CSE231 - Advanced Operating Systems

Serverless Network File Systems

Fabien Savy

Centralized systems are 👎

- scalability: limited by the hardware
- reliability: single point of failure

→ costly *hacks*

Motivation

- improving network capabilities (switches)
- expanding demands on file systems

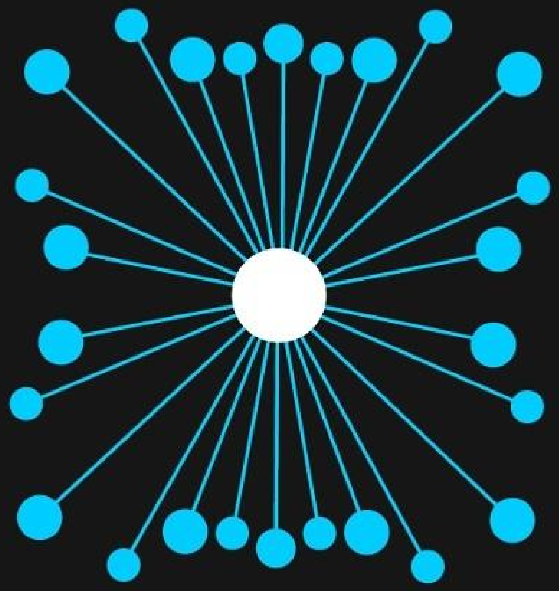
Contribution

Design for a

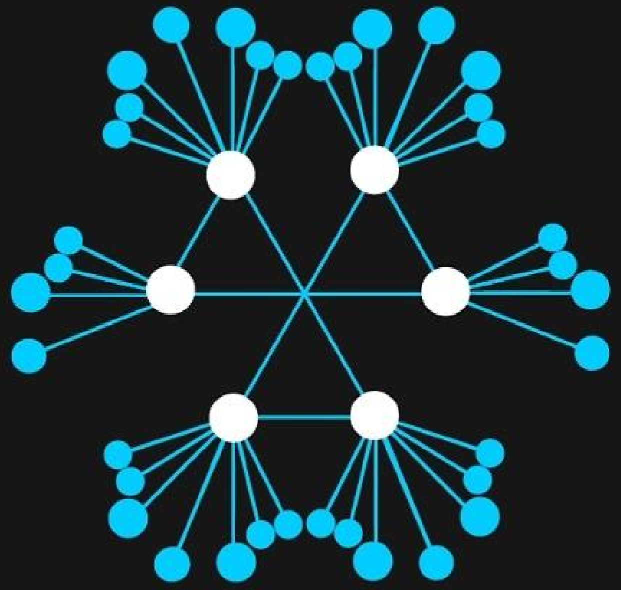
- decentralized
- fault-tolerant
- scalable

network file system

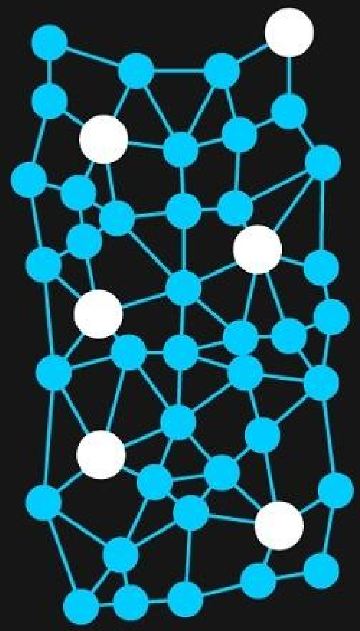
Centralized vs Decentralized vs Distributed Network: An Overview



Centralized Network
All the nodes are connected under a single authority



Decentralized Network
No single authority server controls the nodes, they all have individual entity



Distributed Network
Every node is independent and interconnected with each other

101 Blockchains
Created by 101blockchains.com

Design

> “Anything, anywhere”

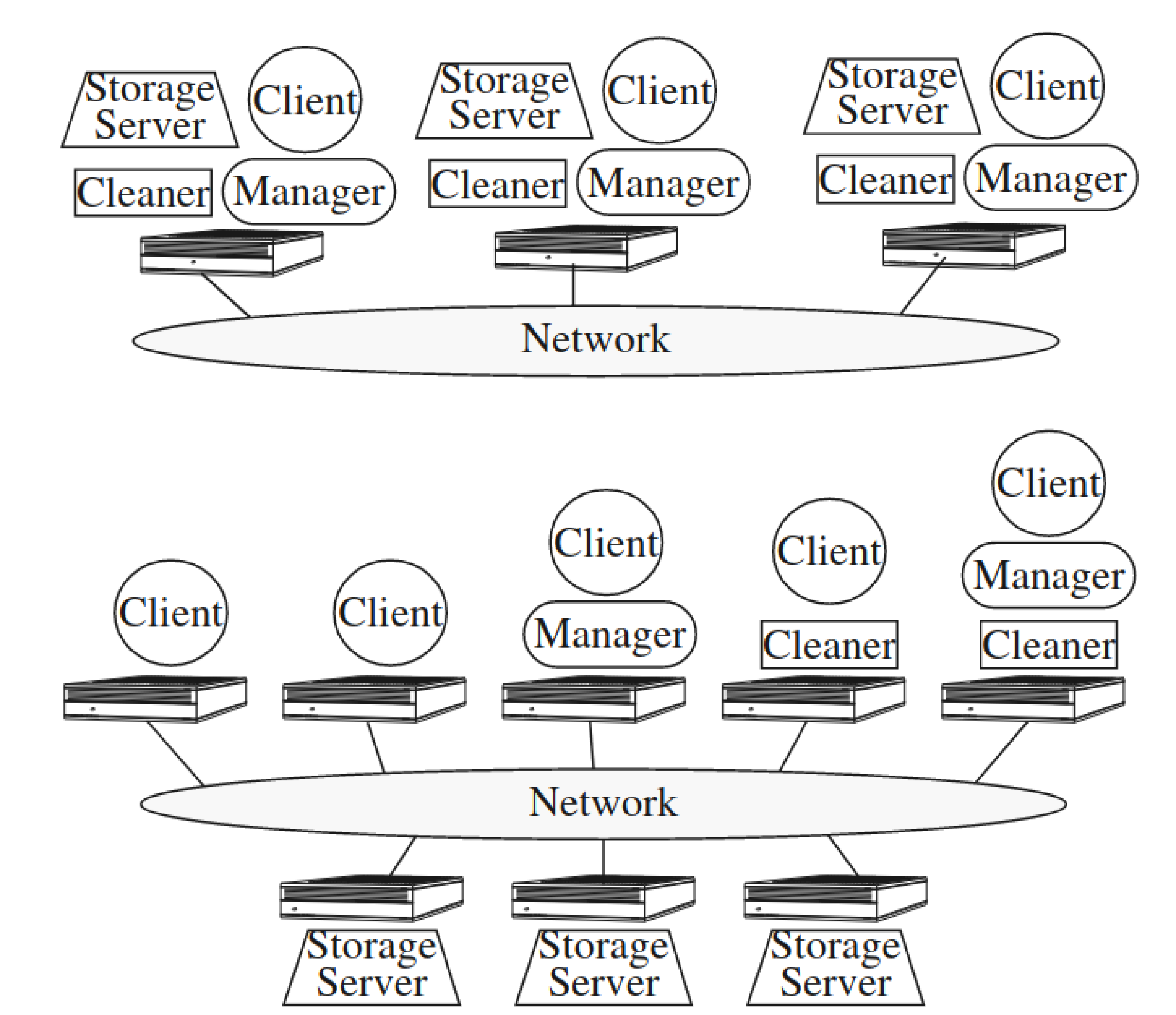
- journal based
- cache sharing
- metadata managers
- stripe groups

Tasks of a file system

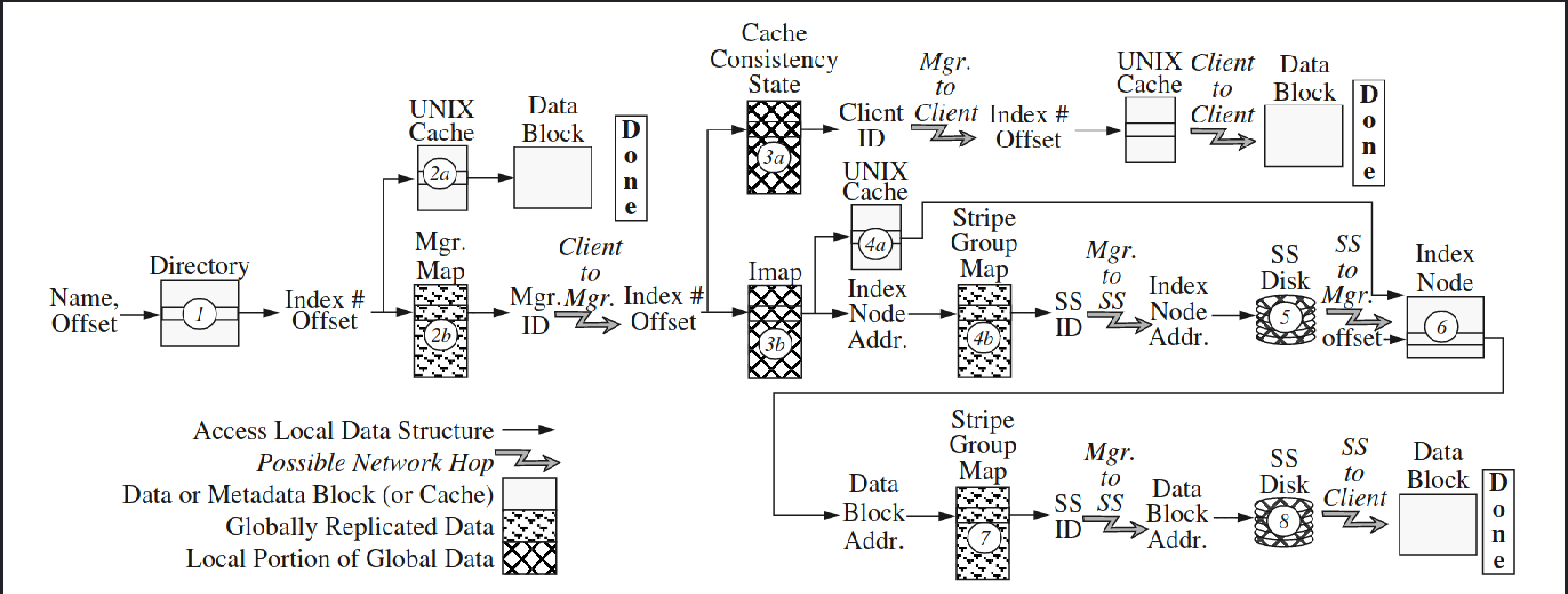
Task	Role
store data blocks	storage server *cleaner*
manage location metadata	manager
maintain a cache	client
manage cache consistency	manager

Roles

- Client
  - Simple client
  - Manager
  - Cleaner
- Storage Server
  - stripe groups
  - parity servers



Reading a file



Main data structures

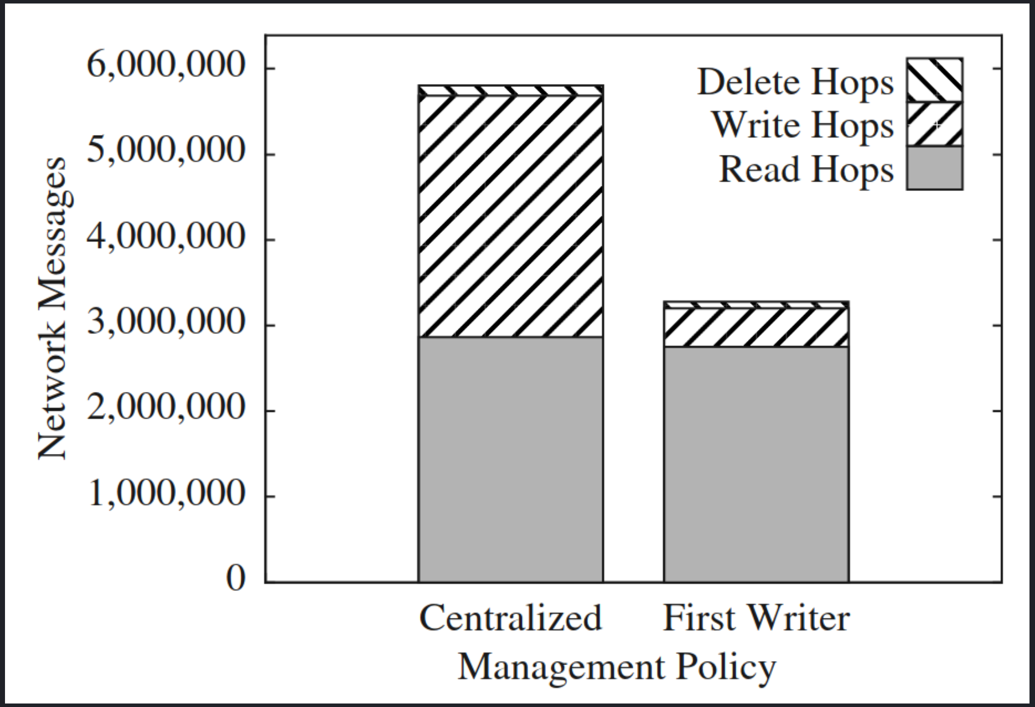
- client cache
- manager map
- cache consistency state
- imap (inode map)
- stripe group map

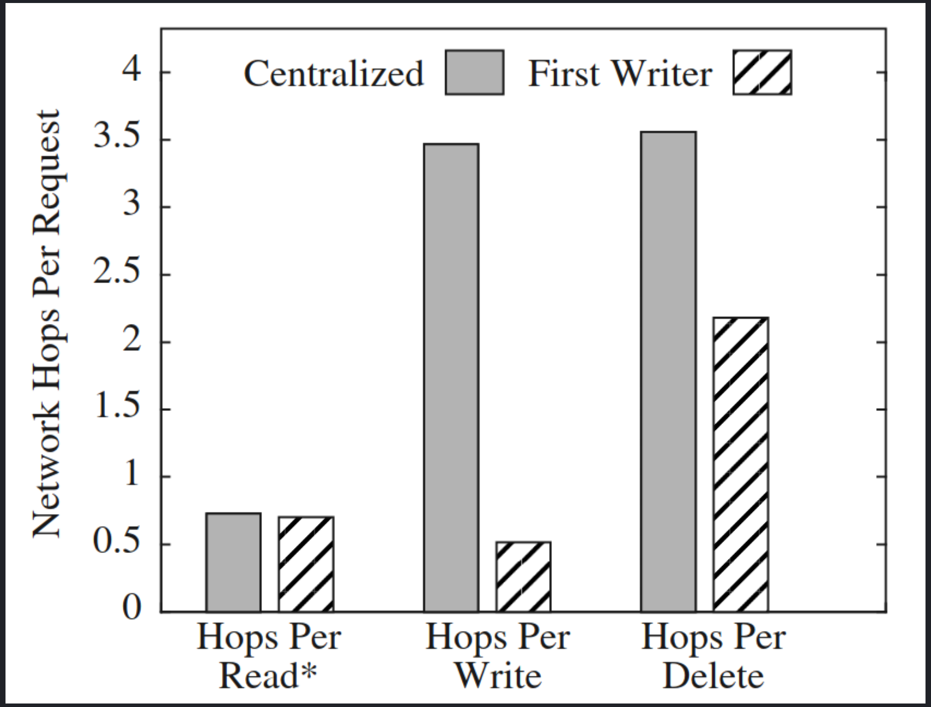
Writing a file

- client directly commits to a storage server
- notifies the associated manager
  - write authorization
  - updates index node
  - invalidates client caches

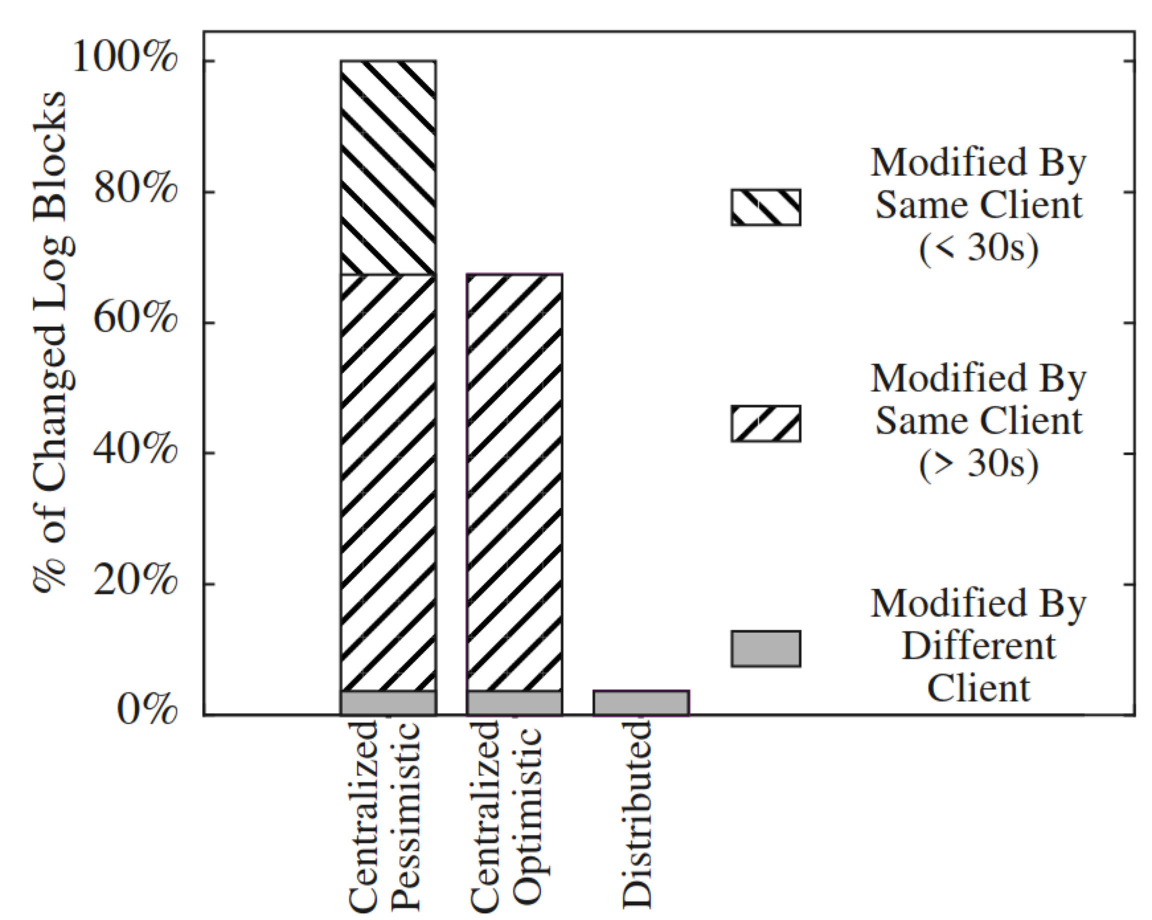
First Writer Policy

Co-locate a file management with its creator client





Cleaning



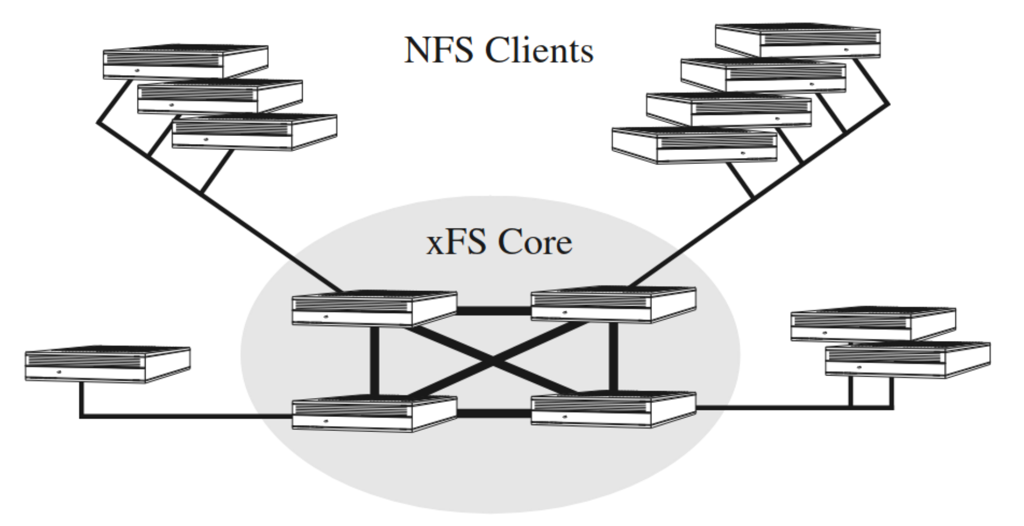
- log-based file systems requirement
- should be decentralized
- simulation to find the best strategy
- concurrent writes/cleaning

Recovery

- checkpoints + roll forward
- each role maintains a log of its actions
- distributed consensus algorithm
  - manager map
  - stripe group map
- reconfiguration is similar to a recovery

Security

- restricted environments
- trust one another
- untrusted support *via* traditional protocols (NFS)



Implementation

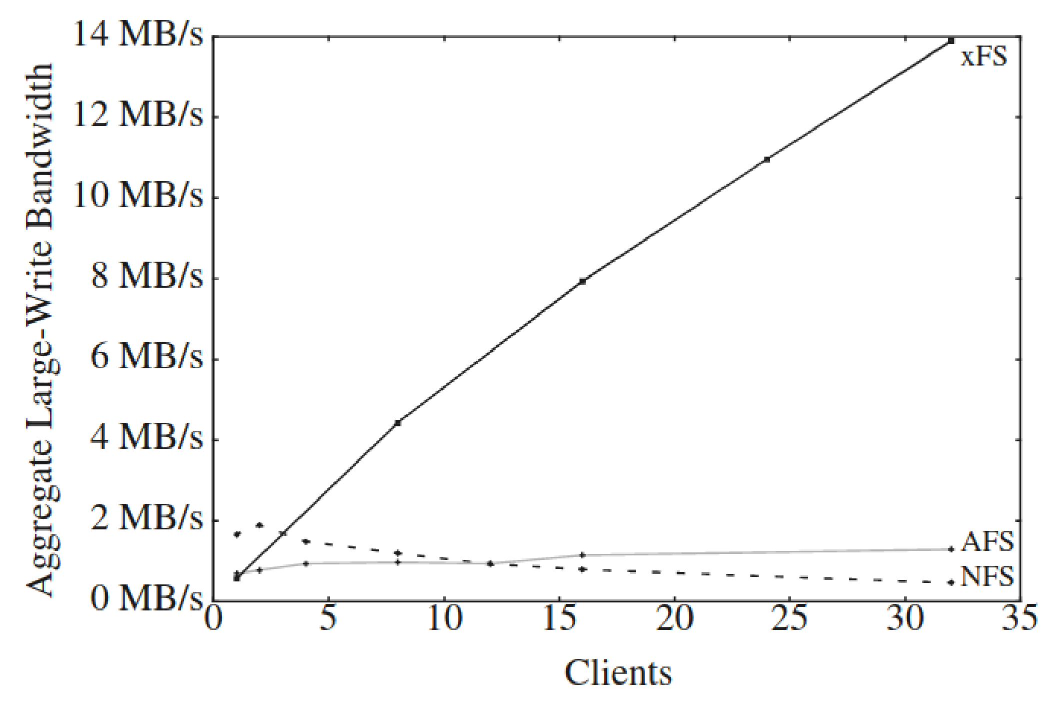
Sun SparcStation 10 - Wikimedia

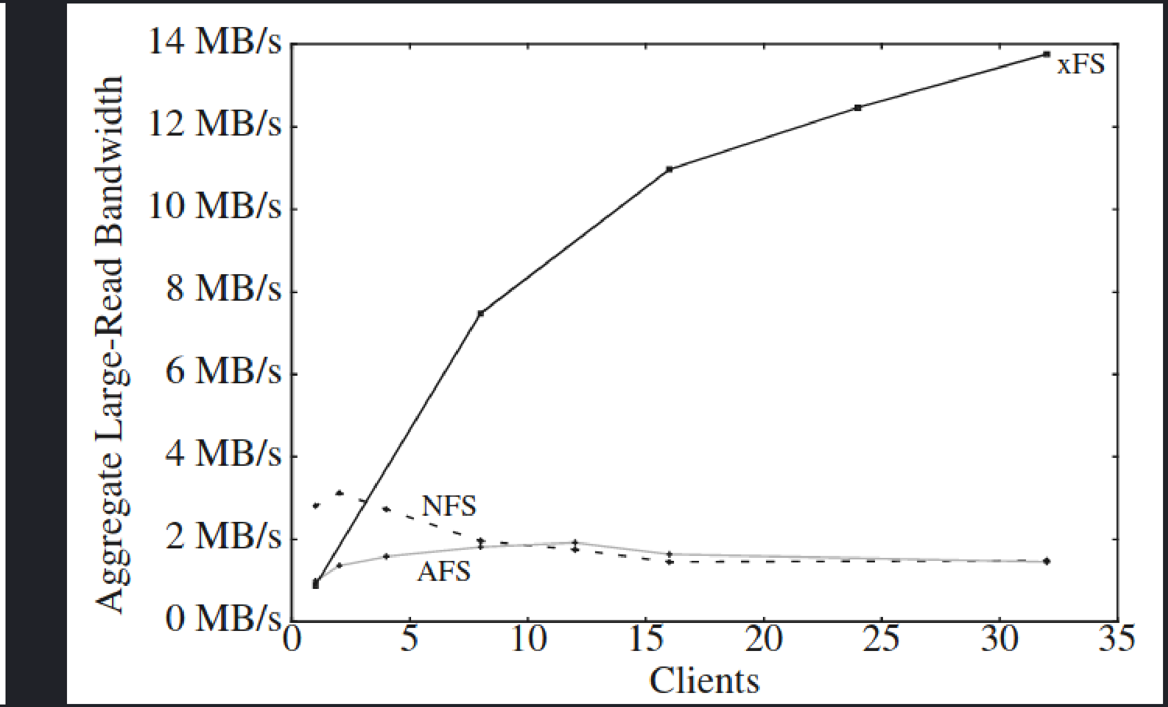
- a kernel module (v-nodes and cache)
- user-level daemons (client, manager, storage server)
- missing crash recovery and cleaning

Evaluation

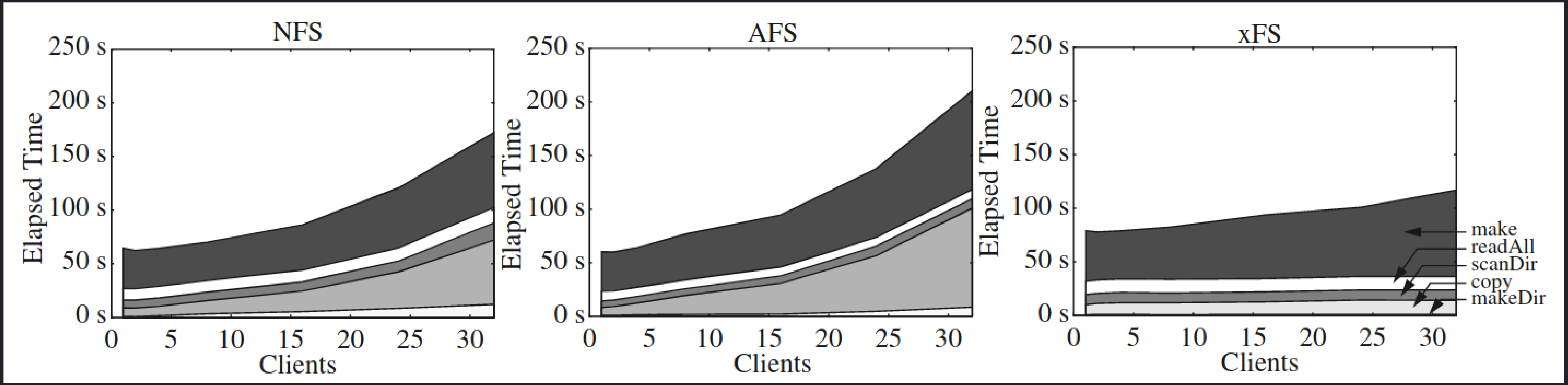
- 32 node cluster
- microbenchmarks against NFS & AFS
  - large files (read and write)
  - impact of stripe size
  - small files (write only)
  - Satyanarayanan’s Andrew benchmark
- setup is a bit dubious 🤔

Serverless Network File Systems





- poor individual performance
- great scalability 🎉



Andrew benchmark results

Missing experiments 🔍

- performance on demanding applications
  - simultaneous multimedia queries?
- impact of reconfiguration or failure
- stripe group size decision

📢 Discussion

- The security model is based on absolute trust. What if a machine gets compromised?
- Is it possible to use this system with mobile nodes (maybe with caching ideas from Coda)?
- Generally, client machines are less powerful than servers. What would be a realistic use case?

Thank you for your attention! 😊