An Analysis of iSCSI for Use in Distributed File System Design

CS740 Project Presentation

Michael Brim & George Kola

May 6, 2003

Motivation

- iSCSI enables network-attached storage by sending SCSI commands over standard IP networks
- Research Questions:
  - Can iSCSI be used as a basis for an application-specific distributed file system?
  - What are the performance implications of using iSCSI in this manner?

Project Overview

- Environment Configuration
- iSCSI Analysis
  - Uses and Limitations
  - Modus operandi
- Performance Profiling
  - iSCSI – raw read/write speeds
  - Versus local disk & remote NFS
- Implications of using iSCSI
- iSCSI-based Distributed File System Design & Implementation

Related Work

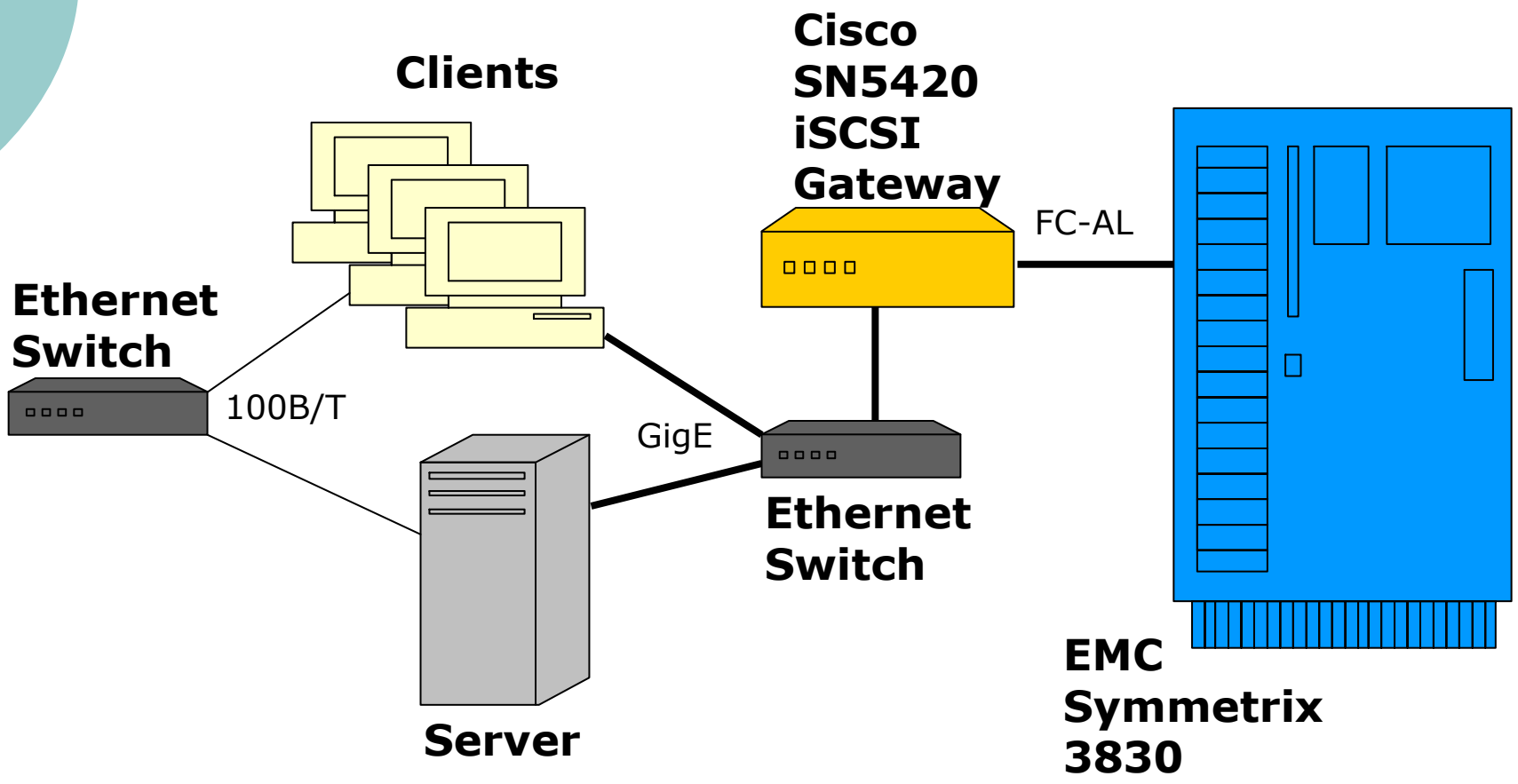
- Network-Attached Storage (NAS)
  - Storage devices directly connected to standard network such as Ethernet
  - Devices provide file-system functionality to clients who access them directly over the network
  - E.g., Network-Attached Secure Disks
- Storage-Area Network (SAN)
  - Block-level interface to networked storage using dedicated Fibre-Channel network

Related Work

- Data Reservoir
  - Multi-tier file transfer for scientific applications over wide-area gigabit networks using iSCSI
- Distributed File Systems
  - Assume server-attached storage
  - iSCSI could be used with a compatible networked storage device, but really just adds more latency

Experimental Design

- Storage Architecture



Profiling iSCSI

- What percentage of CPU does using iSCSI consume ?
  - vs. local storage
  - vs. NFS
- What effect does block size have ?
- Which parts of the kernel are stressed ?

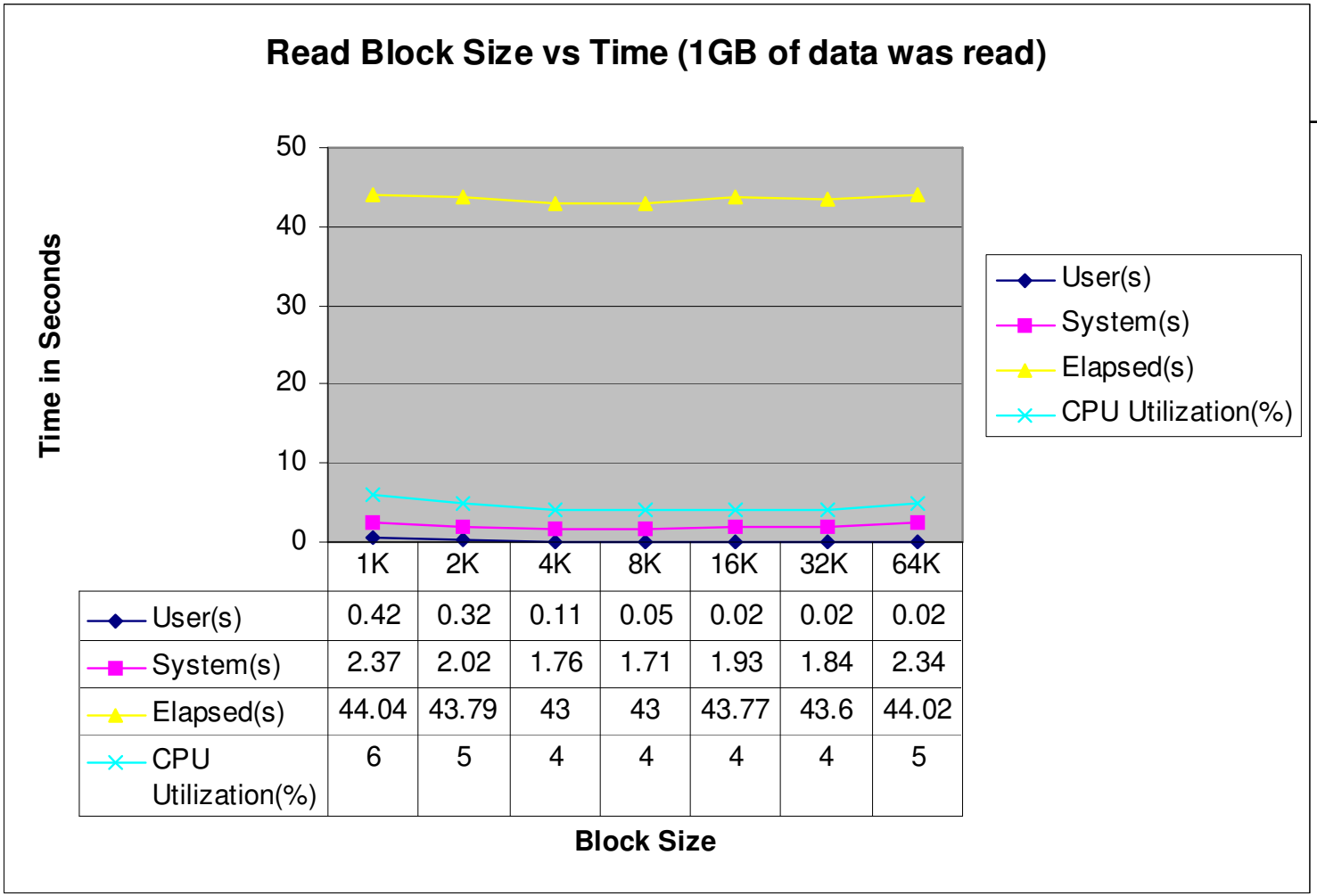
Using iSCSI - DFS Design Issues

- Effects on multiple access
- What's the incremental cost of additional clients ?
- How does it compare to NFS ?

Experiment Methodology

- Used oprofile to collect stats
  - Detailed kernel and user-level stats were collected
- Multiple samples were collected and the samples were collected for long durations (4 hours+)

Effect of Block Size on Performance



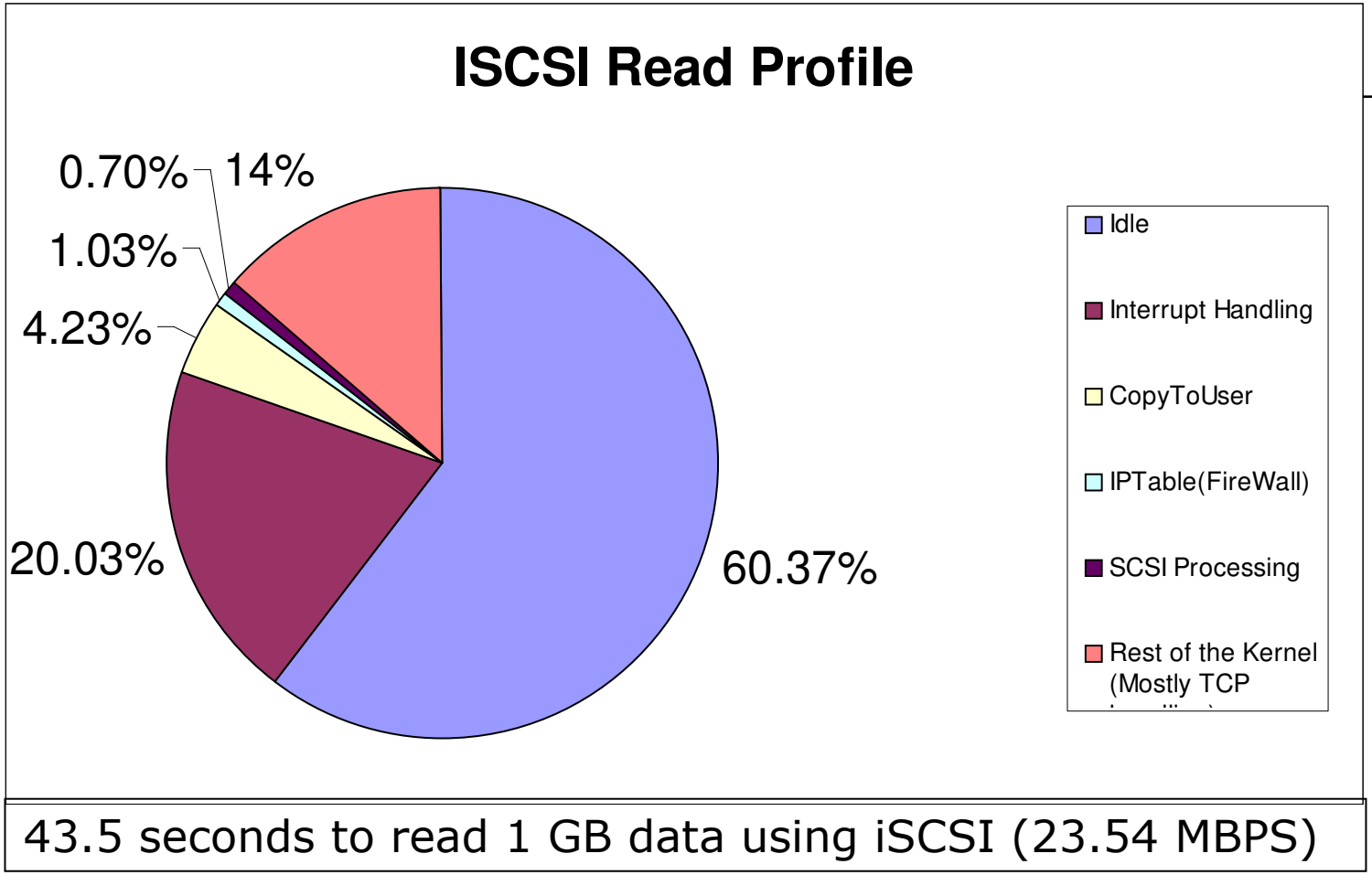
	1K	2K	4K	8K	16K	32K	64K
User(s)	0.42	0.32	0.11	0.05	0.02	0.02	0.02
System(s)	2.37	2.02	1.76	1.71	1.93	1.84	2.34
Elapsed(s)	44.04	43.79	43	43	43.77	43.6	44.02
CPU Utilization(%)	6	5	4	4	4	4	5

Block Size of 8K is optimal

For Writes, 8K is optimal

CPU % utilization just gives the trend. Linux kernel does not account the time spent in kernel correctly to the process

ISCSI Read Profile

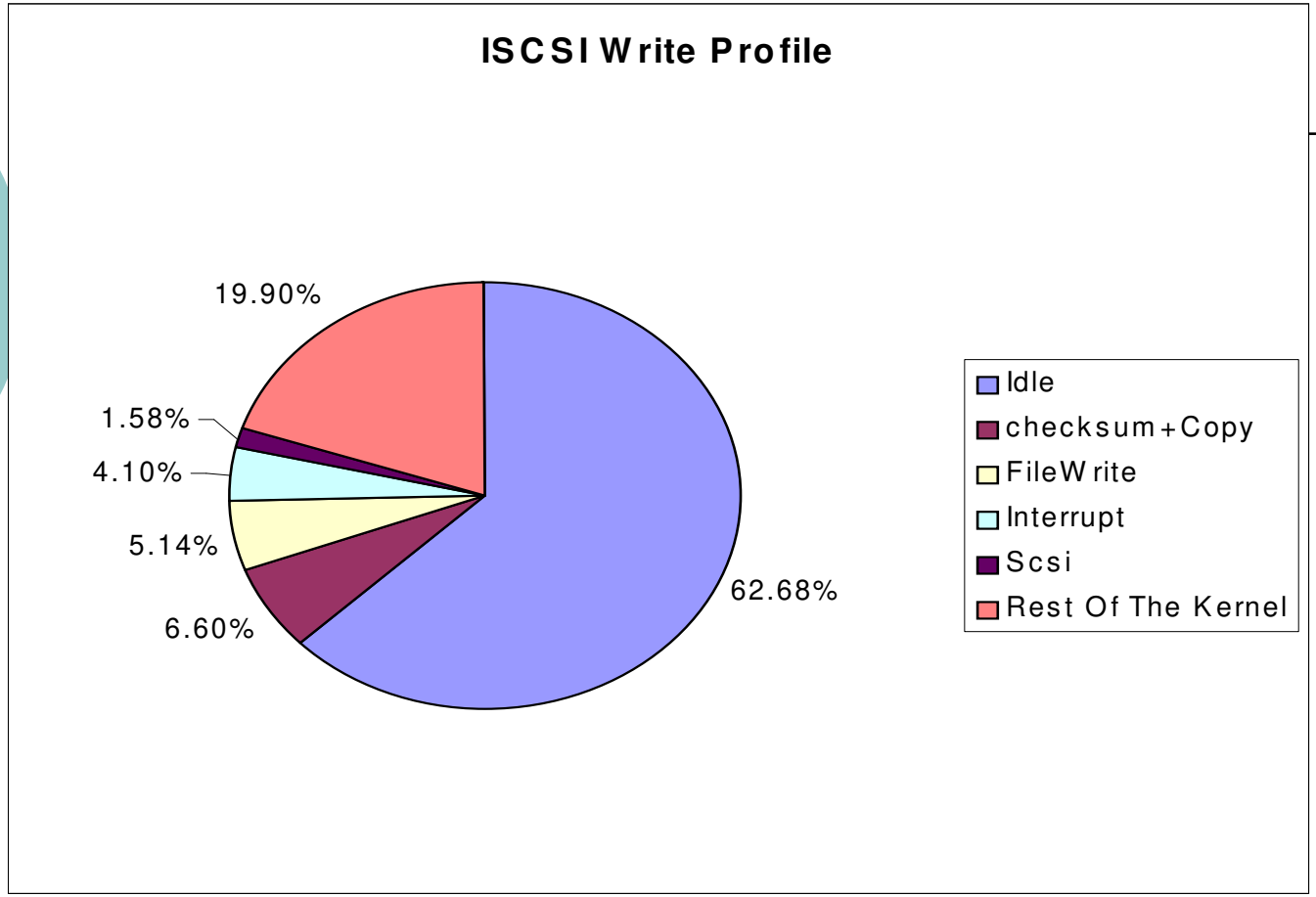


43.5 seconds to read 1 GB data using iSCSI (23.54 MBPS)

Local Disk Read 43.6 seconds to read 1GB data (23.48 MBPS)

Idle : 94.7%, IDE Interrupt Handling < 3.5 %, File Read 1.4 %

ISCSI Write Profile

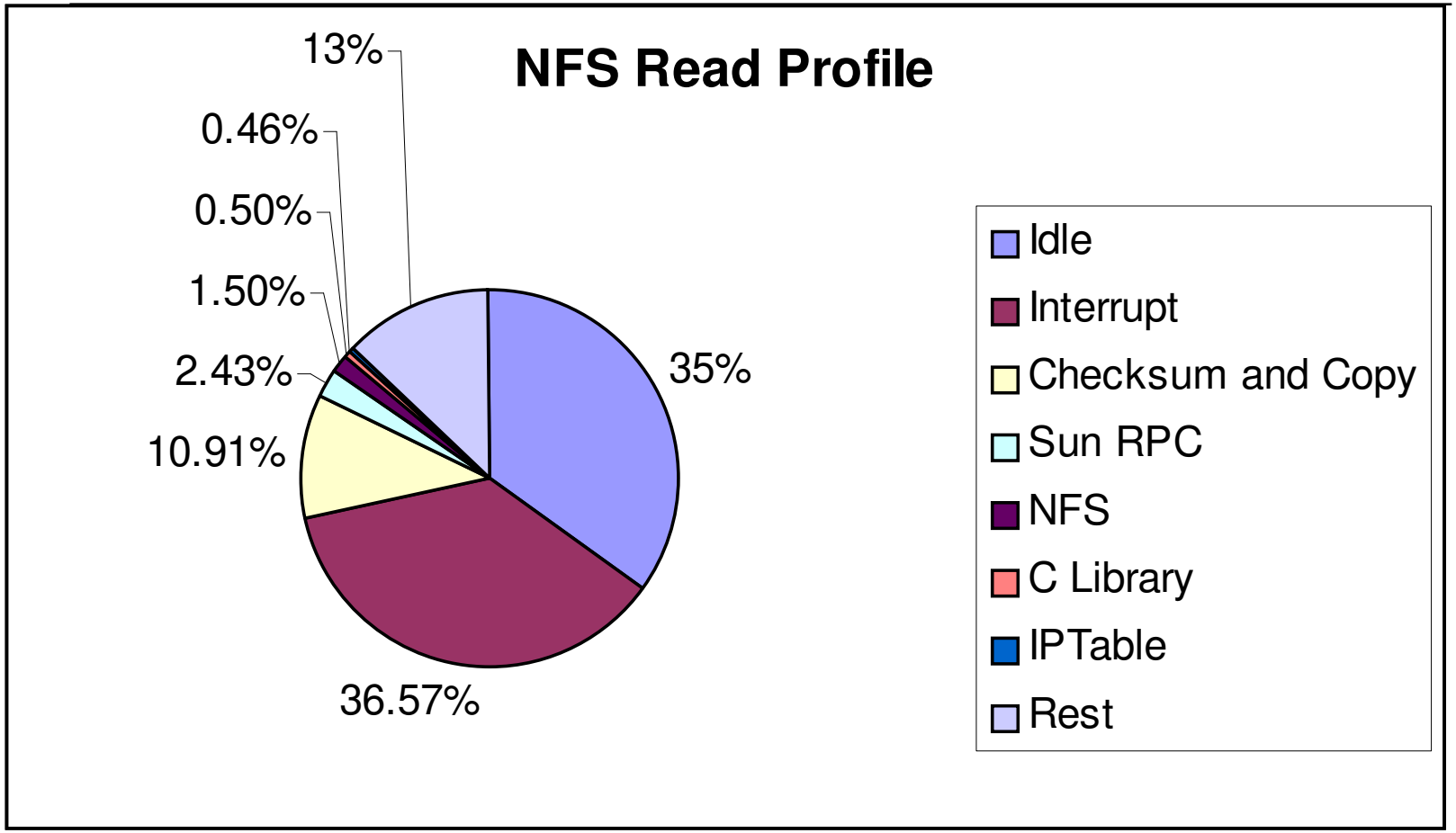


25.8 seconds to write 1GB data using iSCSI (39.6 MBPS)

Jumbo Packet Option in gigE increased performance by 1.2 MBPS

Local Disk Write 54.6 seconds to write 1 GB data (18.75 MBPS)
Idle 93.5 %, ide dma < 3.5, file_write 2.8%

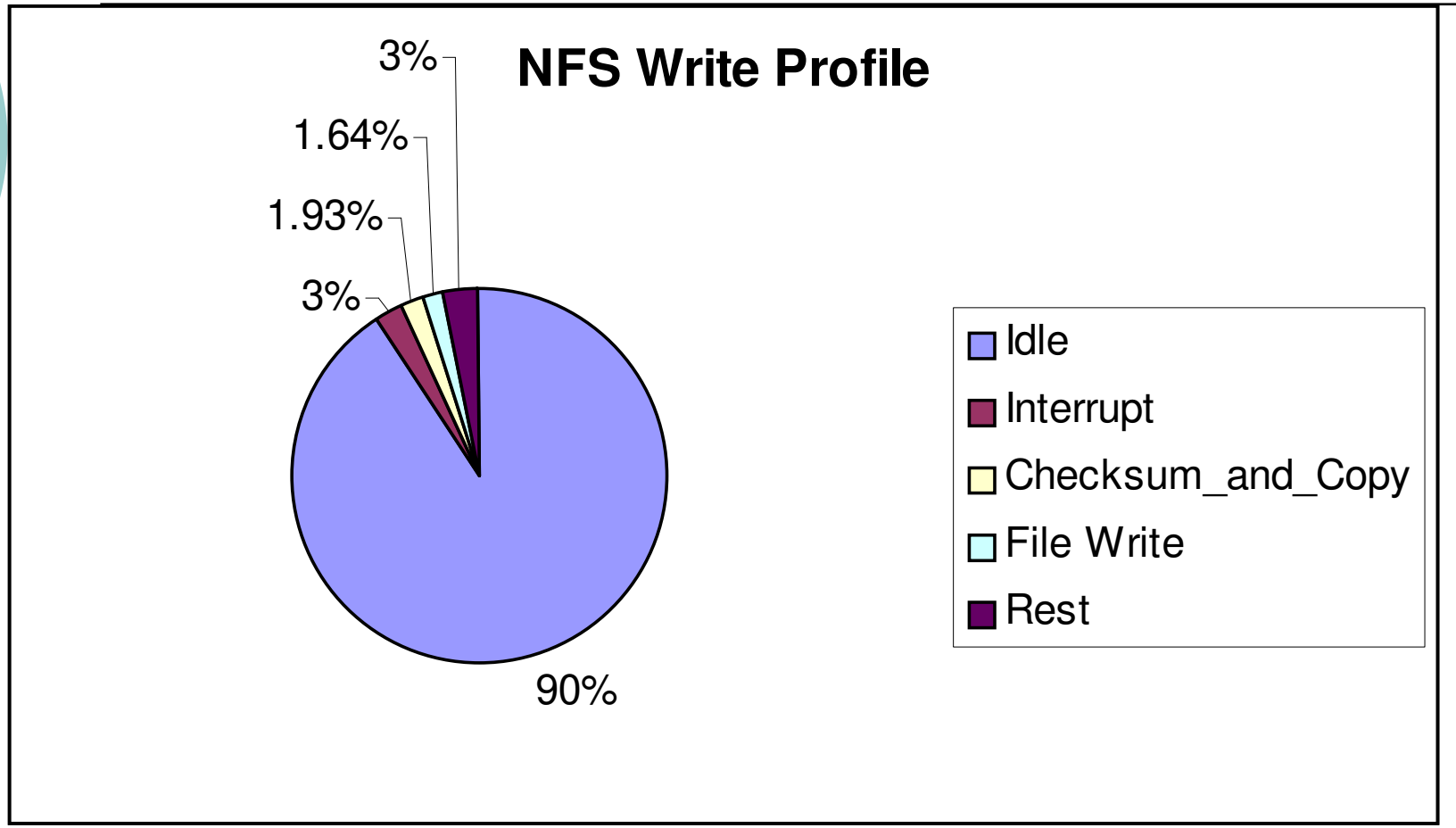
NFS Read Profile



23.23 seconds to read a 1 GB file (44 MBPS)

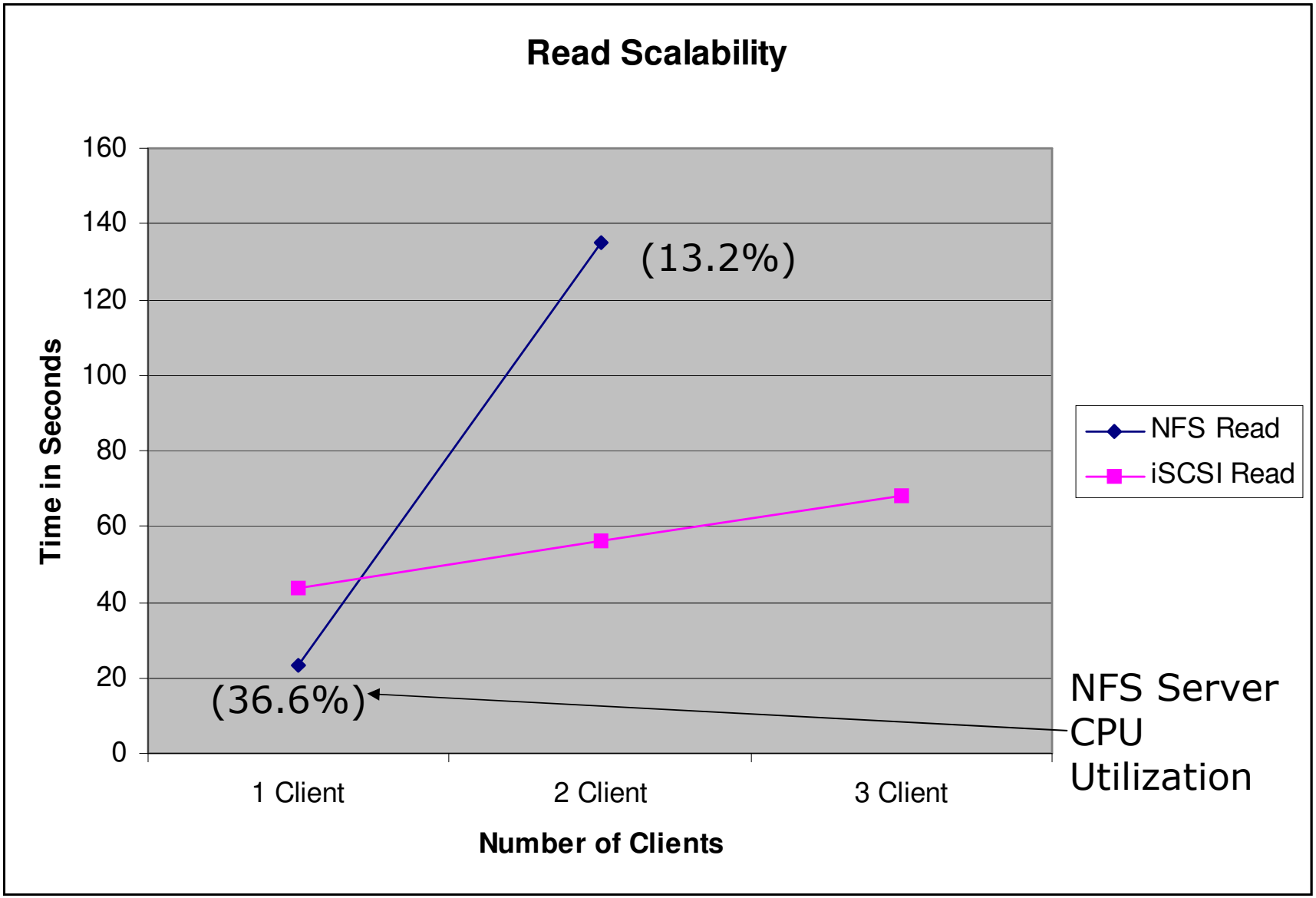
Server-side caching helps

NFS Write Profile

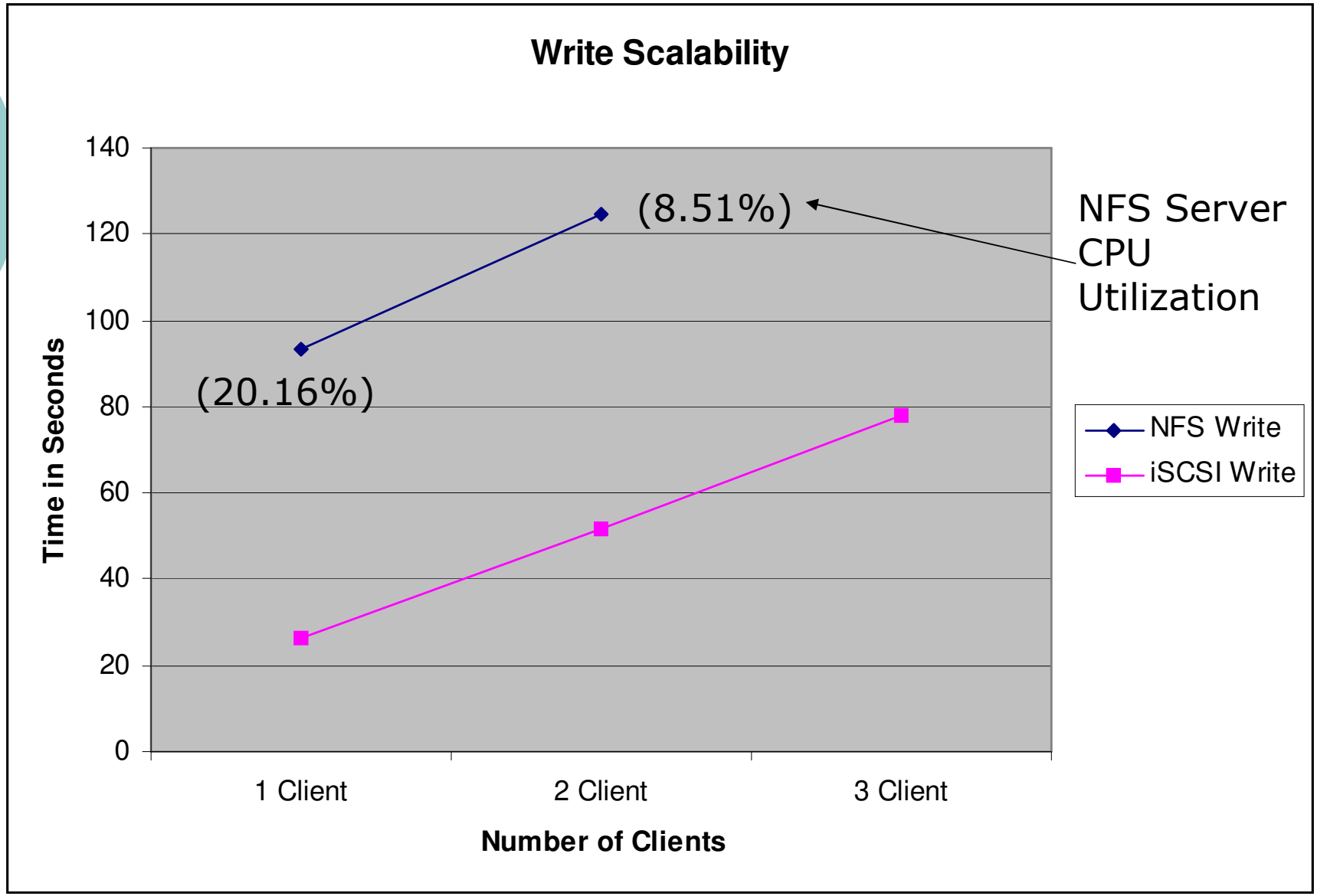


1 min 44 sec to write a 1 GB File (9.85 MBPS)

Read Scalability iSCSI vs NFS



Write Scalability iSCSI vs NFS



Implications

- A distributed file system on top of iSCSI should have:
  - a strong consistency model such as in AFS
  - the ability to cache aggressively to mitigate the overhead of going over gigE.
- The security model of the distributed file system built on top on iSCSI would be similar to NFS with root access

Our DFS Design

- Simple request/reply architecture using TCP
- Security Model: Assume all clients trusted & well-behaved
- Consistency semantics: many readers, one writer with callbacks

Protocol Overview

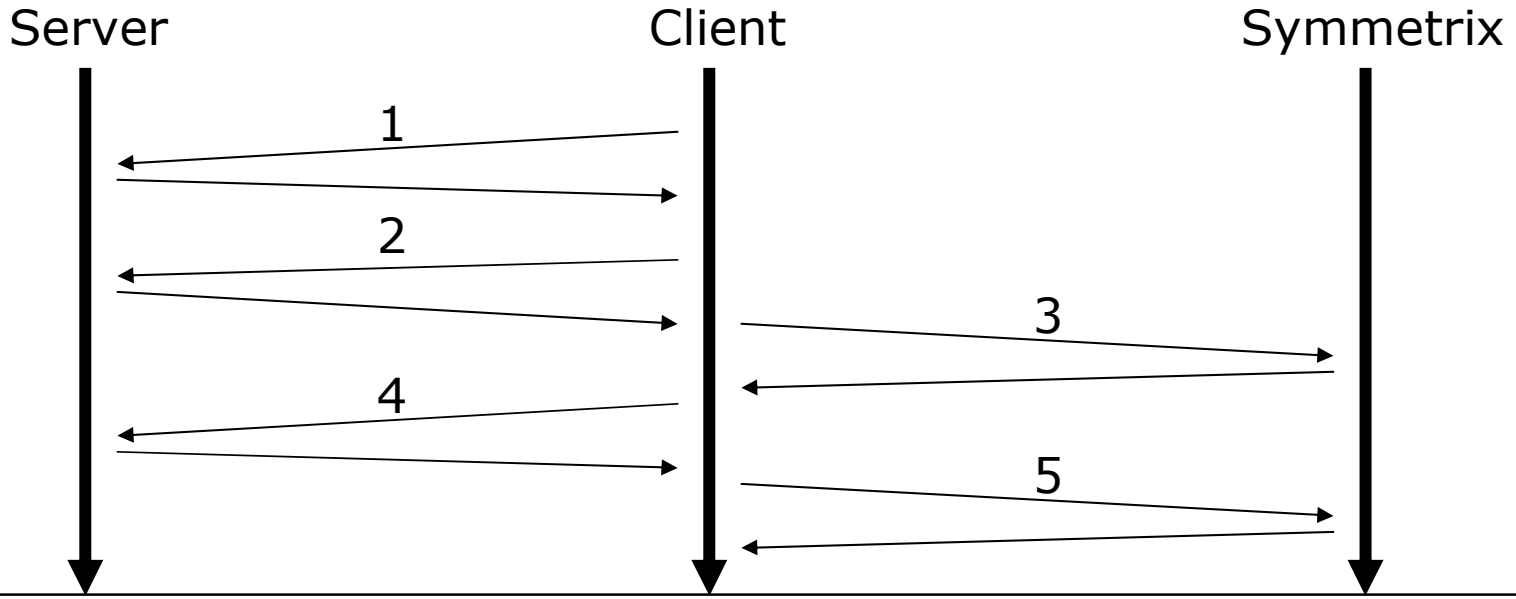
1a. Client: mount /dev/sdc /mnt/iscsi
  b. Server: returns superblock info for /dev/sdc

2a. Client: read_inode(inode 0) – root directory
  b. Server: Return inode 0 info

3a. Client: read file data
  b. Symm: returns data

4a. Client: read_inode(inode #) - file in already discovered directory
  b. Server: returns inode info for file

5a. Client: read file data
  b. Symm: returns data



Future Work

- Finalize DFS
  - Add caching
  - Add consistency semantics
- Benchmark DFS
  - Performance vs. NFS
  - Scalability studies

Conclusions

- iSCSI can provide similar performance to traditional DFS
  - Write is significantly better than read
  - Caching would improve reads
  - Potentially better server scalability
- iSCSI has a high performance cost which is associated with gigE
  - not likely to replace FC for SAN servers or local storage in traditional DFS
  - only tolerable for large files