# GUPFS: The Global Unified Parallel File System Project at NERSC*

**Greg Butler, Rei Lee, and Mike Welcome**
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
Berkeley, California 94720
{GFButler, RCLee, MLWelcome}@lbl.gov
Tel: +1-510-486-4000

**Abstract**
The Global Unified Parallel File System (GUPFS) project is a five-year project to provide a scalable, high-performance, high-bandwidth, shared file system for the National Energy Research Scientific Computing Center (NERSC). This paper presents the GUPFS testbed configuration, our benchmarking methodology, and some preliminary results.

## 1 Introduction

The Global Unified Parallel File System (GUPFS) project is a multiple-phase, five year project to provide a scalable, high-performance, high-bandwidth, shared file system for the National Energy Research Scientific Computing Center (NERSC) [1]. The primary purpose of the GUPFS project is to make it easier to conduct advanced scientific research using the NERSC systems. This is to be accomplished through the use of a shared file system providing a unified file namespace, operating on consolidated shared storage that is directly accessed by all the NERSC production computational and support systems.

In order to successfully deploy a scalable high-performance shared file system with consolidated disk storage, three major emerging technologies must be brought together: shared/cluster file systems, cost-effective, high performance Storage Area Networks (SAN) fabrics, and high performance storage devices. Although they are evolving rapidly, these emerging technologies are not targeted towards the needs of high performance scientific computing. The GUPFS project is intended to evaluate these emerging technologies to determine the best solutions for a center-wide shared file system, and to encourage the development of these technologies in directions needed for HPC at NERSC.

The GUPFS project is expected to span five years. During the first three years of the project, NERSC intends to test, evaluate, and steer the development of the technologies necessary for the successful deployment of a center-wide shared file system. Provided that an assessment of the technologies is favorable at the end of the first three years, the

* This work was supported by the Director, Office of Science, Office of Advanced Scientific Computer Research, Mathematical, Information, and Computational Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

last two years of the GUPFS project will focus on a staged deployment, leading to full production at the beginning of FY2006.

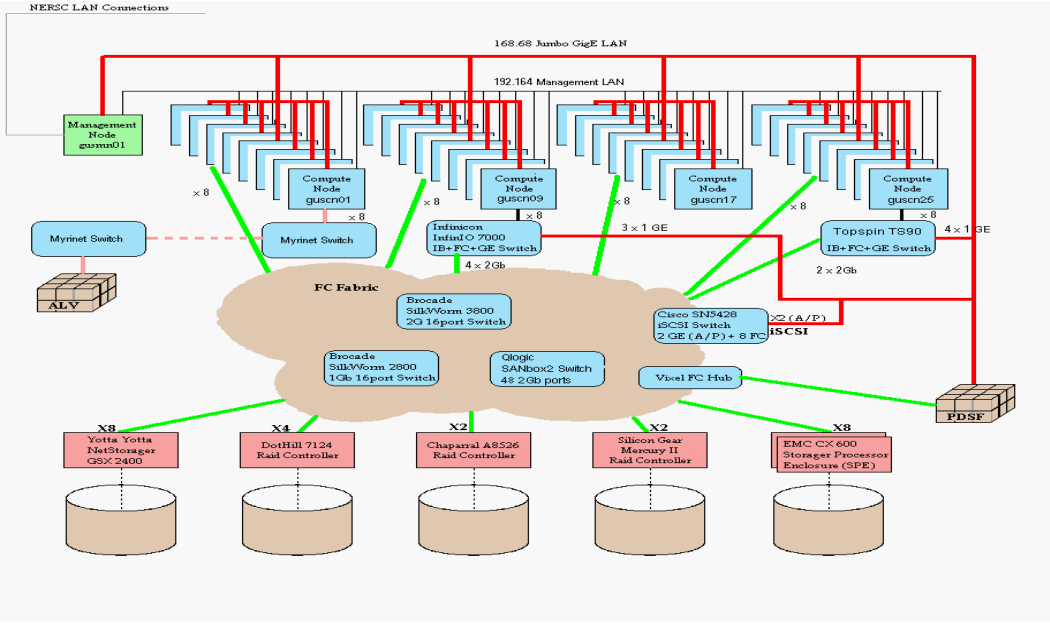
To this end, during the past year the GUPFS project focused on identifying, testing, and evaluating existing and emerging shared and cluster file s ystem, SAN fabric, and storage technologies. During this time, the GUPFS project was also active in identifying NERSC user I/O requirements, methods, and mechanisms, and developing appropriate benchmarking methodologies and benchmark codes for a parallel environment.

This paper presents the GUPFS testbed configuration, our benchmarking methodology, and some preliminary results.

## 2 GUPFS Testbed Configuration

The GUPFS testbed was constructed from commodity components as a small -scale system mimicking a scien tific parallel computational cluster. Its purpose was to assess the suitability of shared -disk cluster file systems with SAN attached storage to the scientific computational cluster environment.

This testbed system presented a microcosm of a parallel sci entific cluster —dedicated computational nodes, special -function service nodes, and a high -speed interconnect for message passing. It consisted of five computational nodes and one management/interactive node, and utilized an internal jumbo frame Gigabit Eth ernet as the high -speed message passing interconnect. An internal 10/100 Mb/s Fast Ethernet LAN was used for system management and NFS distribution of the user home file systems. The configuration of the testbed is illustrated in following diagram.




In designing a testbed for the GUPFS project, a number of factors were considered. The testbed was designed to support the evaluation of three technology areas:

- Shared/cluster file systems
- SAN fabrics
- Storage devices

These three technology areas are key to the successful deployment of a center-wide shared file system utilizing consolidated storage resources.

The testbed is configured as a Linux parallel scientific cluster, with a management node, a core set of 32 dedicated compute nodes and a set of six special-purpose nodes. Each compute node is a dual Pentium IV system with six PCI-X slots. The PCI-X slots allow us to test newer high performance interfaces such as 4x Infiniband HCA. All computer nodes are equipped with a 2Gb/s Fibre Channel HBA and a 1Gb/s Ethernet interface. Various sets of compute nodes are equipped with different groups of interfaces being evaluated such as Infiniband and Myrinet interfaces.

## 3 GUPFS Benchmarking Approach

File systems, and parallel file systems in particular, are extremely complicated and should be evaluated within the context of their intended use. In the commercial world, a file system may be evaluated by how it performs on a single application or a small number of critical applications. For example, a web serving content provider may not be interested in parallel write performance since the primary role of the file system is to provide read-only access to data across a large number of servers without having to replicate the data to multiple farms.

By contrast, the NERSC HPC environment must support a large number of user-written applications with varying I/O and metadata performance requirements. Further, applications running today may not resemble the applications that will be running two years from now. Given this diversity, the GUPFS project is taking a more general, multi-pathed approach to the evaluation of parallel file systems.

Initially, the GUPFS project has performed parallel I/O scalability and metadata performance studies. Later, testing includes reliability and stress-test studies, and finally the project will evaluate the performance with respect to specific I/O applications that emulate real NERSC user codes.

### 3.1 Parallel I/O Performance Studies

Evaluating the performance of a file system, and a parallel file system in particular, can only be done within the context of the underlying system and storage environment. If the underlying storage network or device can only perform at a rate of 30 MB/sec then we cannot expect sustained I/O performance through a file system to exceed this. In addition, for the case of a parallel file system, we need to understand the scalability of the underlying storage network before passing judgment on the file systems' ability to scale to a large number of clients. To aid in this portion of the study, we have developed a parallel I/O benchmark named 'MPTIO' which can perform a variety of I/O operations on files or devices in a flexible manner. MPTIO is an MPI program in which each process spawns multiple threads to perform I/O on the underlying file or device. MPI is

used to synchronize the I/O activity and to collect the per-process results for reporting. It does not use the MPI-I/O library. When acting through a file system, MPTIO can have all threads perform I/O to a single global file, or all the threads of a process can perform I/O to a single file per-process, or all the threads can perform I/O to distinct per-thread files. In addition, I/O can be performed directly to a block device or Linux RAW device to help characterize the underlying storage devices and storage area network. Further, each thread can perform I/O on distinct, non-overlapping regions of the file or device, or multiple threads can perform overlapping I/O. The code can run five different I/O tests, including sequential read and write, random I/O and read-modify-write tests. Aggregate and per-process results are reported. For a complete description of this code, see the *Benchmark Code Descriptions* section in [2].

We can baseline the performance of the storage network and a device using raw device I/O as follows:

- First we measure the I/O rates from a single node to or from the device, by varying the number of I/O threads to the point that the I/O rate saturates.
- Second, we scale up the number of processes (each on a separate node) and also vary the number of I/O threads per process.
- Third, if multiple paths exist through the network to the controller, we can use MPTIO to perform I/O through all paths.

If the raw device I/O rates do not improve past a single node, then the performance bottleneck is in some portion of the network or on the storage device. If they do improve, the first test gives us a good estimation of the peak sustainable raw device I/O rate onto a single node. In this case, as the number of nodes increases, eventually the aggregate raw device I/O rate will again saturate. This bottleneck is either in the network or on the storage controller.

Once a profile of the storage device and network using raw device I/O is complete, a file system can be built over the device. We can then perform I/O and scalability studies over the file system. Since most file systems cache data in host memory, large I/O requests have to be used to minimize the effects of caching. For example, if we observe better performance through the file system than to the RAW device, we can conclude it is the effect of data caching on the node. In addition, one can compare the performance difference between multiple processes writing to the same file versus each writing to different files. If the performance difference is substantial, it will likely be due to write-lock contention between the processes. This might be alleviated if the file system supports byte-range locking so that each process can write its own section without obtaining the single (global) file lock. If the file system supports DIRECT I/O, then one can compare I/O performance through the file system with what was measured to the RAW device. The difference will indicate the amount of software and organizational overhead associated with the file system. For example, file block allocation will probably be distributed across the device resulting in multiple non-contiguous I/O requests to the device. Such I/O performance can degrade as the file system fills and block allocation is more fragmented. File system build options and mount options may also play a

substantial role in the performance. Additional tuning may need to be examined to see how they may affect the I/O performance.

### 3.2 Metadata Performance Studies

In a typical (local) file system, the metadata is often heavily cached in the system memory and updated in an order such that, in the event of a system crash, the file system would be re-constructible from the on-disk image. Modern file systems maintain transaction logs, or journals, to keep track of which updates have been committed to stable storage and which have not. Each of the data structures generally contain one or more locks such that operating system actors wishing to modify the structure do so in an atomic manner, by first acquiring the lock, modifying the structure and then releasing the lock. Some file system operations require modifying many of the structures and thus the actors must acquire multiple locks to complete the operation. In a parallel file system multiple clients, running on different machines under different instances of an operating system will want to modify these data structures (to perform operations) in an unpredictable order. In these cases, where is the metadata is maintained and who controls it can have a dramatic effect on the performance of a file system operation.

There are currently two main approaches to managing metadata in parallel file systems: symmetric (distributed) and asymmetric. In a purely symmetric file system, all the file system clients hold and share metadata. Clients wanting to perform an operation must request locks from the clients holding them via some from of distributed lock manager. In an asymmetric file system, a central (dedicated) metadata server maintains all the information. The clients request updates or read and write access to files. Clearly, this latter case is easier to manage but establishes a single point of failure and may create a performance bottleneck as the number of clients increases. Note that in both cases, once a client is granted read or write access to a file, it accesses the file data directly through the SAN.

One aspect of evaluating a file system is how well it performs various metadata operations required for concurrent file operations by large number of clients in a parallel environment. Clearly, where and how the metadata is maintained will have a substantial impact on the performance of various file system operations. Clients have to send messages to other nodes in the cluster requesting locks, or asking for operations to be performed. The latency of the interconnection network and the software overhead of processing the communication stack will be a major portion of these costs. Apart from the issues of parallel file systems, a portion of the metadata performance will have to do with how the data structures are organized internally. For example, some file systems will maintain a directory structure as a linear linked list whereas others will use more sophisticated schemes, such as hash tables and balanced trees. Linear lists are easy to implement but access and update performance degrades rather seriously as the number of directory entries increase. The other schemes provide near-constant or log-time access and update rates and perform well as the directory grows. This, of course, is at the expense of a more complicated implementation. Another example is how the file system maintains information about which underlying blocks are free or in use. Some systems use a bitmap whereby the value of the bit indicates the availability of the block. Other

systems use extent-based schemes whereby a small record can represent the availability of a large region of contiguous blocks.

In our study, we measured how the various parallel file systems perform with respect to certain metadata intensive operations. To this end, we have developed a file system metadata benchmark code called 'METABENCH'. This is a parallel application that uses MPI to coordinate processes across multiple file system clients. The processes perform a series of metadata operations, such as file creation, file stating and file utime and append operations. Details on the current state of METABENCH can be found in the *Benchmarking Code Descriptions* section in [2].

### 3.3 User Applications Emulation

Although micro benchmarks such as MPTIO and METABENCH can provide a wealth of information about how a file system behaves under controlled conditions, the true test of a file system is how it performs in a real user environment. The NERSC user community is large, with a diverse collection of codes that evolve over time. In addition, the codes are complex and may not easily be ported to our test system, or even scale down to that size. Further, the I/O and file operation portion of the code may only consume a small portion of the run-time so attempting to run the actual application on the testbed would be inefficient. In order to address these issues, we plan to develop a small collection of I/O applications that emulate the I/O and file management behavior of real NERSC user applications. This will be a time-consuming task and will only be successful with the aid of the user community. We have begun an informal survey of some of the larger NERSC projects to understand their I/O requirements. As a part of this, we will select a few applications and solicit the users to help us create an I/O benchmark that emulates their code.

## 4 Preliminary Performance Results

During the past year, we have evaluated a number of products and technologies that we believe are key technologies to the GUPFS Project. We will present testing results in this section for some of the following products and technologies evaluated:

- File Systems: Sistina 5.1 & 5.2 Beta; ADIC StorNext (CVFS) File System 2.0; Lustre 0.6 (1.0 Beta 1) and 1.0; and GPFS 1.3 for Linux
- Fabric Technologies
    - Fibre Channel Switches: Brocade SilkWorm and Qlogic SANbox2-16 and SANbox2-64
    - ISCSI[3]: Cisco SN 5428, Intel iSCSI HBA, iSCSI over IB
    - Infiniband: InfiniCon and Topspin (IB to FC and GigE)
    - Inter-connect: Myrinnet, GigE
- Fibre Channel Storage Devices
    - 1Gb/s FC: Dot Hill, Silicon Gear, Chaparral
    - 2Gb/s FC: Yotta Yotta NetStorager[8], EMC CX 600, 3PARdata

### 4.1 Storage Performance and Scalability

Storage can be a performance bottleneck of any file system. A storage device may be able to sustain a very good single -port performance. However, having good single -port performance is not sufficient for a shared disk file system like GUPFS. For GUPFS, the underlying storage devices must demonstrate a very good scalability wh en the number of clients increases (to thousands or tens of thousands). A shared file system will not scale if the underlying storage does not scale.

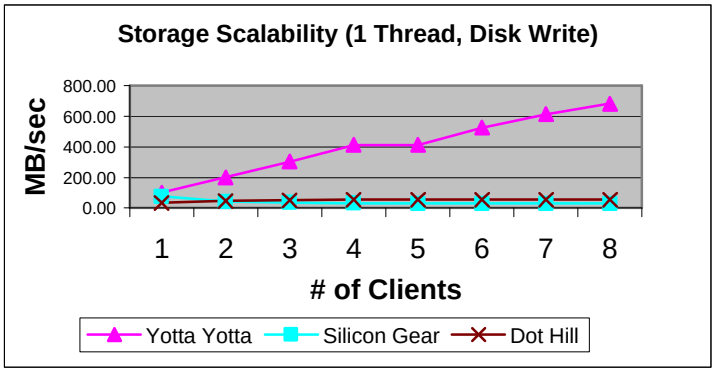



**Figure 1. Storage Scalability**

Figure 1 shows how st orage devices scale when the number of clients increase. The figure shows the results of three storage devices: Yotta Yotta GSX 2400 (YY), Silicon Gear Mercury II (SG), and DotHill SANnet (DH). Both Silicon Gear and Dot Hill have only 2 1Gb/s front-end ports, while Yotta Yotta has 8 2Gb/s ports and 3PARdata has 16 2Gb/s ports but only 8 were used during the test. On each storage device, we created a single LUN to be shared by multiple clients for shared access.

The figure shows that both the Silicon Gear and Dot Hill devices did not scale when the number of clients increased. Silicon Gear performance actually dropped when the number of clients increased. On the other hand, the figure shows that the Yotta Yotta storage did scale very well when the number of clients increased.

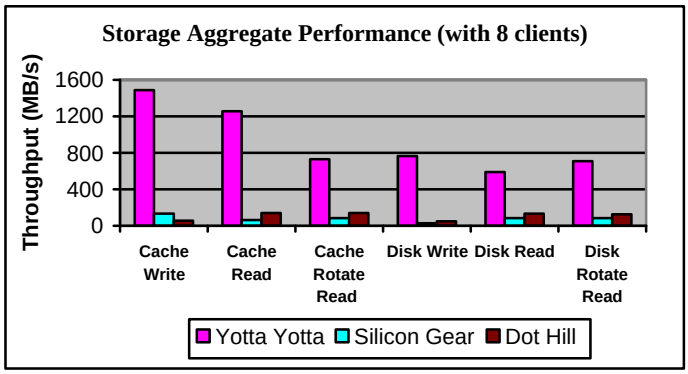



**Figure 2. Storage Aggregate Performance**

Figure 2 shows the aggregate performance of the three storage devices: DotHill, Silicon Gear, and Yotta Yotta, using the MPTIO benchmark w ith different test conditions. The

results indicate that the Yotta Yotta storage will be able to sustain higher performance than Silicon Gear or Dot Hill in a shared file system.

### 4.2 Parallel File I/O Performance

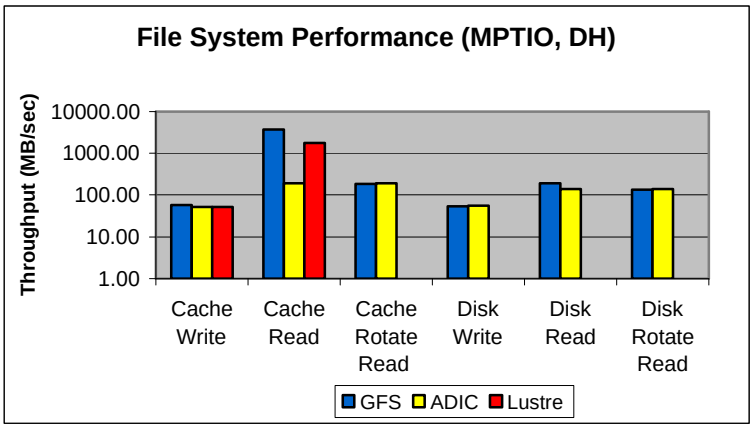



**Figure 3. Shared File System Performance**

During the last year, we have tested several file systems, including Sistina's GFS [4], ADIC's StorNext File System [5], and Lustre [7]. The above diagram shows the 8 -client MPTIO results on these file system, under different test scenarios. These results indicate that there is not much difference in parallel I/O performance between GFS and ADIC's StorNext File System, except for 'Cache Read'. ADIC's StorNext File System was probably doing direct I/O even when operating on files that can fit in the OS cache.

With the award of the ASCI PathForward SGSFS [6] file system development contract to HP and Cluster File Systems, Inc., there has been rapid progress on the Lustre file system [7]. The earlier Lustre file system version (0.6) we tested failed to complete all but the 'Cache Write' and 'Cache Read' tests. Luster1.0.0 was recently released and Figure 4 shows the latest result of Lustre scalability with six clients and two Object Storage Servers (OSS).

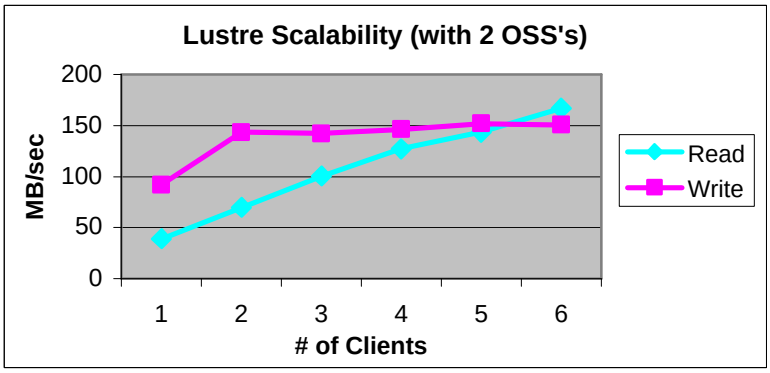



**Figure 4. Lustre Scalability**

The figure seems to indicate that reads and writes were limited by the GigE interface as the test was running with two OSS's and each OSS was equipped with one GigE interface. Additional test with more OSS's and tuning should improve performance.

### 4.3 Fabric Performance

Storage area networks, by providing a high performance network fabric oriented toward storage device transfer protocols, allow direct physical data transfers between hosts and storage devices. Currently, most SANs are i mplemented using Fibre Channel (FC) protocol-based fabric. Emerging alternative SAN protocols, such as iSCSI (Internet Small Computer System Interface), FCIP (Fibre Channel over IP), and SRP (SCSI RDMA [Remote Direct Memory Access] Protocol) [9], are enabl ing the use of alternative fabric technologies, such as Gigabit Ethernet and the emerging InfiniBand, as SAN fabrics.

Here we present some performance results of several fabric technologies: Fibre Channel (FC), iSCSI over GigE, iSCSI over IP over Infiniband (IPoIB), and SRP.

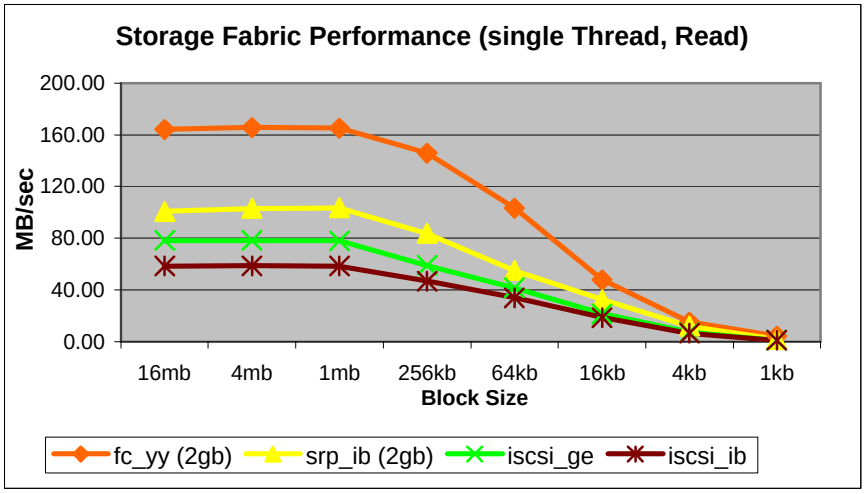



**Figure 5. Storage Fabric Performance**

Figure 5 shows the results of single -thread reads of different I/O size using different fabric technologies. The best performance was achieved by the 2Gb/s FC interfa ce, followed by the SRP protocol over Infiniband. Since the iSCSI traffic was passing through a single GigE interface, the iSCSI performance was less than 100 MB/s. With the additional stack overhead of IPoIB, iSCSI over IPoIB delivered the lowest performa nce for single-thread reads.

Figure 6 shows the CPU overhead of different protocols for single-thread reads. FC, while delivered the best performance, used the least CPU overhead. The iSCSI protocol allows the standard SCSI packets to be enveloped in IP packets and transported over standard Ethernet infrastructure, which allows SANs to be deployed on IP networks. This option is very attractive as it allows lower-cost SAN connectivity than can be achieved with Fibre Channel, although with lower performance. It will allow large numbers of inexpensive systems to be connected to the SAN and use the shared file system through

commodity-priced components. While attractive from a hardware cost perspective, this option does incur a performance impact on each host due to increased traffic through the host's IP stack, as shown in Figure 6.

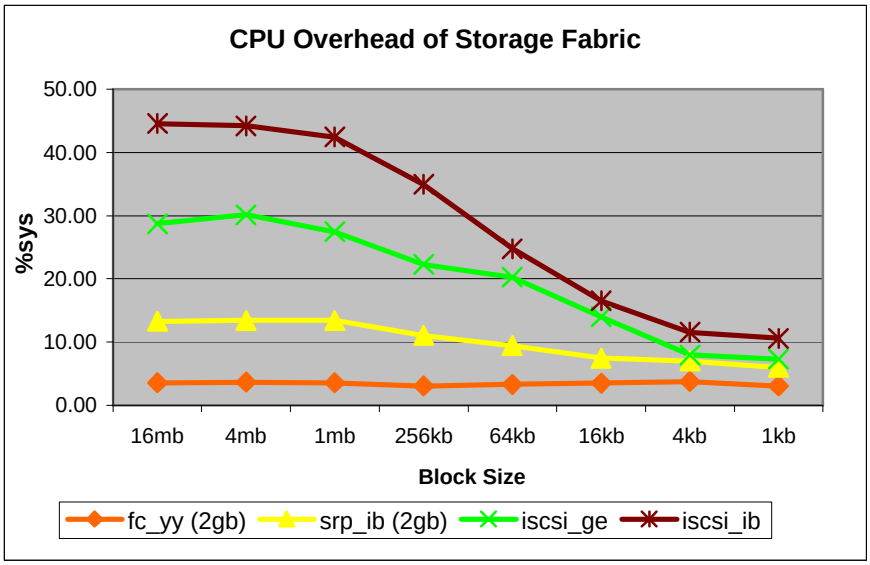



**Figure 6. CPU Overhead of Storage Fabric**

## 5 Conclusions

The GUPFS project started in the last half of FY 2001 as a limited investigation of the suitability of shared-disk file systems in a SAN environment for scientific clusters, with an eye towards possible future center-wide deployment. As such, it was targeted towards initial testing of the Sistina Global File System (GFS), and included a small testbed system to be used in the investigation.

With the advent of the NERSC Strategic Proposal for FY 2002 –2006, this modest investigation evolved into the GUPFS project, which is one of the major programmatic thrusts at NERSC. During the first three years of th e GUPFS project, NERSC intends to test, evaluate, and influence the development of the technologies necessary for the successful deployment of a center-wide shared file system. Provided that an assessment of the technologies is favorable at the end of the first three years, the last two years of the GUPFS project will focus on a staged deployment of a high performance shared file system center-wide at NERSC, in conjunction with the consolidation of user disk storage, leading to production in FY 2006.

## Reference

[1] NERSC Strategic Proposal FY2002-FY2006, http://www.nersc.gov/aboutnersc/pubs/Strategic_Proposal_final.pdf.
[2] The Global Unified Parallel File System (GUPFS) Project: FY 2002 Activities and Results, http://www.nersc.gov/aboutnersc/pubs/GUPFS_02.pdf.
[3] Julian Satran, "iSCSI" (Internet Small Computer System Interface) IETF Standard, January 24, 2003, http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-20.pdf.

[4] Global File System (GFS), Sistina Software, Inc., http://www.sistina.com/downloads/datasheets/GFS_datasheet.pdf.
[5] StorNext File System, Advanced Digital Information Corporation (ADIC), http://www.adic.com/ibeCCtpSctDspRte.jsp?minisite=10000&respid=22372&section=10121.
[6] ASCI Path Forward, SGSFS, 2001, http://www.lustre.org/docs/SGSRFP.pdf.
[7] "Lustre: A Scalable, High-Performance File System," Cluster File Systems, Inc., November 2002, http://www.lustre.org/docs/whitepaper.pdf.
[8] Yotta Yotta NetStorager GSX 2400, Yotta Yotta, Inc., http://www.yottayotta.com/pages/products/overview.htm.
[9] *SRP:* SCSI RDMA Protocol: ftp://ftp.t10.org/t10/drafts/srp/srp-r16a.pdf.