

# Serverless Network File System

Thomas Anderson, Michael Dhalin,  
Jeanna Neefe, David Patterson, Drew  
Roselli, Radolph Wang, UC Berkeley

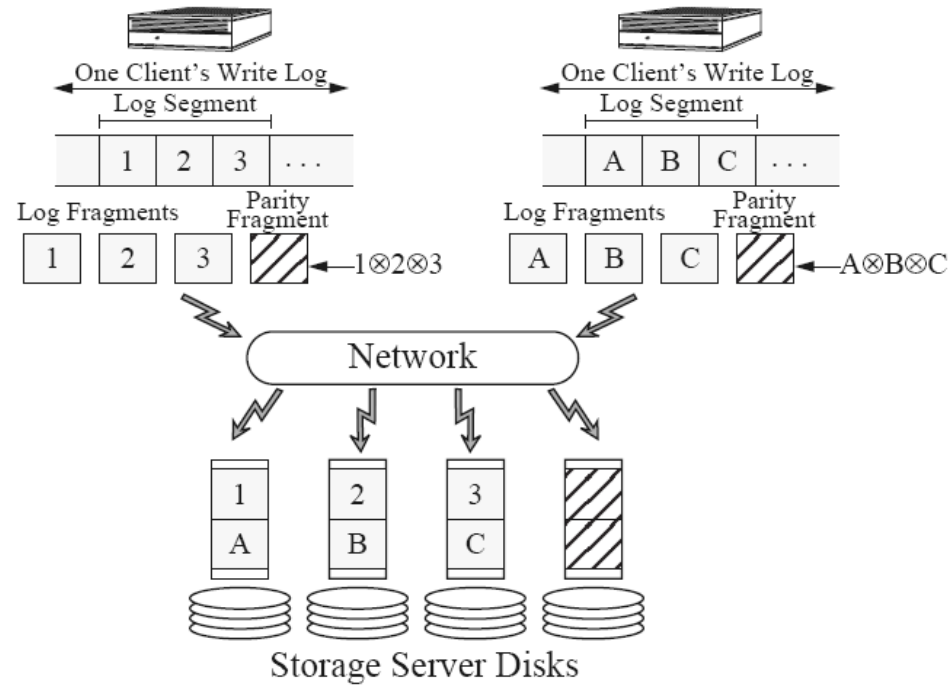
# xFS

- xFS idea: distribute file service functionality across all nodes in the system
  - “anything, anywhere” philosophy
  - all clients cooperate in providing file services
  - assumption: fast network, trusted environment; outside nodes with NFS
  - should be scalable, fault-tolerant, good performance...
- Issues with central server(s) approaches?

# Background ideas

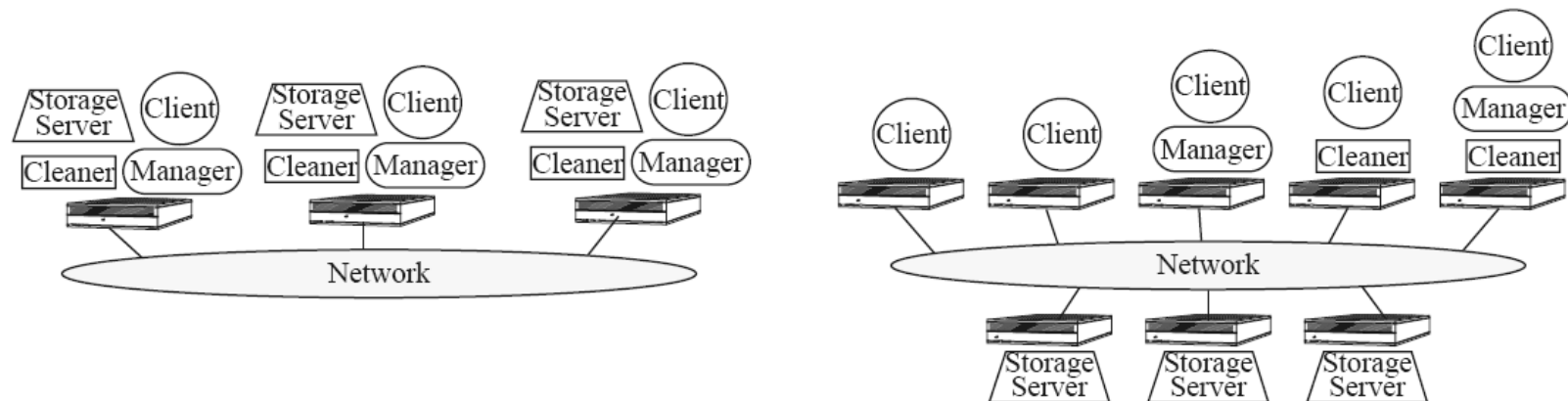
- RAID
  - Across stripe groups; groups small writes
- LFS
  - Helps recovery (checkpoints); must perform cleanup
- Zebra
  - Software RAID for LFS; use of *deltas* (atomic operations)
- Multi-processor Cache Coherency
  - Multiple cache managers for subsets of files

### Client Memories



# XFS entities

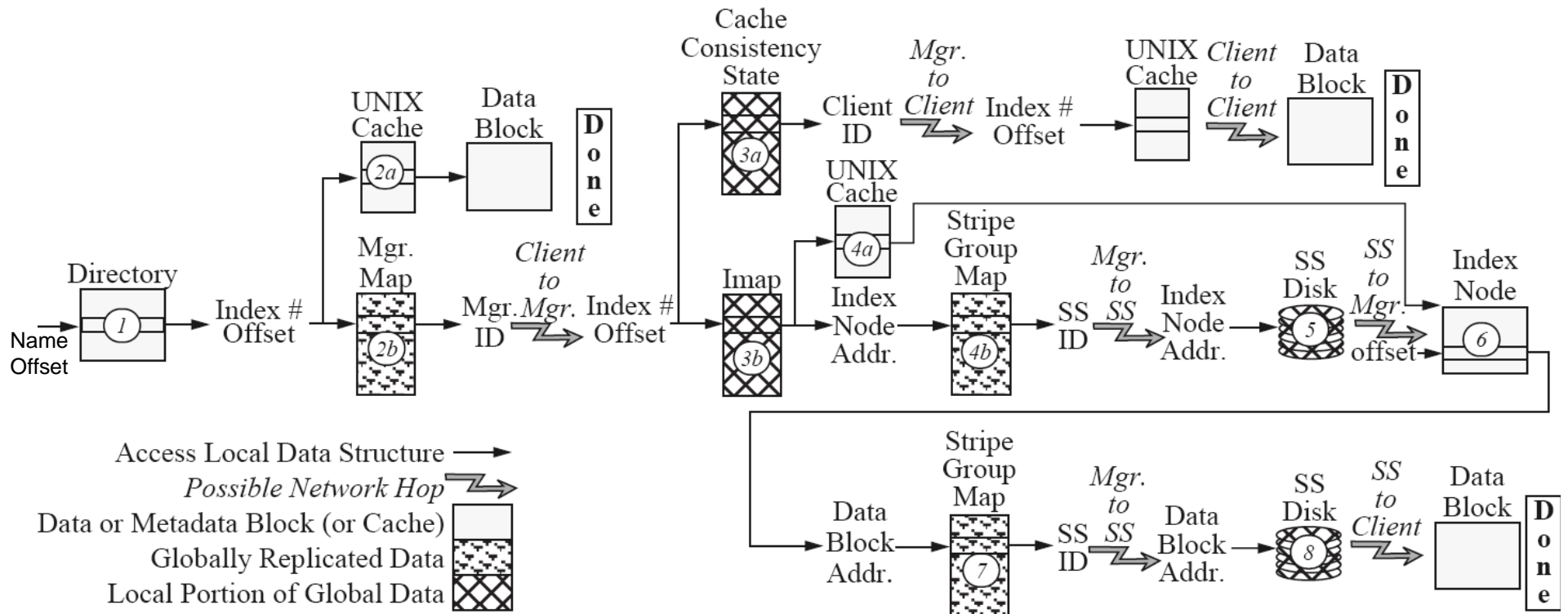
- Client – performs access
- Storage Server – stores data
- Manager – for cache consistency and file location
- Cleaner – fix LFS to remove old writes



# XFS Data Structures

- Manager Map
  - Index# -> manager; partitioning based on subset of bits in index#; can dynamically change
  - globally replicated
- Imap
  - Index# -> disk log addr of file inode
  - Partitioned across managers
- File Directory
  - File name -> index#
  - Index# selection based on load balancing or locality criteria (here locality)
- Stripe Group Map
  - Disk log addr (stripe group ID) -> list of storage servers;
  - globally replicated;
  - Support dynamic reconfig of servers; stripe group may be current or obsolete (cleaner will eventually remove obsolete entries)

# XFS Read



- Inodes cached at managers only
- Write creates new segment -> must notify managers to update inode and imap; deltas used to ensure atomic update
- Manager distribution policy – First Writer adopted – collocates manager with client; empirically shown to reduce network traffic

# Cache Consistency

- Per-block
- Token-based
  - token owner has write permission
  - other caches will be written back and/or invalidated
- Same data structure used for cooperative caching

# Other features

- **Cleaning**
  - Distributed – each client responsible for own data;
  - Optimistic – segment info for writes while in write buffer delayed
- **Recovery**
  - Period checkpoints on logs, also imap file and segment file
  - Scan log until checkpoint determined, then scan forward to “replay” operations... clients responsible for replaying delta logs...
  - Managers, Servers selected based on consensus
  - More on recovery in next set of papers
- **Security**
  - Combine with ACLs
  - Encrypt and protect access to logs at clients
- **Implementation**
  - Under VFS layer (except as user process)

# Some performance trends

