The Design and Performance of a Shared Disk File System for Silicon Graphics' Irix

Grant Erickson
Steve Soltis
Ken Preslan
Matthew O'Keefe
Tom Ruwart

Department of Electrical and Computer Engineering
and
Laboratory for Computational Science and Engineering

UNIVERSITY OF MINNESOTA
Minneapolis, MN

`gfs@lcse.umn.edu`
`http://www.lcse.umn.edu/GFS/`

Outline

- Problem Definition
- Enabling Technologies
- Global File System
  - Architecture
  - History
- Performance Evaluation
- Future Work
- Conclusions

Problem Definition

- Widespread usage of computer networks
  - Enable distributed work environments
  - Promote sharing and exchange of information
- Sun's *Network File System* (NFS) is the de facto file sharing standard.
  - Transparent: looks and feels like a local file system
  - Portable: runs on a wide variety of client and servers
  - Robust: simple crash recovery
- The strengths of NFS also lend to its weaknesses
  - Large files restrict work to local storage
  - Raises barriers to sharing and exchange of large data sets

NFS Limitations

- ❖ Performance is dictated by latency and throughput of the network
- ❖ 10 or 100 Mbps Ethernet can deliver <u>at best</u> only 1.2 to 11.9 MBps of bandwidth via NFS
- ❖ NFS Server
  - ❑ Requires expensive, dedicated computer or network appliance
  - ❑ Single point of failure limits reliability and availability
  - ❑ Scales poorly in high-demand environments
- ❖ NFS performance limitations are further aggravated by the trend toward large file sizes

Explosive Data Growth

- Both documents and applications are becoming more media-rich, driving up file sizes
- Continued growth in capacity of memories and disks promotes further file growth
- Example environment: digital production houses
  - *Sneaker net* is preferred data transport media
  - *Vista Vision* film format: 4096 lines of 6144 pixels per frame
  - *Cineon Lighting* scanner captures at 14-bits per RGB component
  - At 24 frames/second—3.0 GB for 1 second of film
  - 42 minutes to transfer using 10 Mbps Ethernet.

Enabling Technologies

- Fibre Channel
  - High bandwidth, low latency network and channel interface
  - Highly scaleable, very flexible topologies
  - Becoming high-volume, hence lower-cost
  - Support from a wide-variety of adapter, computer, networking, and storage vendors
  - Supports the connection of storage devices to the network
- Network-attached Storage (NAS)
  - Have your disks and share them too
  - Allows direct data transfer between disks and clients
- Together, Fibre Channel and NAS enable storage area networks (SANs).

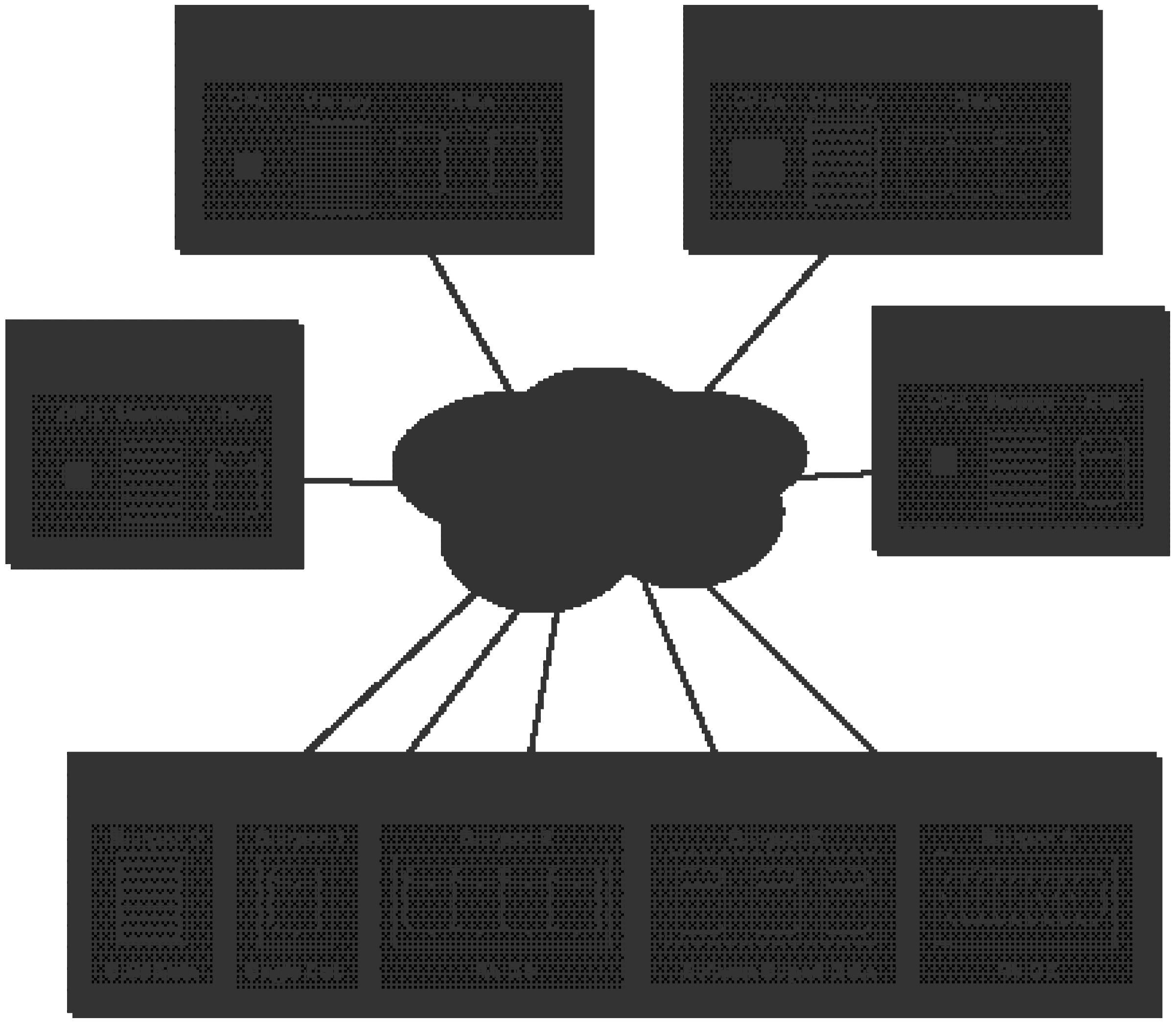
Global File System

- Novel block-addressable, serverless, hardware-based solution to distributed file systems
- Leverages high-bandwidth and high-availability of SANs to facilitate applications with large storage requirements.
- Symmetric architecture
  - Modeled like a shared memory multiprocessor
  - Clients are independent and have equal access to storage
- Hardware-based mutual exclusion locking mechanism used to ensure data consistency
- Layered on top of a *Network Storage Pool*

Network Storage Pool

- Coalesces a heterogeneous collection of shared storage devices into a single, logical contiguous *pool* of storage space
  - Allows for striping across multiple devices
  - Similar to Silicon Graphics' *xlv* logical volume manager
- Devices may be divided into *subpools* according to device performance characteristics
- Provides an interface for a *pool* of device locks
  - Hides actually locking implementation from file system layer
  - Locks may be located on one or more storage devices or on a dedicated lock device

A Distributed GFS Environment



Device Locks

- Device Locks
  - Facilitate atomic read-modify-write operations
  - Similar in operation to memory locks with *test-and-set* and *clear* operations.
  - Many locks (□ 1024) per device leads to greater parallelism
  - Provide mechanism for client initiated error-recovery
- Lock structure
  - State bit, activity bit, multi-bit counter
    - State bit indicates whether lock is held or available
    - Activity bit used to initiate client-based lock recovery
    - Counter is only incremented on modify operations
  - Increase/decrease in counter resolution may be exchanged for decrease/increase in lock quantity

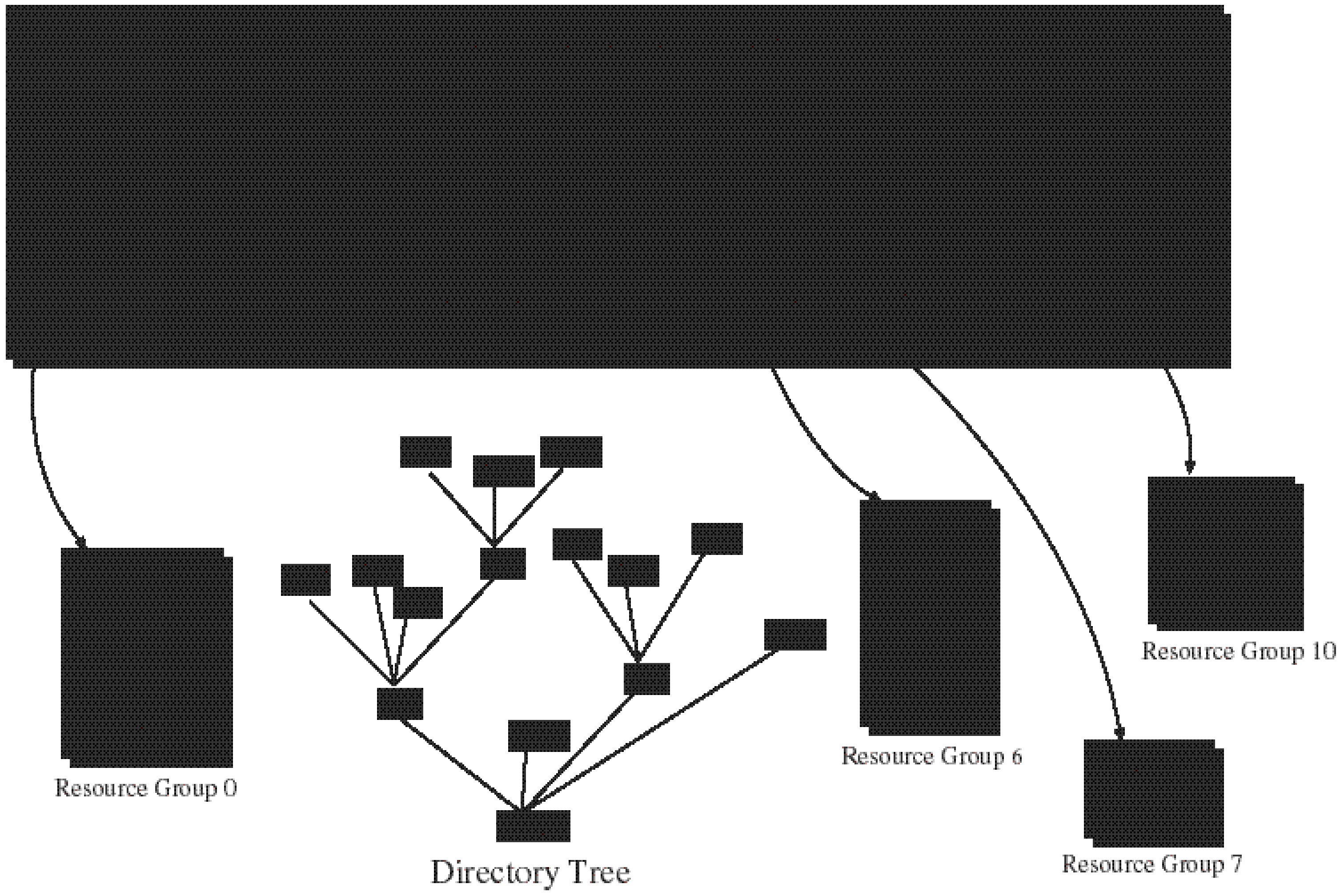
File System Consistency

- File system implements a many-to-one mapping of files to locks
- GFS maintains perfect file consistency
  - Utilizes write-through caching
  - All client reads obtain the most recent data
  - Limits damage during client failure
  - Simplifies file system recovery
- State of the lock counter is used for limited client-side caching of file dinodes

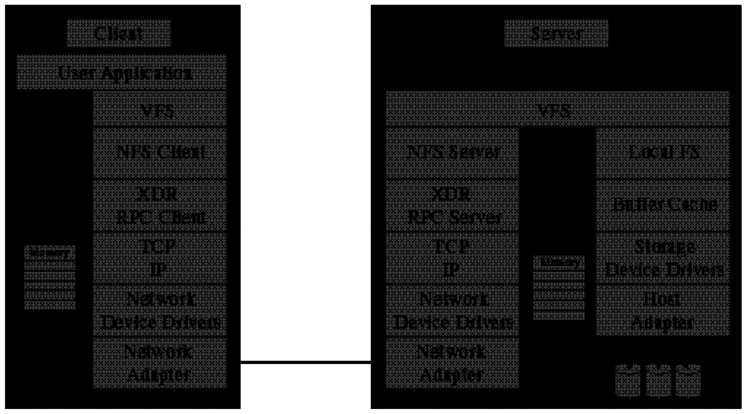
GFS Organization and Architecture

- Super block
  - Maintains mount information and static file system attributes
- Resource Groups
  - Partitions and distributes file system resources for parallel accesses
  - Allocated per subpool in the network storage pool
  - Contains bitmaps used for block allocation
  - Similar to *Allocation Groups* in Silicon Graphics' *XFS*
- Dynamic Block Allocation
  - Available file system blocks may be freely allocated to directory or file dinodes, pointer blocks, or data blocks
  - Inode and dinode numbers based on storage pool address eliminating lookup indirection

Mapping Files to Resource Groups and Subpools

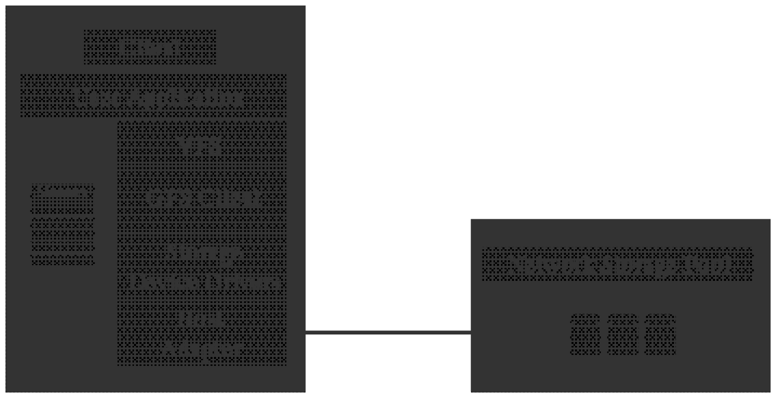


Comparison of GFS and NFS Control and Data Paths



NFS

GFS

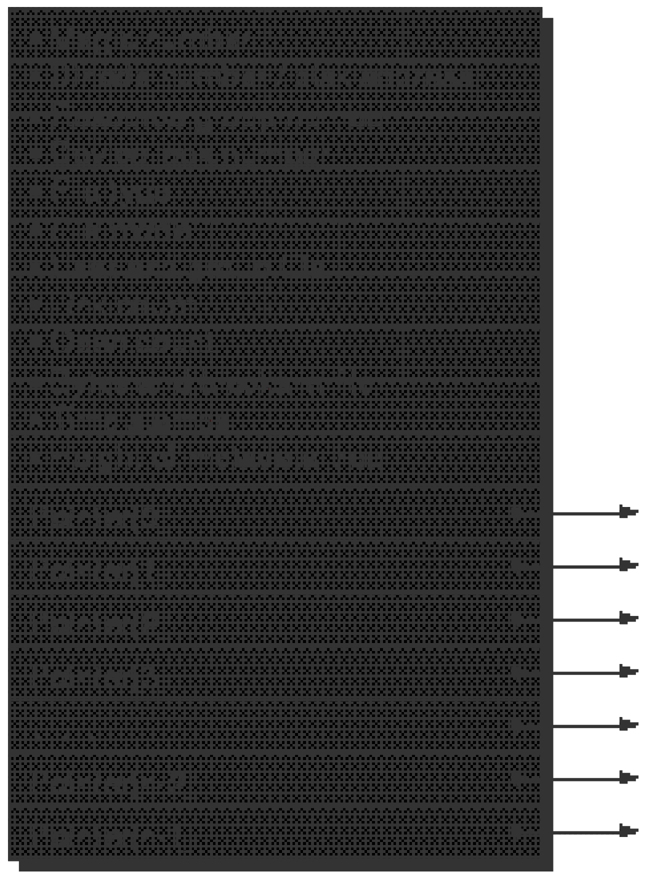


Silicon Graphics File System Interface

- Irix file system interface based on the *Virtual File System* and *Virtual Node* (*vfs/vnode*) interface
  - Developed concurrently with NFS for the Sun Microsystems *Solaris* operating system
  - Extended and formalized by the UNIX System V Release 4 (SVR4) specification
  - Although interface is standardized, implementations vary widely
- Irix completely implements SVR4 interface specification
  - Implementation is both proprietary and undocumented
  - Significant impediment to third-party file system development
- Proprietary implementation motivates ports to open platforms
- Ports to open platforms eased by *vfs*/*vnode* interface

Dinode Stuffing

- ❖ Improve small file performance
- ❖ Directory and file dinodes occupy an entire file system block
  - ❑ As block size increases header information stays constant
  - ❑ Block utilization decreases decreases leading to internal fragmentation
- ❖ Place user data in the unused dinode space
  - ❑ Reduce internal fragmentation
  - ❑ Eliminate pointer indirection
  - ❑ Eliminate and additional read operation



Performance Evaluation

- Bandwidth Characterization
- Scaling Study

Historical Perspective

- Early GFS prototype first presented at 1996 NASA/IEEE Mass Storage Systems and Technologies conference
- Three node Silicon Graphics *Indy* system
  - Modified parallel SCSI interconnect
  - Single shared *Seagate Barracuda 2LP* disk
  - SCSI *reserve and release* locking
- Today GFS is a fully-functional distributed file system architecture
  - Leverages the flexibility of Fibre Channel SANs
  - Support for any SCSI storage device
  - Low latency, fine-granularity device locks

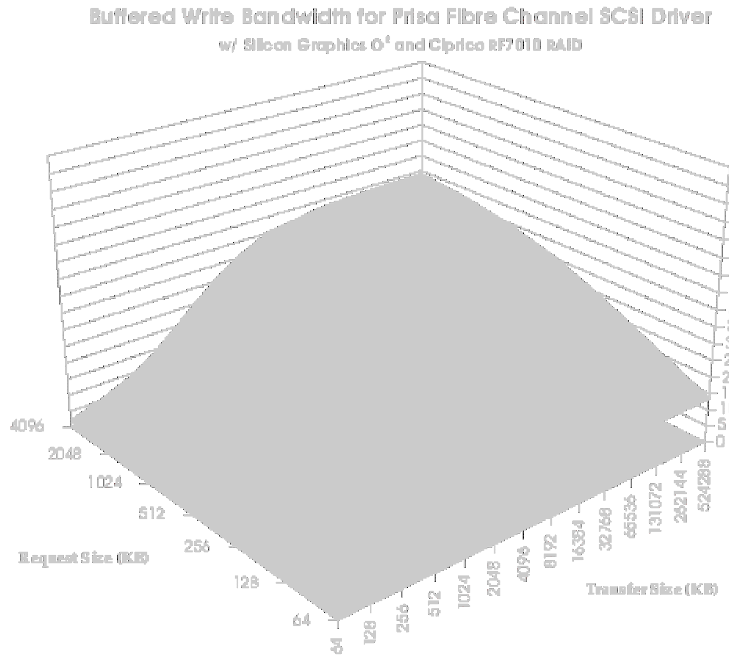
First Generation Implementation

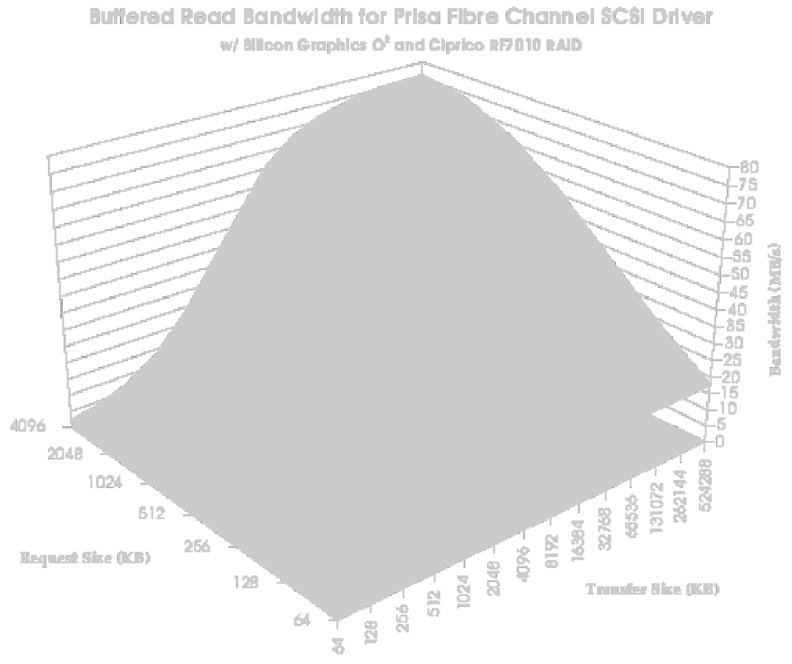
- Initial implementation of GFS interacted with the Irix kernel in a very limited fashion
  - Support for reading and writing data files
  - Limited directory support
  - Little or no error checking and recovery
- Features and functionality expected of a UNIX file system and now implemented in GFS today
  - Symbolic and hard file links
  - Access permissions
  - 255 character file names
  - Execution of binaries and memory mapping of files
  - Correct and robust operation in the face of both system and user errors

Bandwidth Characterization

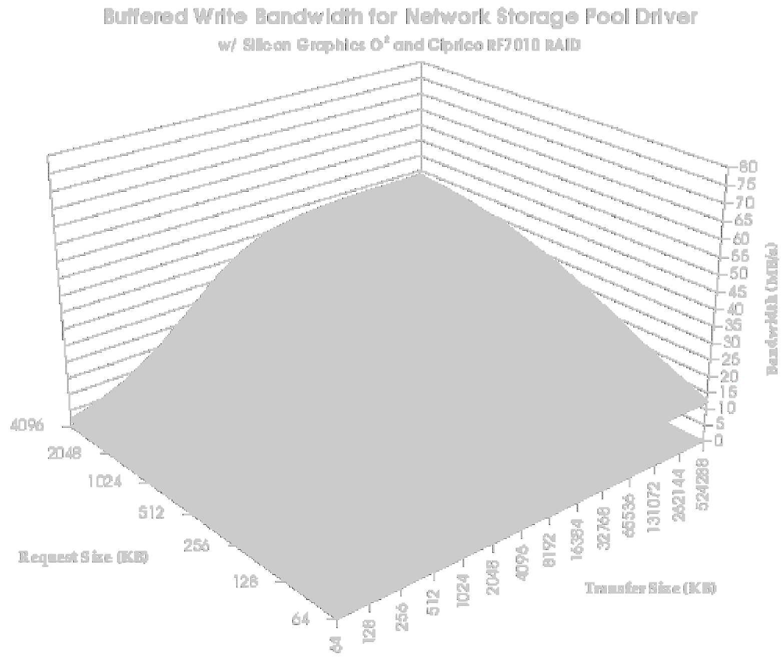
- Two parameter tests
  - Request size varied exponentially from 64 KB to 4 MB
  - Transfer, or file, size varied exponentially from 64 KB to 512 MB
- Test configuration
  - Single *Silicon Graphics O2* desktop workstation
  - *Prisa NetFX PCI-32* Fibre Channel host bus adapter
  - Single *Ciprico Rimfire* 7010 Fibre Channel RAID-3
  - *Brocade Silkworm* 16-port Fibre Channel switch
- Characterize the bandwidth for each subsystem
- Quantify the amount of overhead incurred by each subsystem by examining bandwidth losses

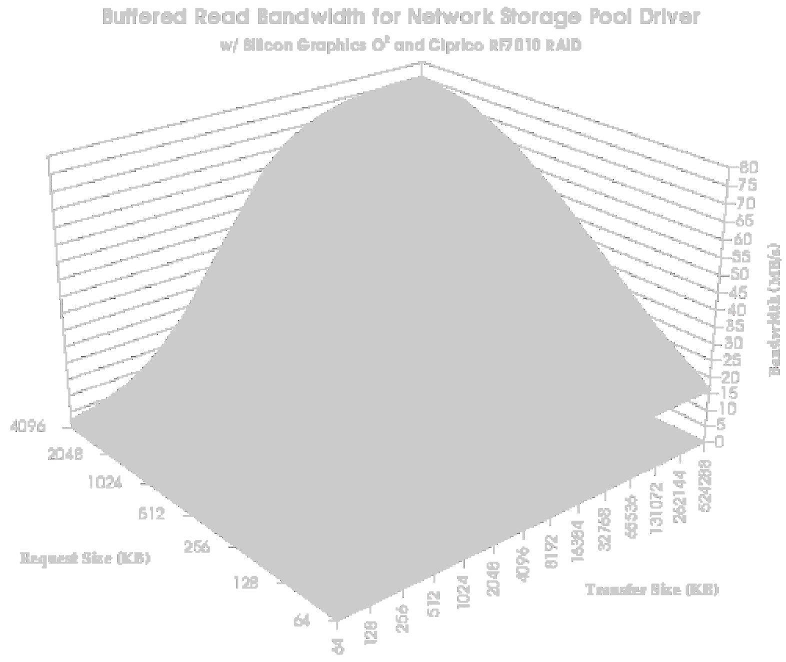
Host Adapter Bandwidth



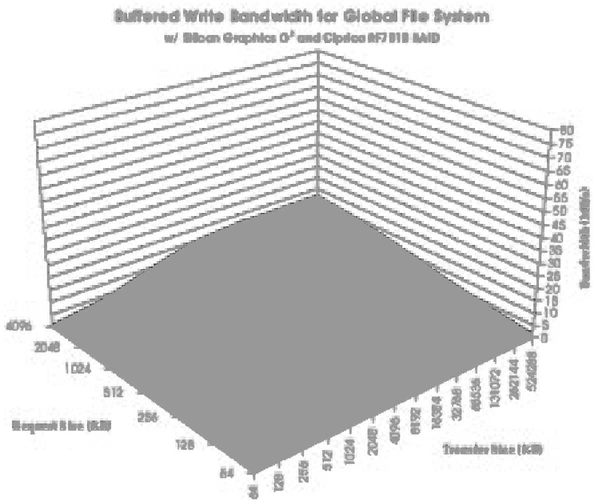


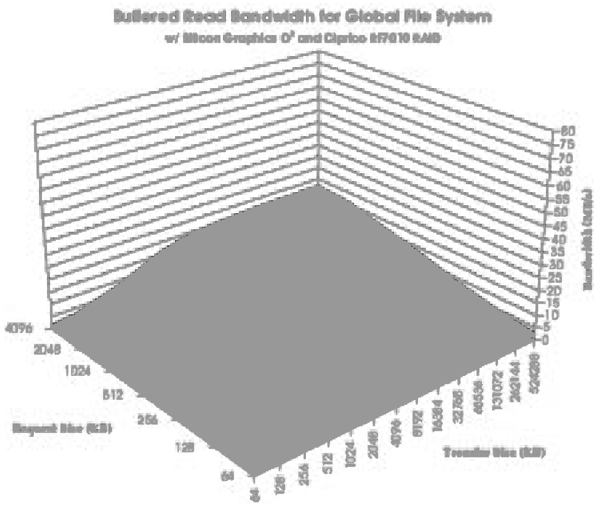
Network Storage Pool Bandwidth

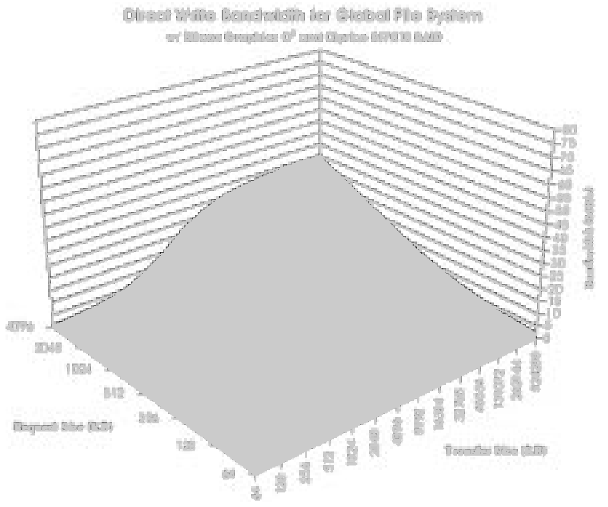


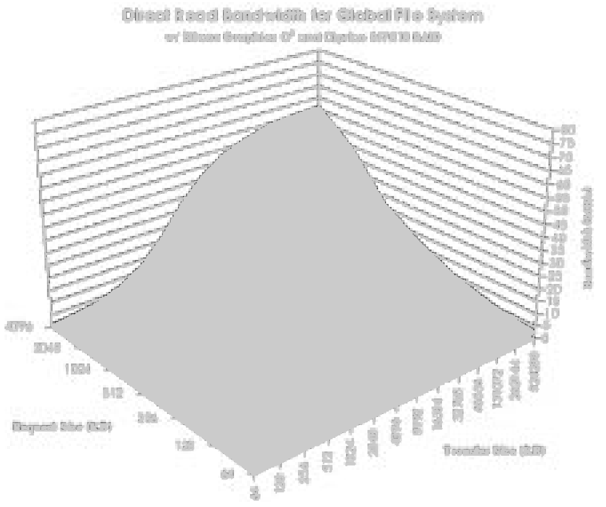


GFS Bandwidth

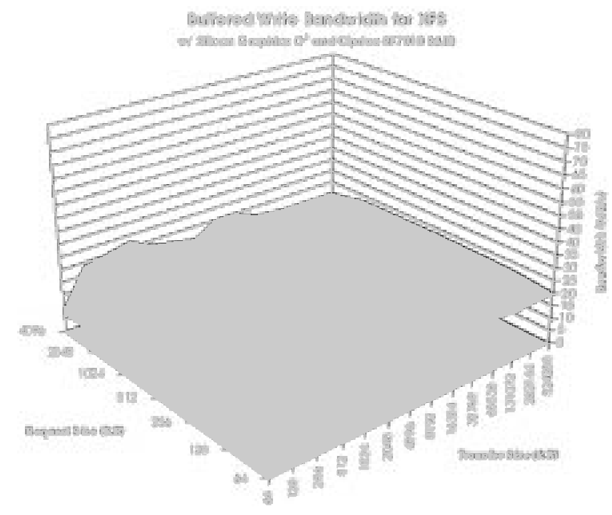


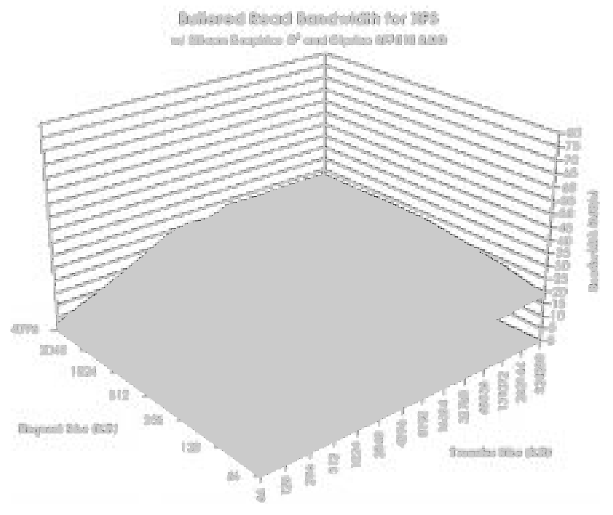


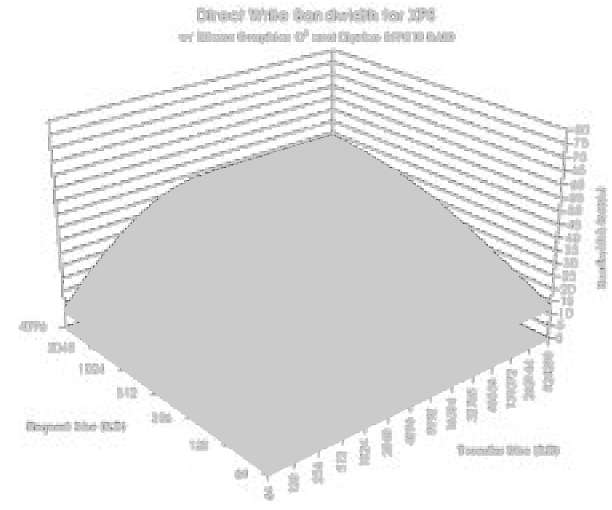


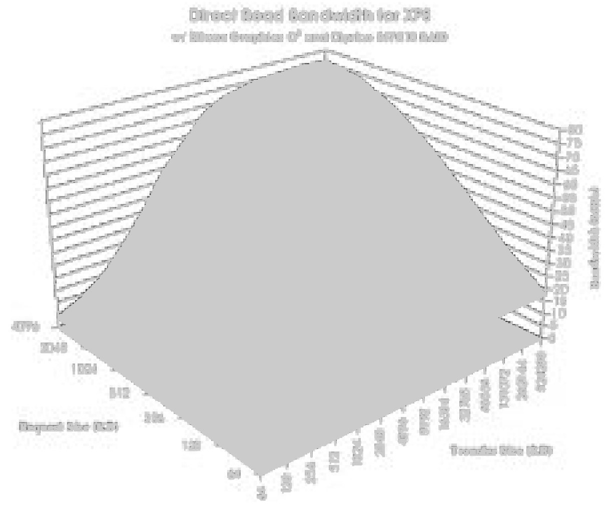


XFS Bandwidth





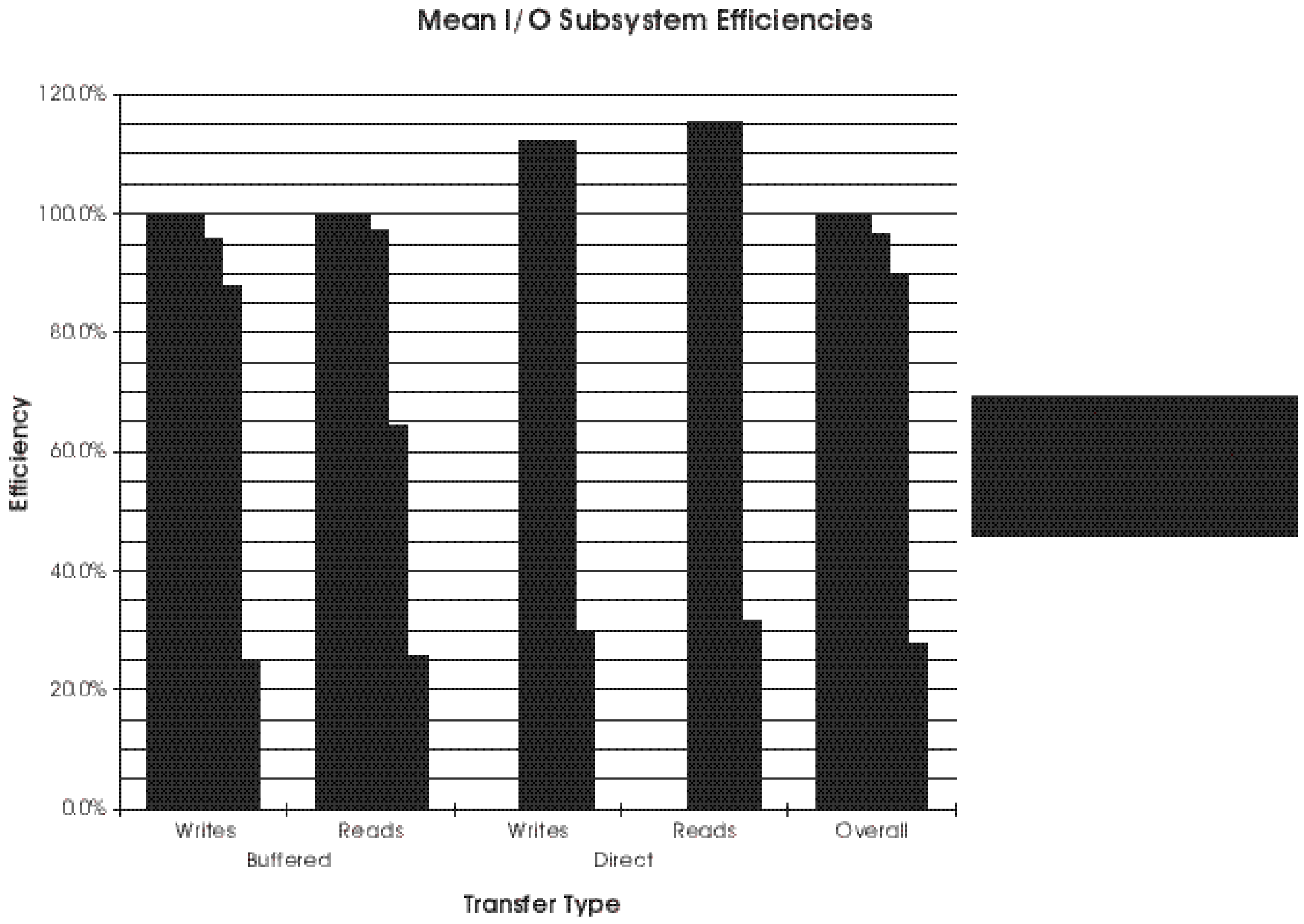




Relative Subsystem Efficiencies

	Relative Efficiency			
	Prisa NetFX Driver	**Network Storage Pool Driver**	**Global File System**	**XFS**
Buffered I/O				
Writes				
Mean	**100.0%**	**95.6%**	**24.9%**	**87.8%**
Standard Deviation	0.00	0.03	0.08	0.90
Minimum	100.0%	90.4%	9.7%	43.7%
Maximum	100.0%	100.5%	38.8%	475.0%
Reads				
Mean	**100.0%**	**97.2%**	**25.7%**	**64.5%**
Standard Deviation	0.00	0.02	0.12	0.58
Minimum	100.0%	92.3%	14.7%	31.8%
Maximum	100.0%	103.6%	68.8%	319.0%
Direct I/O				
Writes				
Mean			**29.6%**	**112.3%**
Standard Deviation			0.19	0.53
Minimum			16.4%	94.6%
Maximum			79.1%	375.8%
Reads				
Mean			**31.4%**	**115.4%**
Standard Deviation			0.17	0.29
Minimum			16.9%	99.4%
Maximum			74.5%	215.9%
Overall				
Mean	**100.0%**	**96.4%**	**27.6%**	**90.0%**
Standard Deviation	0.00	0.03	0.15	0.63
Minimum	100.0%	90.4%	9.7%	31.8%
Maximum	100.0%	103.6%	79.1%	475.0%

Mean I/O Subsystem Efficiencies



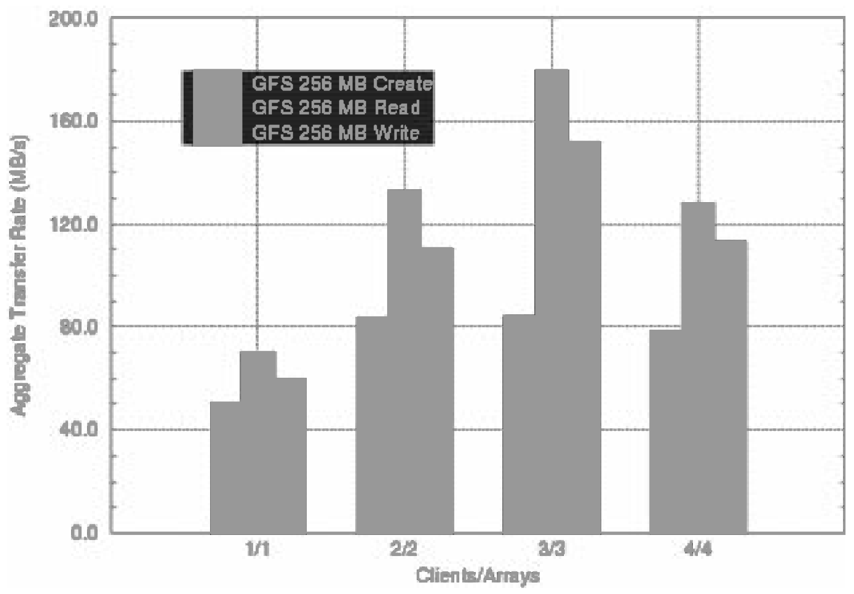
Scaling Studies

- Barrier throughput tests
  - Large transfers size of 256 MB
  - Highly parallel test—each client reads and write its own data from its own device
  - With and without dedicated root directory device
- Test configuration
  - Four *Silicon Graphics Challenge XL* servers
  - *Prisa NetFX HIO-64* Fibre Channel host bus adapter
  - Four *Ciprico Rimfire 7010* Fibre Channel RAID-3s
  - *Brocade Silkworm* 16-port Fibre Channel switch

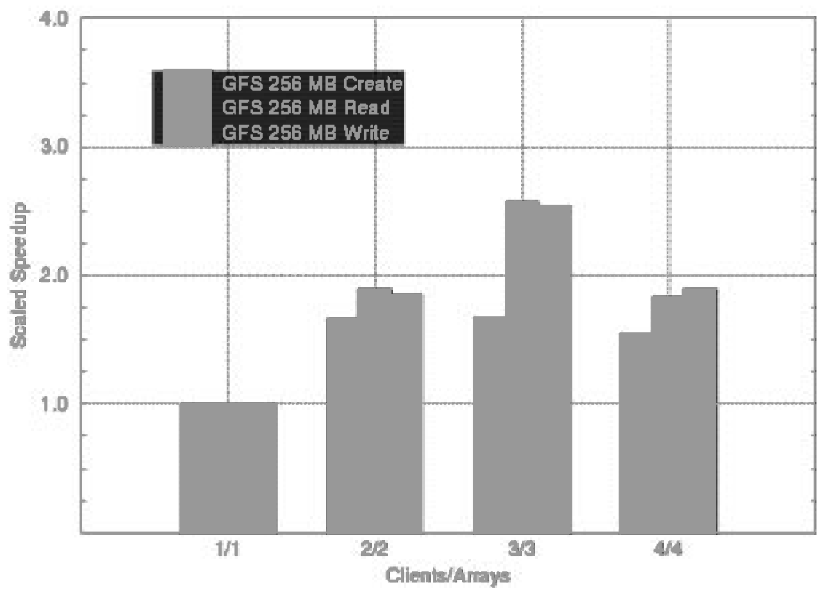
Scalability: Shared Directory Device

- First device contains both the file system root directory and the first client's data.

Throughput



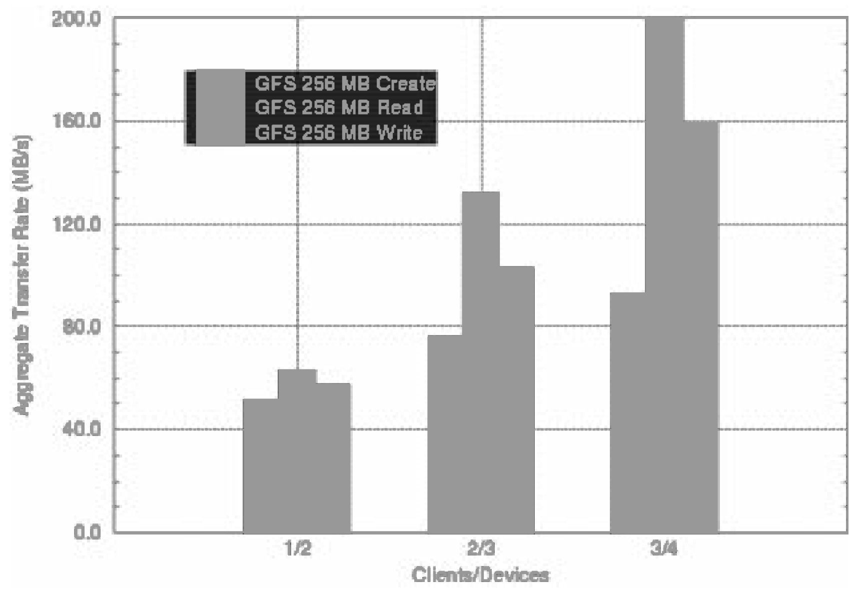
Speedup



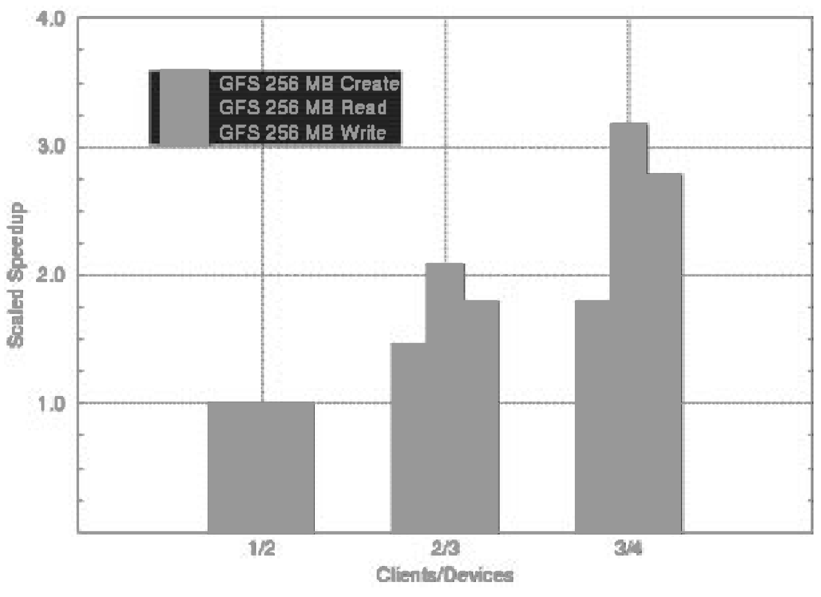
Scalability: Dedicated Directory Device

- First device contains only the file system root directory.

Throughput



Speedup



Future Work

- Ports to open platforms: *Linux*, *FreeBSD*, and *NetBSD*
- Develop heuristics for the optimal sizing of file system blocks and allocation of resource groups at file system creation
- Hide latency of metadata accesses
  - Aggressive management of buffer cache
  - Implement logging
- Quantify performance effects of head-of-queue lock tagging
- Extend locking semantics to improve file system utilization
  - Allow for multiple readers or a single writer
  - Maintain fairness policy close to current implementation
- Scaling to 8, 16, 32 and 64 clients

Conclusions

- Metadata accesses are limiting factor in GFS performance
- Improvements in locking semantics should improve scalability
- GFS architecture is still viable, implementation needs further improvement
- Open licensing
  - Binaries for Silicon Graphics *Irix* 6.2 and 6.3: Today
  - Source code: Summer 1998