# Serverless Network File Systems

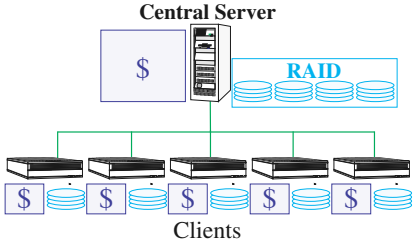
**Mike Dahlin, Tom Anderson, Jeanna Neefe, Drew Roselli,
Dave Patterson, Randy Wang
University of California, Berkeley**

**SOSP-15
December 4, 1995**





## Central Server Architecture




All Requests Go Through Central Server

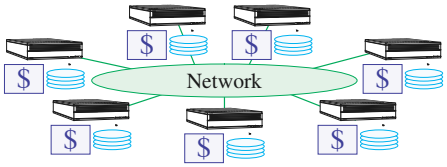
- Server Limits Performance
  - ➔ Buy Lots of Cache and a Hardware RAID
- Server Limits Scalability
- Server Limits Availability

Powerful Clients





## Serverless Network File System




Technology:

- Capable, Cheap Microcomputers
- Fast, Scalable Network

Distribute Storage, Cache, Management

- Scalability
  Exploit Aggregate Resources
- Performance
  Striped Disk, Cooperative Cache, No Bottlenecks
- Availability
  Redundant Storage
  Interchangeable Components





## Outline

- Introduction
- ***Design of xFS***
- Results
- Conclusions

## Design of xFS

### Central Server = 4 Functional Pieces

- Disk Storage
- Disk-Location Metadata
- In-Memory Cache
- Cache Consistency Metadata

### xFS Design Based on 3 Distributed Modules:

- Disk Storage → Storage Servers
- In-Memory Cache → Cooperative Cache
- Disk-Location Metadata & Cache Consistency Metadata → Distributed Manager

### Key Design Principle:

- Location Independence
  Data, Metadata, Control

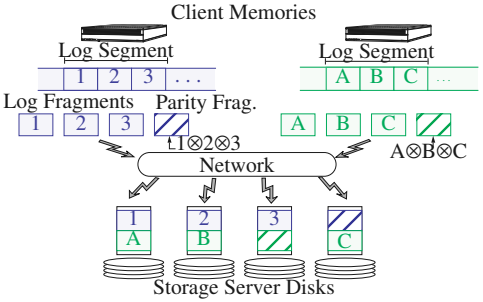




## Storage Servers

### Log-Structured, Distributed RAID

- Build on Zebra, LFS, RAID



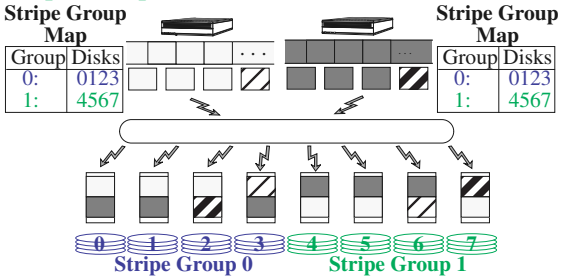

- Efficient Writes (LFS)
- Efficient Parity (Per-Client Striping)
- High Bandwidth (Parallelism of Striped Disks)
- High Availability (Redundant Storage, LFS)





## Storage Server Scalability

### Stripe Groups




- Improve Performance — Match BW of Disk and Net
- Improve Availability — More Redundancy

### Other Improvements v. Zebra

- Distributed LFS Cleaner
- Cooperative Cache, Distributed Manager





## Cooperative Caching

### Motivation:

- Remote Memory Faster Than Disk

### Eliminate Server Cache

- Coordinate Client Caches
- Clients Forward Data to One Another

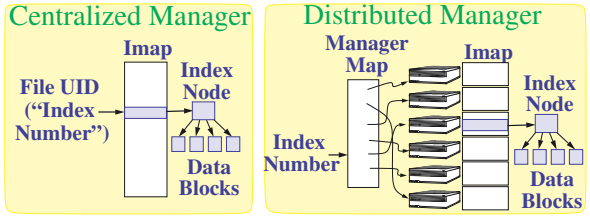
### Performance

- Improves Read Time ~2x
  Simulation Study: Sprite, Auspex Workloads

*Details: See (OSDI'94) or Feeley et. al (Tomorrow)*

## Distributed Manager

- Track Where All Blocks Stored




### Distributed Management:

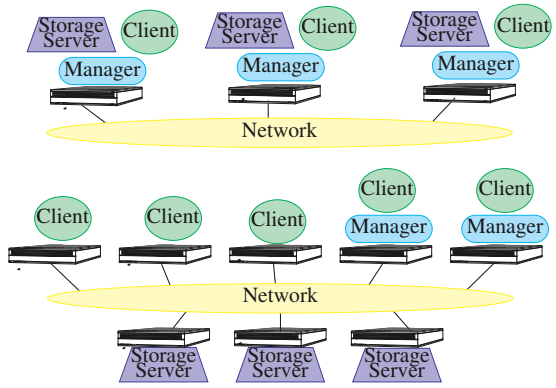
- Avoid Bottleneck
- Exploit Locality
- Dynamic Reconfiguration of Management

### Similar Structures For Cache Metadata





## High-Level Architecture



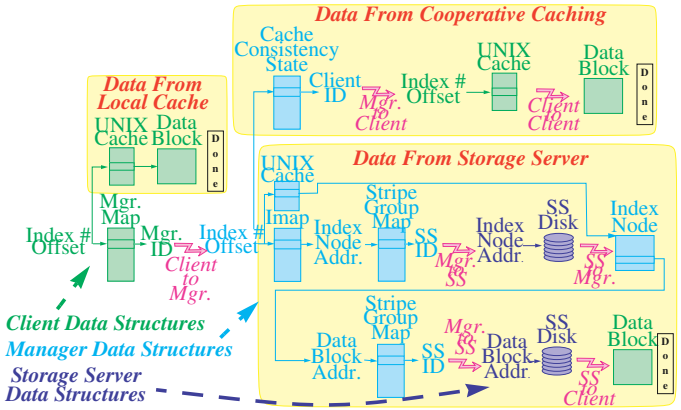

### Different Performance, Reliability, Trust

- Peer-to-Peer
- "Scalable Server"





## Example: Read




- 3 Main Paths
- Common Case: Avoid Disk, Few Net Hops





## Prototype

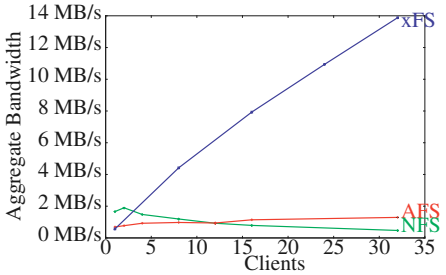
### Hardware

- 32 SPARCStation10's and 20's
- Myrinet
  Switched, 80 MB/s LAN

### Status: Evolving Prototype

- Performance Limitations
  Slow Network: RPC on TCP → Active Messages
  User-Level Daemons → All-Kernel Version
  General Tuning → Profiling, Tuning
- Functional Limitations
  Limited Recovery
  No Cleaner

## Performance Results: Write BW




### Each Client Writes Private, 10 MB File

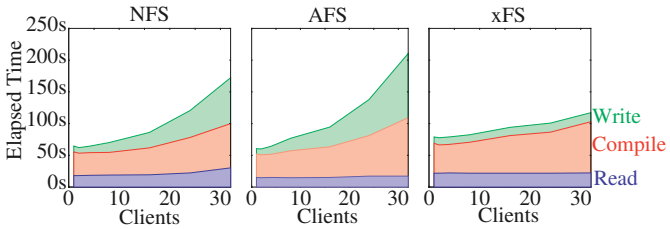
- 32 Storage Servers, 32 Managers
- One Client:
  NFS = 1.7 MB/s AFS = 0.7 MB/s xFS = 0.6 MB/s
- Peak:
  NFS = 1.9 MB/s AFS = 1.4 MB/s xFS = 13.9 MB/s

### Similar Results for Large Reads, Small Writes





## Performance: Andrew Benchmark




### Application-Level Benchmark

- "A Computer Scientist at Work"

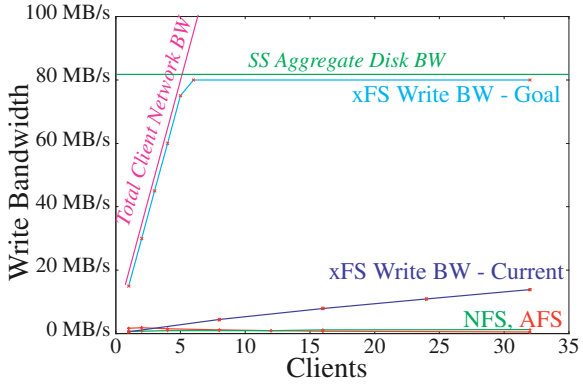
### Results

- One Client: NFS and AFS beat xFS
- Many Clients: xFS Beats NFS and AFS





## Future Performance Goals




- One/Few Client: Full Network Bandwidth
- Many Clients: Full Disk Bandwidth





## Conclusions

### xFS Designed to Avoid Bottlenecks

- Scalability
- Performance
- Availability

### Architecture: 3 Distributed Components

- Distributed Storage
- Cooperative Caching
- Distributed Management

### Promising Initial Results

***http://now.cs.berkeley.edu/Xfs/xfs.html***