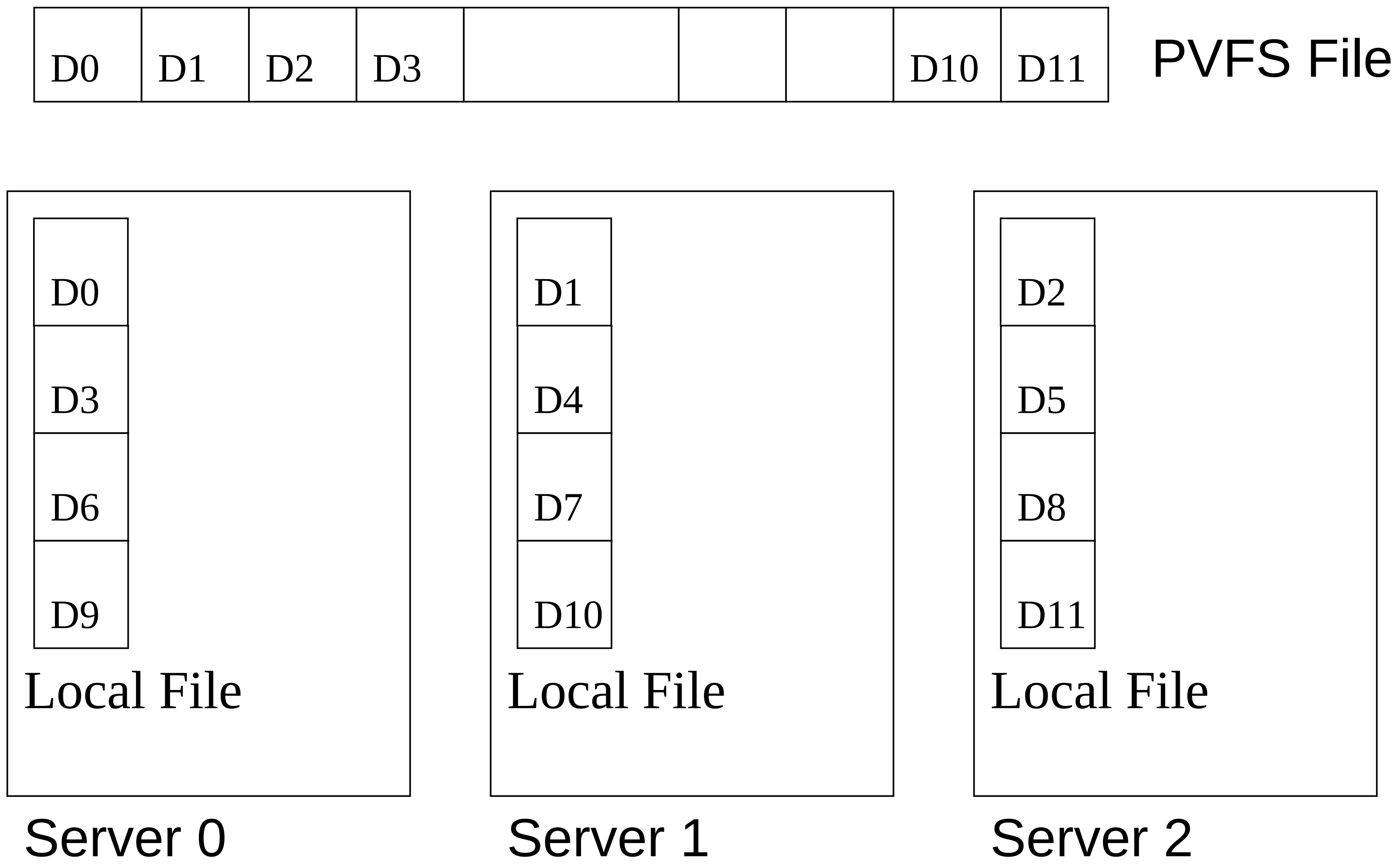
CSAR Overview

- We built a proof-of-concept implementation of our scheme using a general purpose striped file system
- PVFS was chosen for its availability and popularity
  - native, MPI-IO and Linux kernel interfaces available to programmers
- We augmented PVFS with three redundancy schemes - RAID1, RAID5 and Hybrid
  - simple locking scheme takes care of RAID5-related consistency
  - comparative analysis using popular benchmarks
- A number of lessons learned using a real system as opposed to a simulation …

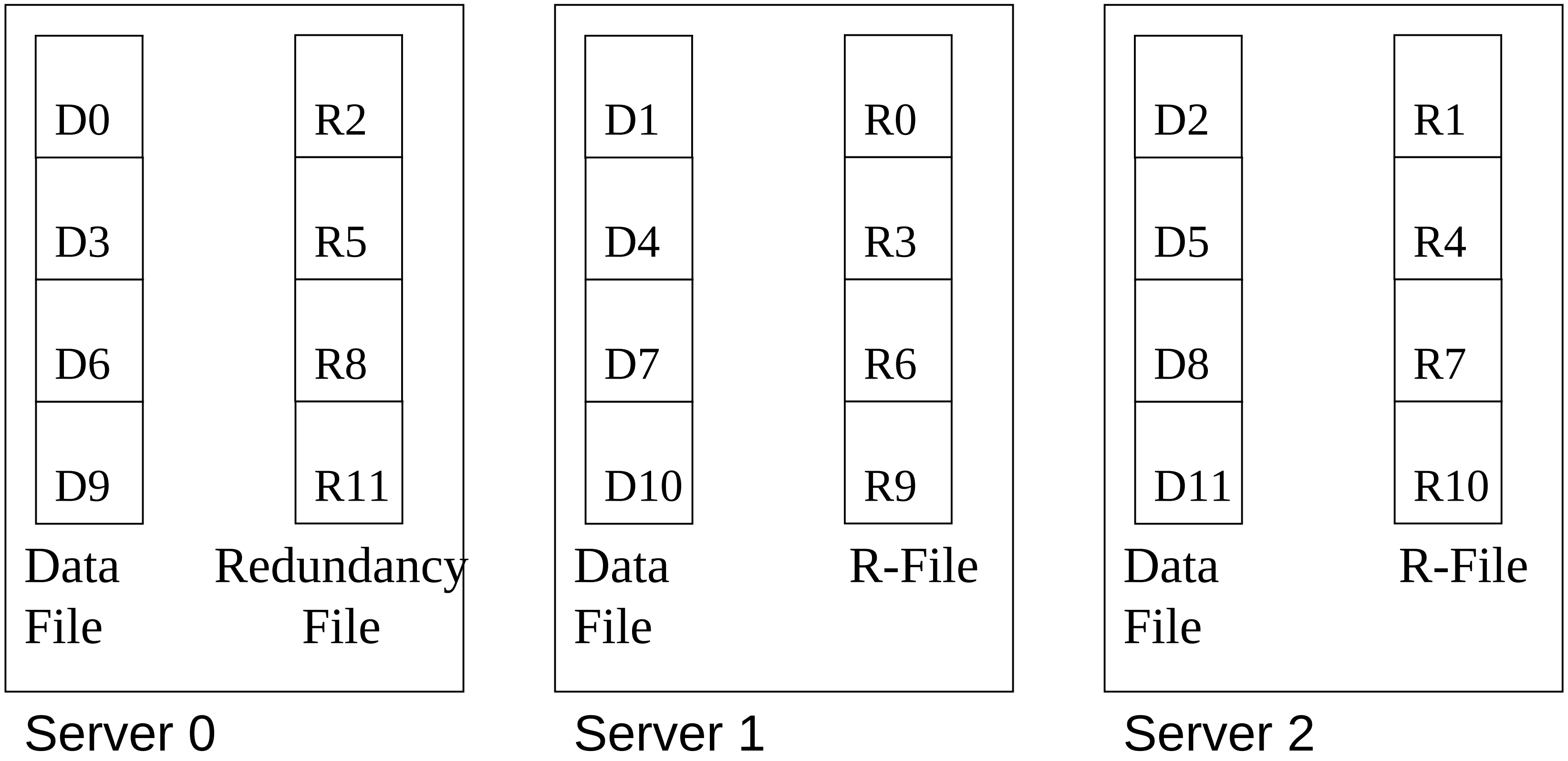
PVFS Overview



Native Data Layout in PVFS

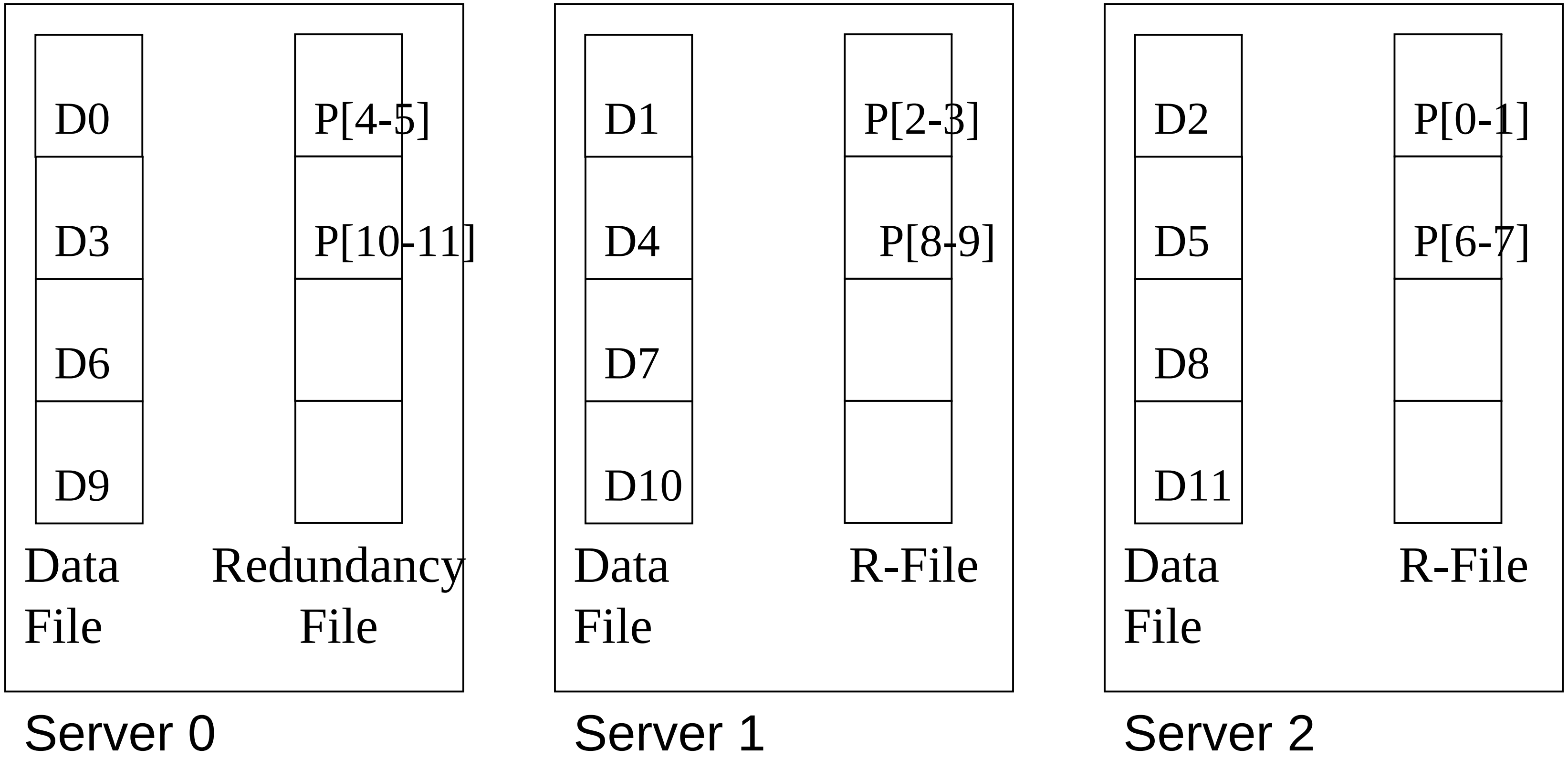


Data Layout in PVFS + RAID1



- Uses an additional file at each server to store redundancy
- *Data* blocks distribution identical to PVFS

Data Layout in PVFS + RAID5

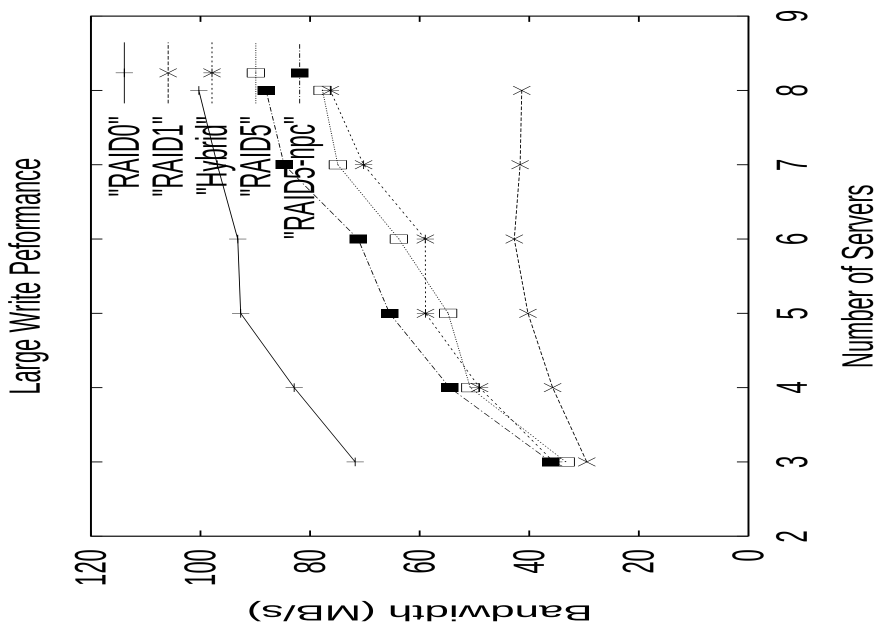


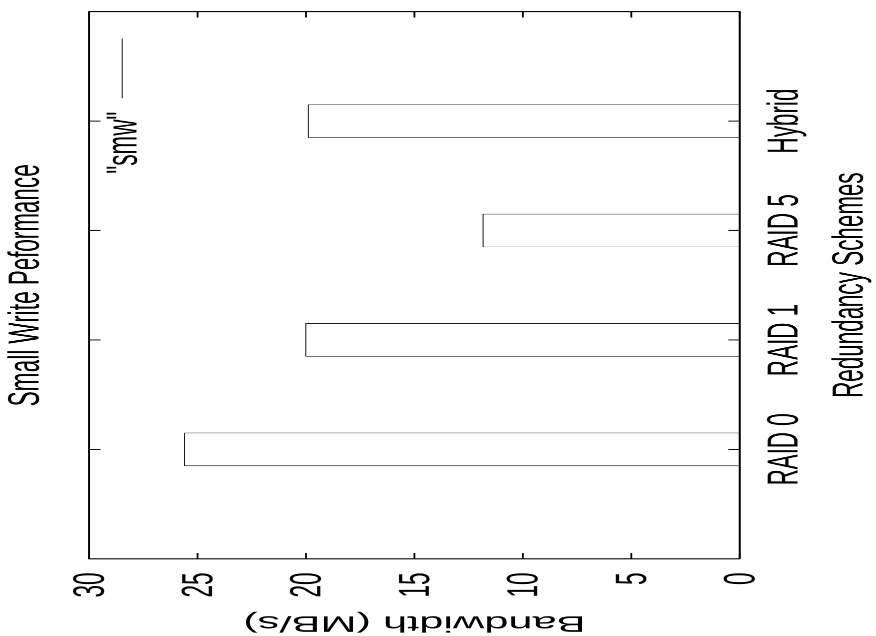
- Hybrid's layout adds a third "overflow" file used to write the second copy of blocks (partial-stripe writes)

Experimental setup

- We used two clusters for our experiments
  - 8x dual 1GHz Pentium III, 1GB RAM, Myrinet network, high performance disks (2x IDE connected to striping 3Ware controller)
  - 64x dual 833MHz Itanium cluster at OSC
- Benchmarks used
  - microbenchmarks for small write vs. large write analysis
  - ROMIO - parallel I/O
  - BTIO (class B and C), FLASH I/O - data intensive applications

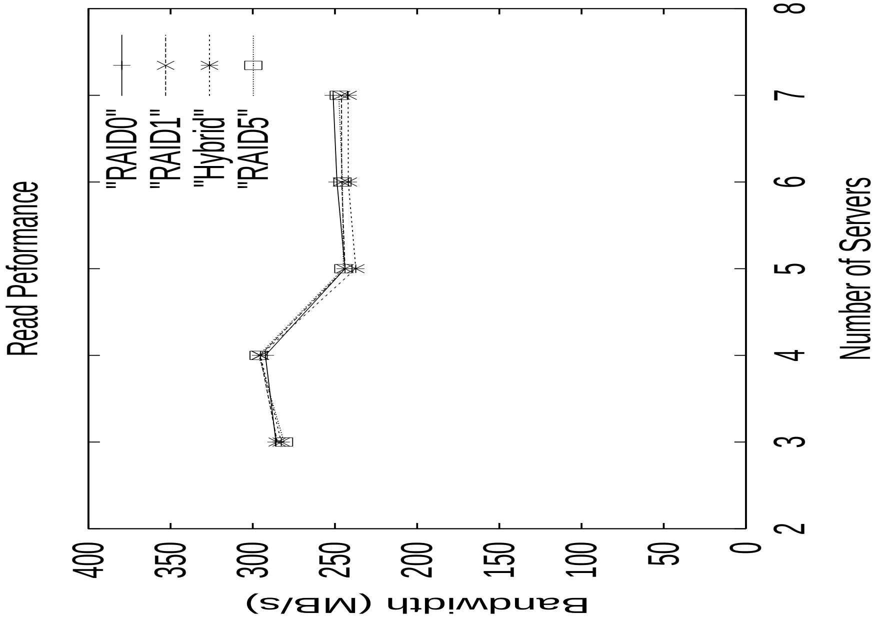
Large and Small Write Performance

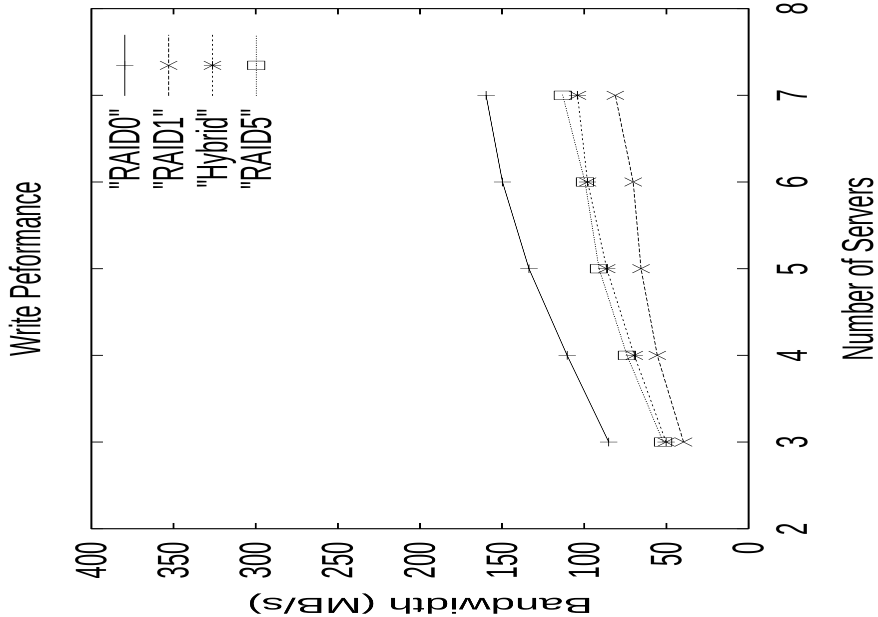




- Overhead of parity computation is low
- Hybrid performance is comparable to RAID5 for large writes, and to RAID1 for small writes

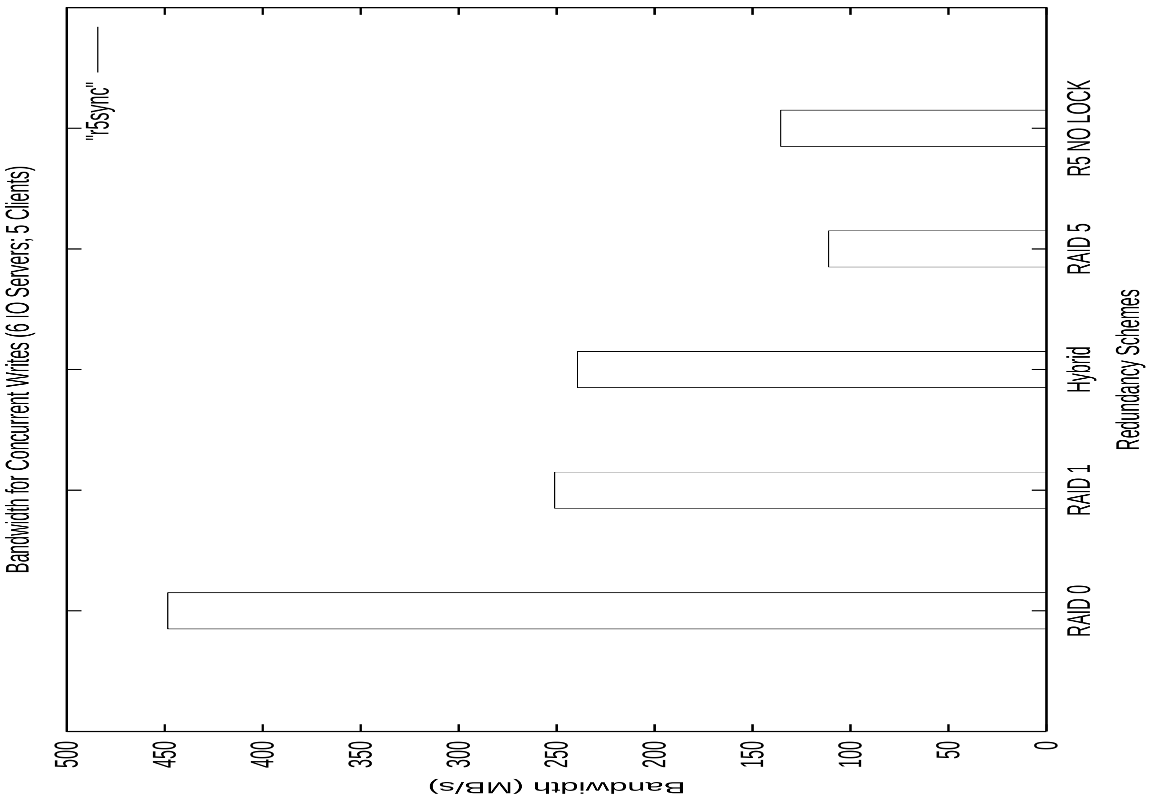
ROMIO-perf Benchmark





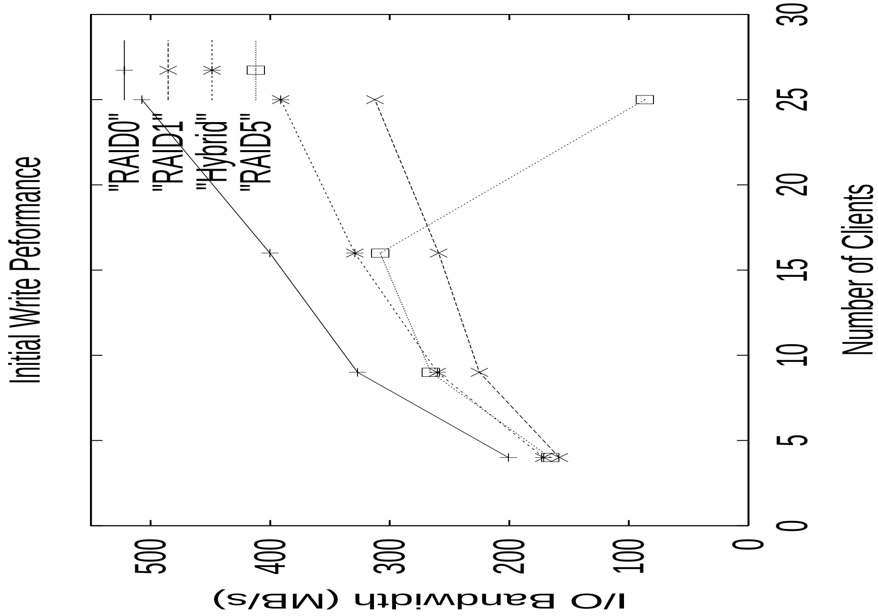
- Read performance is nearly identical for all schemes.
- Write performance is similar to large write micro-benchmark

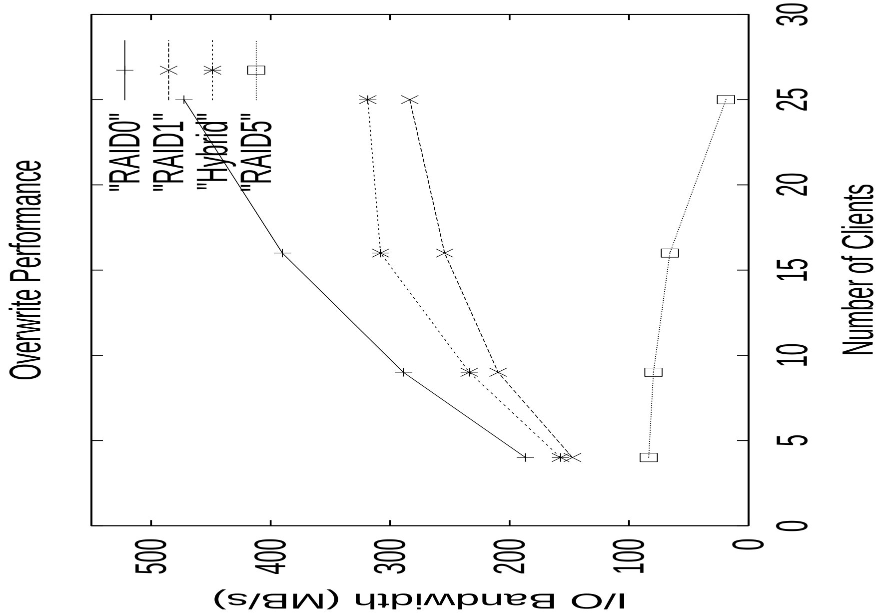
Locking Overhead in RAID5



- Locking further affects the partial-write performance of RAID5; 20% in this benchmark compared to a non-locking version.

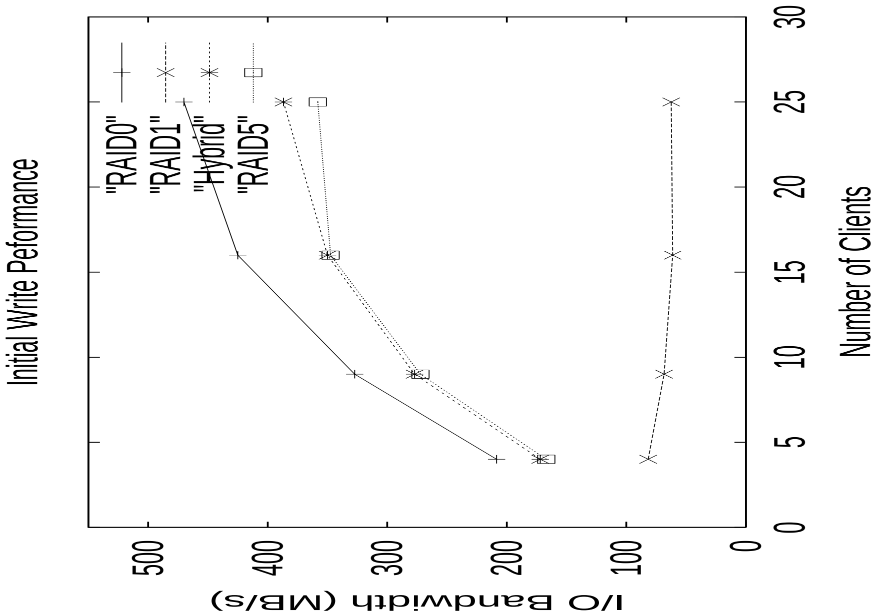
BTIO Class-B Benchmark

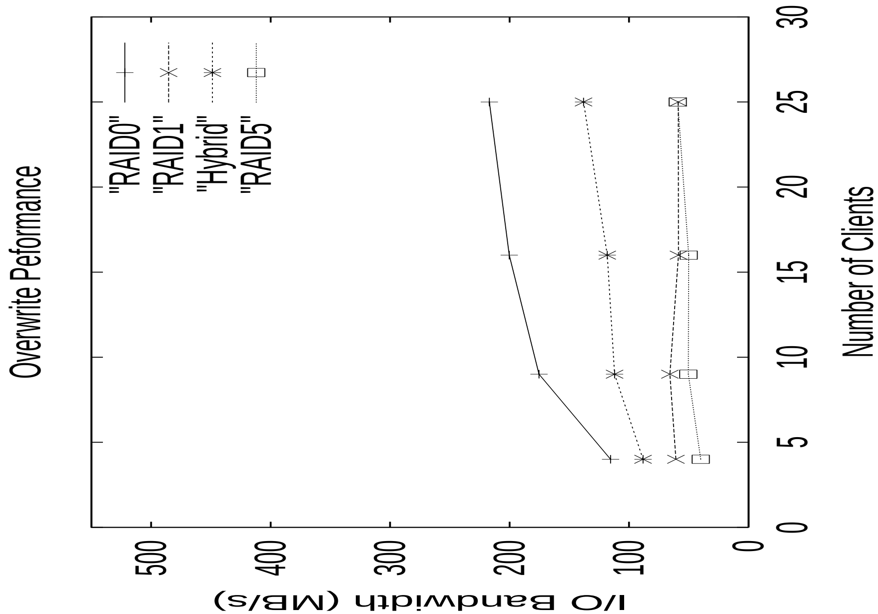




- In both cases, Hybrid outperforms the other two schemes.
- For overwrite with no caching, RAID5 much worse.

BTIO Class-C Benchmark





- RAID1 performance suffers because caches at the servers overflow

Normalized Application Performance

- Hybrid is comparable to or better than the best of RAID1 and RAID5

Normalized Storage Requirement

- Storage allocation in Hybrid is not optimized. Still, except for FLASH-IO, it is better than or equal to RAID1.

Conclusions

- Both RAID1 and RAID5 exhibited the anticipated performance problems in our benchmarks
- Hybrid scheme provides good performance for a range of workloads
- Storage requirement of the Hybrid scheme is application specific, in most cases is intermediate between RAID1 and RAID5
- Our approach is justified by technological trends that put bandwidth at a premium over storage

Related Work

- Petal/Frangipani
- RAID-x
- HP AutoRAID
- Zebra, xFS