



Evaluation of dCache (Chimera/NFSv4.1) for Data Storage Applications in the Intensity Frontier Experiments

Art Kreymer, Stephan Lammel, Margaret Votava, and Michael Wang

Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

December 14, 2011

1. Introduction

The dCache system [1] is a distributed data caching system based on a large number of heterogeneous (and often commodity-component-based), network-connected disk storage servers offering a unified, rooted filesystem tree view of the entire data repository. A key feature is the separation of the file namespace of the repository, which is managed by a database, from the actual location of the data files. This allows files to reside anywhere in the repository and the possibility of multiple copies of the same file. Other key features include load balancing through caching mechanisms, user authentication, and easy integration of tertiary storage systems like robotic tape libraries. The dCache product, which is jointly maintained by the Deutsches Elektronen-Synchrotron (DESY) and Fermi National Accelerator Laboratory, has a wide installation base in the High Energy Physics community. The CDF collider experiment at Fermilab, for example, has nearly a petabyte (~950TB) of dCache-based storage space.

Despite its success, dCache has seen only limited use in the experiments of the Intensity Frontier (IF) program at Fermilab. One major reason is the limited functionality of the “filesystem” view dCache exports, via NFS, to its clients. While standard, unix-style file namespace manipulations (like *ls*, *mv*, *rm*) are supported, direct POSIX-compliant I/O (*open*, *create*, *read*, *write*...) is not. In essence, only the meta-data associated with the data files (and not the data files themselves) are distributed to the clients through NFS, requiring the use of a special dCache Access Protocol (DCAP) to access the actual data files. Partly because of this limitation, much of the disk storage needs of the IF experiments is satisfied using commercial high performance BlueArc NFS servers [2].

Recent developments in the evolution of the NFS standard, however, have provided a way to address the limitations described above. The new NFS Version 4.1 protocol [3] includes a specifi-

cation for parallel NFS (pNFS) which is backed by various industry giants (Panasas, IBM, EMC, etc.). Similar to the dCache approach, it extends NFS by moving the metadata server out of the data path. The metadata server, which provides a single namespace of the data repository, presents a map or *layout* of the requested data to the pNFS clients which, in turn, interact directly with the data servers. Because of the similarities in their basic architectures, dCache would seem the ideal environment to implement the new NFSv4.1 protocol.

The latest version of dCache (Version 1.9.12+) does, in fact, implement a new *Chimera* namespace server that is based on NFSv4.1. This new version of dCache supports POSIX-compliant I/O. Being based on an open standard like NFSv4.1, it also takes advantage of standard pNFS client implementations now available in the latest Linux kernels, completely eliminating the need for special protocols like DCAP.

In this document, we will look in some detail at the new dCache that uses the NFSv4.1 *Chimera* namespace server, with the goal of evaluating its applicability as a data storage solution for IF experiments. We will begin by measuring and characterizing its basic I/O performance with commonly used benchmarking tools, and studying the behavior of its system components in response to increasing loads. This will be followed by custom tests that better represent its application in the data handling environment of a typical IF experiment. We hope the results presented here will provide valuable input to the process of planning the data handling infrastructure of the IF experiments.

2. Test Setup

A block diagram of the test dCache server used in this evaluation is shown in Figure 1. It consists of a head node and two pool nodes running dCache version 1.9.12 with version 2.6.18 Linux kernels (Scientific Linux Fermi 5.3 distribution). Hardware configuration details of each machine are listed in Table 1. Each pool node is attached to a Nexsan SATABlade disk array through a 2Gb Fiber Channel link. Each array consists of eight 250GB SATA drives (Hitachi Deskstar HDT722525DLA380) configured as two RAID 6 partitions of four drives each, which show up as `/dev/sdb1` and `/dev/sdb2` under Linux on the pool nodes. All four partitions are exported by the head node as a single NFSv4.1 filesystem. The exported filesystem is mounted by clients running pNFS/NFSv4.1 enabled version 2.6.40 Linux kernels (replacing the default kernel in the Scientific Linux Fermi 6.1 distribution) on virtual machines (VMs) in the Fermicloud cluster. Configuration details of each VM are shown in Table 2. Network access between the clients and the dCache server is via Gigabit ethernet (1 GigE).

To better understand the limitations imposed on the dCache system by the capabilities of the disk hardware and network infrastructure, simple benchmarks were run to characterize their performance. Using the *hdparm* utility program (with `-t -T` options), each partition was found to achieve ~ 150 MB/s when reading from the disk without prior caching of the data. The network performance between the Fermicloud VM clients and the dCache server was measured to be ~ 120 MB/s using the *ttcp* utility program.

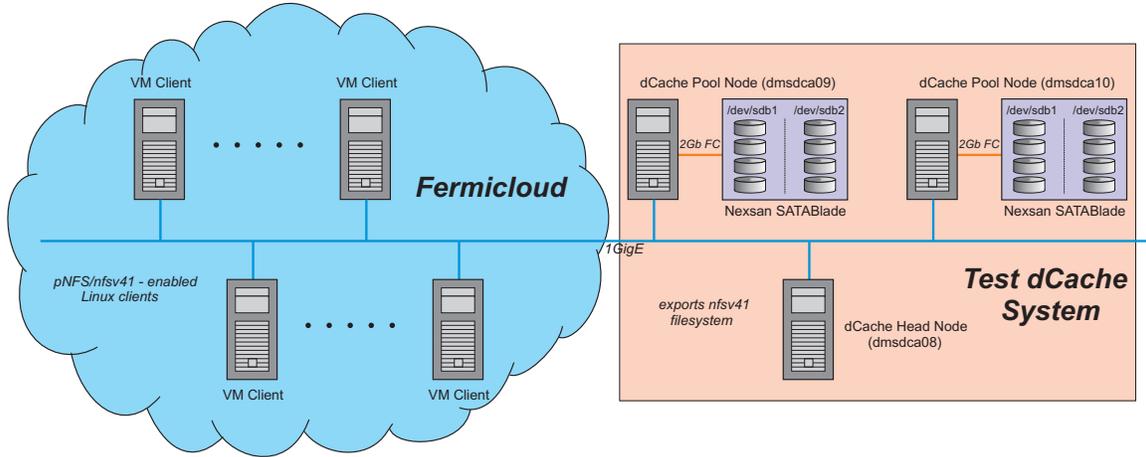


Figure 1: dCache setup

3. IOzone Throughput Studies

3.1. Configuration

Basic write and read performance of the exported dCache filesystem is evaluated using the IOzone Filesystem Benchmark (version 3.397) [4] program installed on the Fermicloud VM clients. IOzone is run in distributed mode (`-+m client.list`) with a *master* process running on one machine coordinating *slave* processes running remotely on pNFS enabled client machines that do the actual I/O. Results from each client are sent back to the controlling process which measures the aggregate throughput. Only sequential writes and reads are performed with no retests specified (`-+n`) due to the immutability of dCache files which prevents modification of the files once they are created. A fixed file size of 4 GB, chosen to be double the available memory (RAM) on each VM, is used for all tests.

3.2. Monitoring of System Activity

As the tests are conducted, performance metrics are collected from the dCache head and pool nodes and from each of the VM clients using the Performance Co-Pilot monitoring tool (version 3.5.8) [5] in order to understand dCache system behavior better.

Figure 2 shows histograms in time of disk I/O in MB/sec for all four partitions on the two dCache pool nodes. Distributions for each partition of `dmsdca09` and `dmsdca10` are individually shown in the top two rows and bottom two rows, respectively. Vertical black lines indicate the start of the sequential write test for a given number of clients. These black lines are followed by vertical red lines indicating the start of the sequential read test of the newly created files. Histograms for memory usage and network I/O for the dCache pool nodes, corresponding to those for disk I/O in Fig. 2, are shown in Fig. 3. Corresponding histograms for memory usage and network I/O for all VM clients are also shown in Figs. 4-7. The same vertical lines indicating the start of write and read tests are shown on all plots.

As indicated in the lower left corner of Fig. 2, when the tests begin with a single client, the file is initially written to one partition on `dmsdca10`. In the region immediately to the right of

the leftmost vertical red line, one sees that there is no visible disk activity in the ensuing read test, since the 4GB file is read using the cached copy in the pool node's 8GB memory buffer (see corresponding memory usage distributions in Fig. 3). As the number of clients increases and the amount of free memory on the pool node decreases, disk activity begins to show up in the read test when files are read directly from the disk.

For many of the tests, especially those for dmsdca10, the distributions indicate a significant amount of disk read activity occurring even before the start of the read test. This is due to the default dCache policy of calculating checksums on the file written to disk. We repeated the tests for 3 and 4 clients after disabling the *checksum on write* feature and verified that the yellow distributions representing disk read activity occurring before the start of the read test do indeed disappear.

One also observes significantly less disk read activity occurring on dmsdca09 in the top two rows of Fig. 2. This is due to the fact that the memory buffer on this node is twice as large as that for the other node and files are read using cached copies. As Fig. 3 indicates, the memory buffer on this node only begins to fill up when the tests are done with nine clients in the upper right hand corner of the figure.

3.3. Results

The aggregate throughput measured for the sequential write and read tests conducted with IOzone described above are plotted as a function of the number of clients in Figure 8. As the plot shows, in the initial half of the test, the write rates average about 125 MB/sec and rise abruptly to over 250 MB/sec in the second half of the test. This "turn-on" is due to the initiation of write activity in the second dCache pool node (dmsdca09) as indicated by the leftmost distribution of the network activity plot in the second row of Figure 3. A similar trend is seen for the read rates. In both halves of the plot defined by the sharp "turn-on", the I/O rates saturate at levels imposed by each pool node's network connection (~ 120 MB/sec). The fact that the write rates seem to exceed the network capabilities of the pool nodes might be attributed to caching effects on the clients. To some extent, the performance of a dCache system, represented by a single filesystem, can be improved by *scaling-out* and increasing its throughput in proportion to the number of pool nodes in response to growing data-handling needs.

3.4. Extending the Throughput Tests to Many More Clients

In this next study, we use IOzone to test the ability of dCache to handle an increasing number of concurrent clients beyond ten. We use a total of ten Fermicloud VMs and distribute the total number of IOzone clients evenly among the ten VMs. The aggregate amount of data transferred by all clients is always fixed at 40GB (or 4GB per VM) with each client transferring $40GB/n$ where n is the total number of clients. Except for the data transfer size, we use the same IOzone settings as those described in Section 3.1. We perform our tests for up to a total of 200 concurrent clients since this is close to IOzone's maximum of 256 clients in cluster mode. Unlike the previous study which focuses on maximum throughput and how it scales with more pool nodes, this study largely looks at the ability of the metadata server component of the dCache head node to deal with a large number of concurrent clients. To gauge this ability, we pay attention to the constancy of the combined throughput of all clients as a function of the number of concurrent clients. In Fig. 9, we plot the combined read rate of all clients versus the number of clients ranging from 20 up to 200 clients and find that the rate remains constant throughout the range.

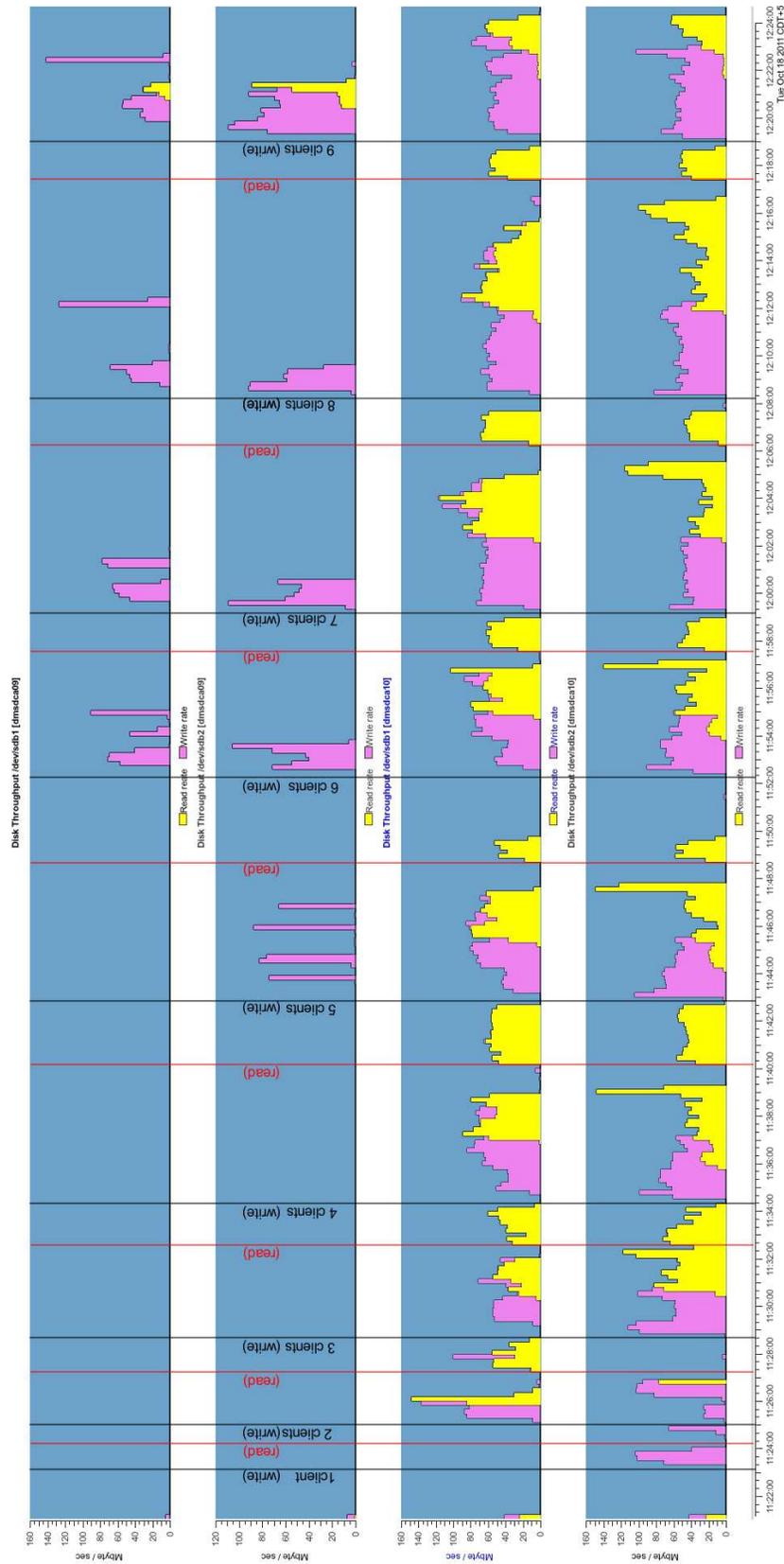


Figure 2: Disk throughput on deache pool nodes.

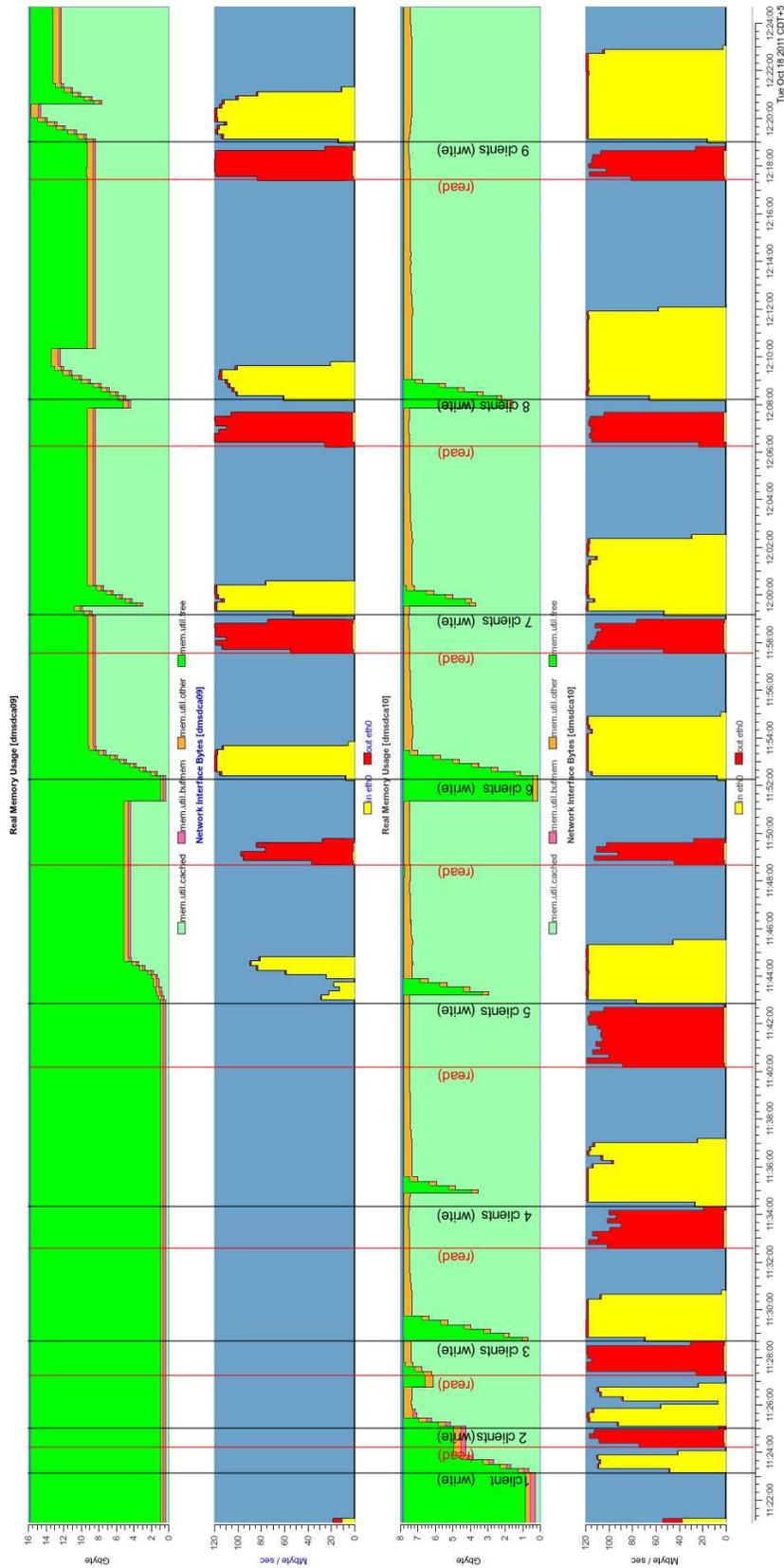


Figure 3: Memory usage and network activity on dcache pool nodes.

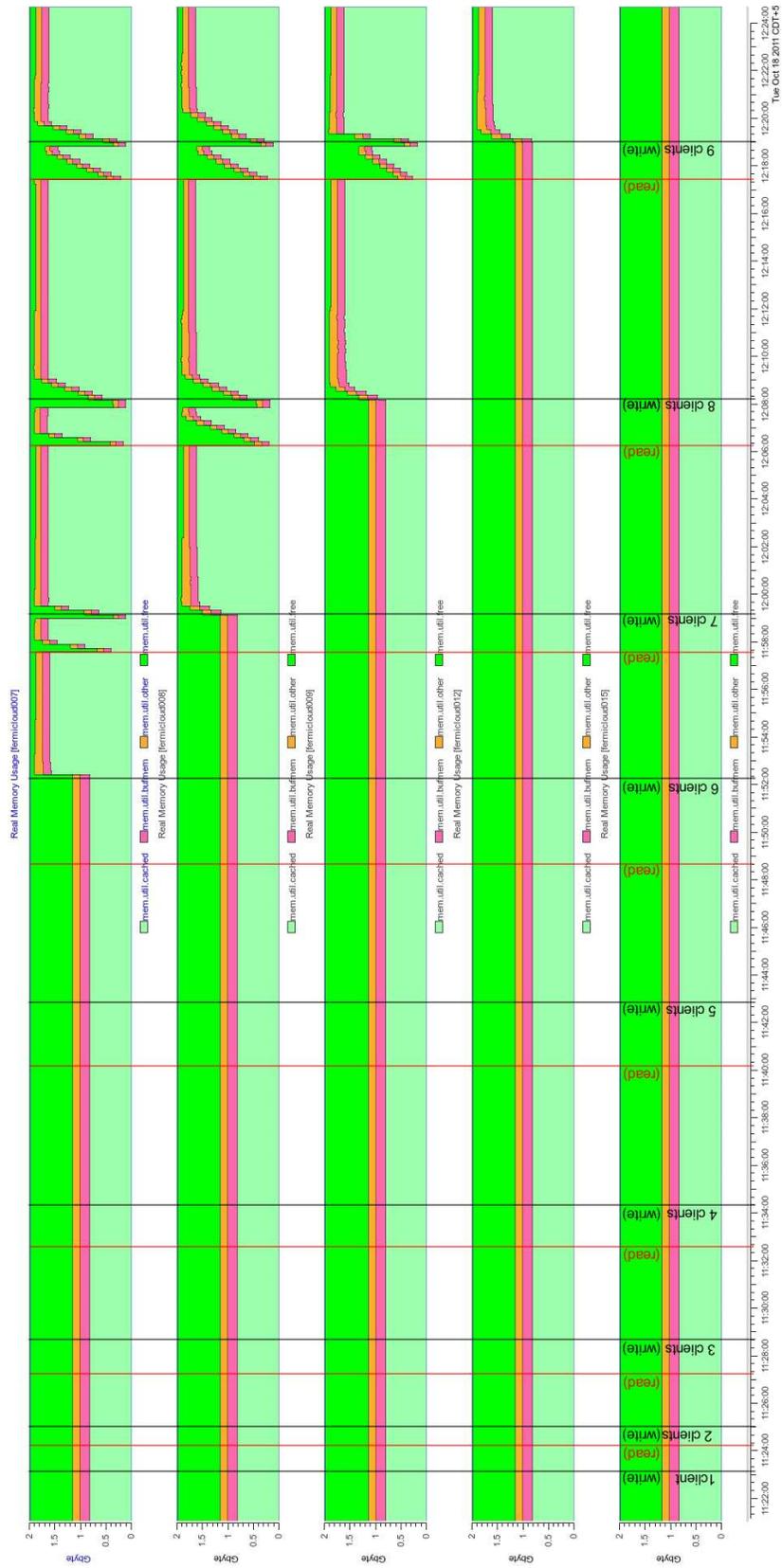


Figure 5: Memory usage on second group of five Fermicloud virtual machine clients.

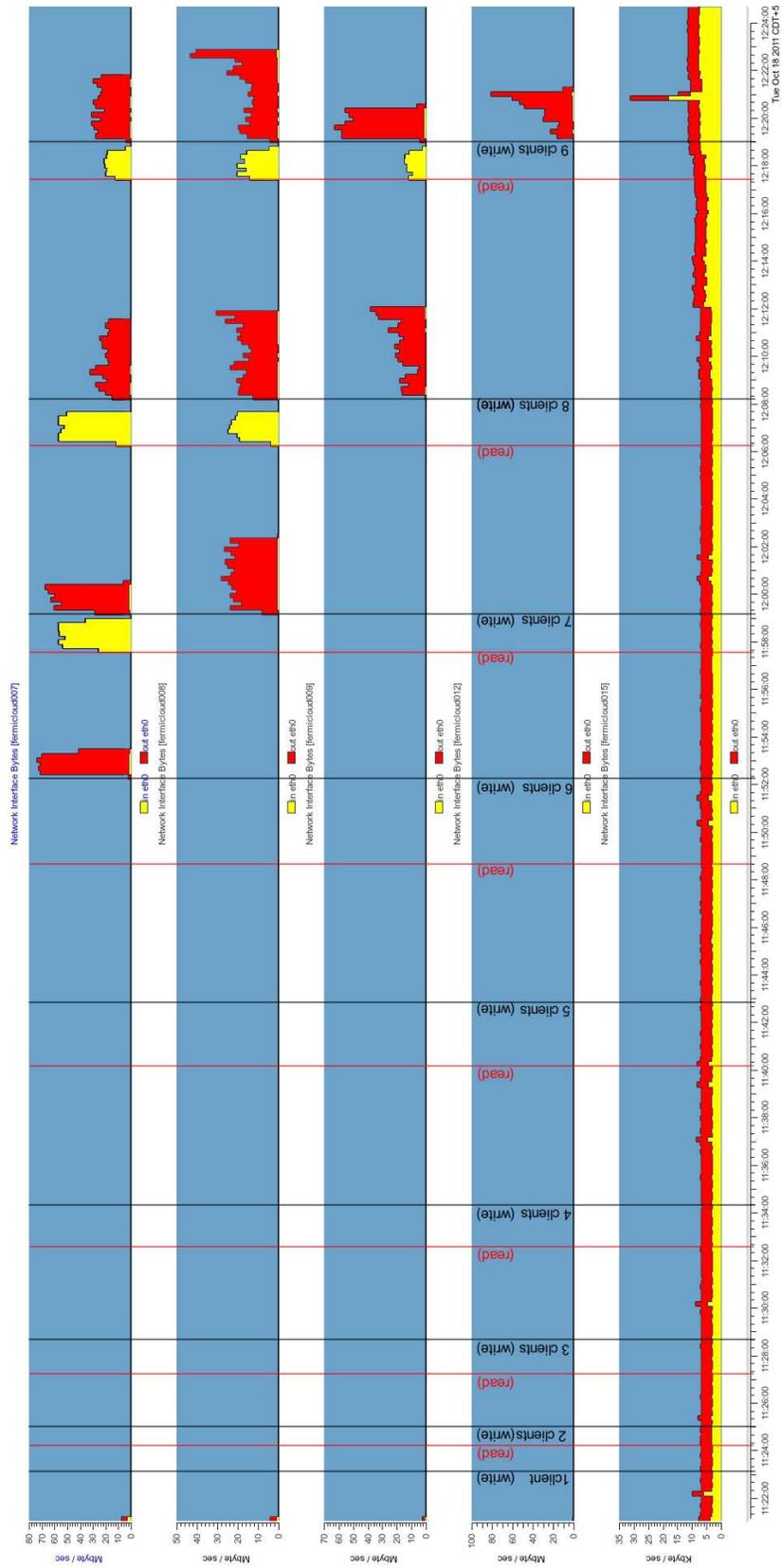


Figure 6: Network activity on first group of five Fermicloud virtual machine clients.

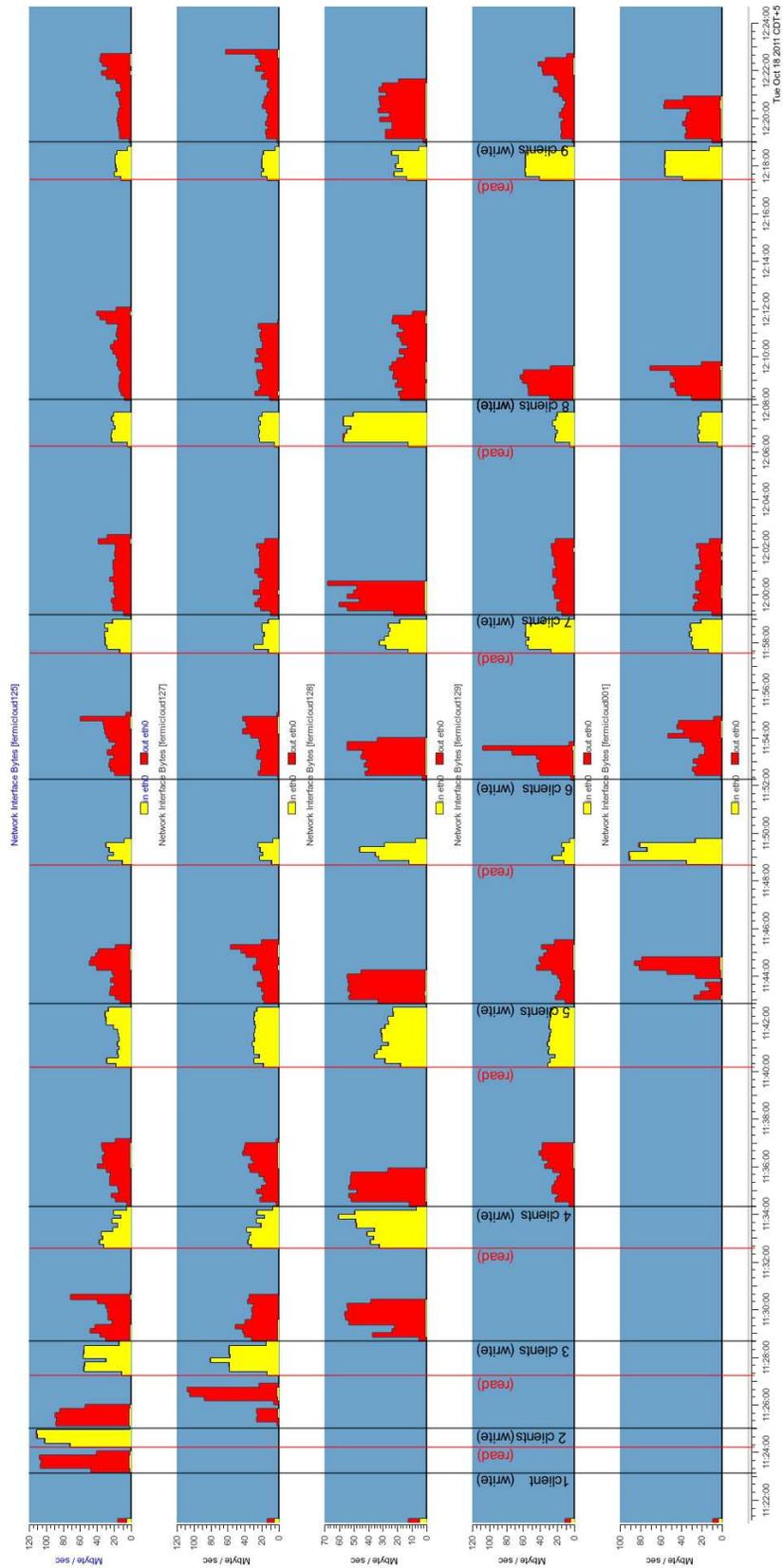


Figure 7: Network activity on second group of five Fermicloud virtual machine clients.

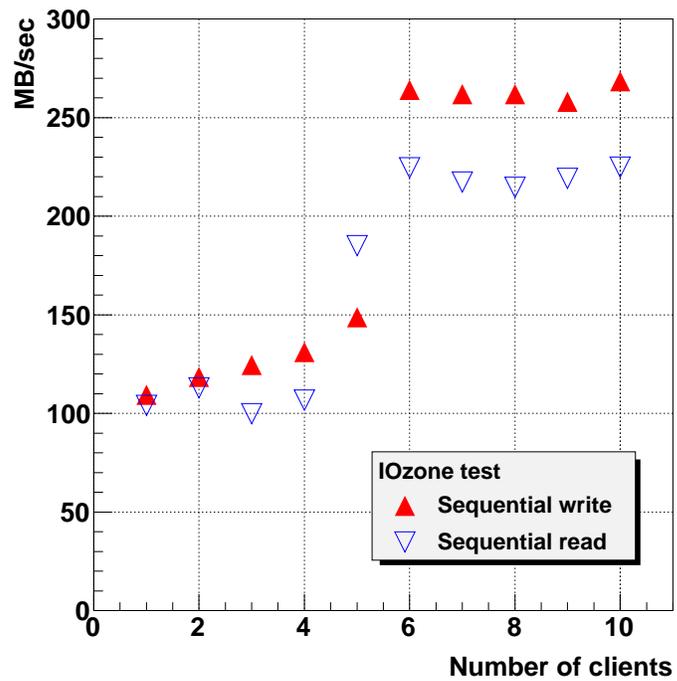


Figure 8: Results of the sequential write and read tests using IOzone are shown as a function of the number of clients.

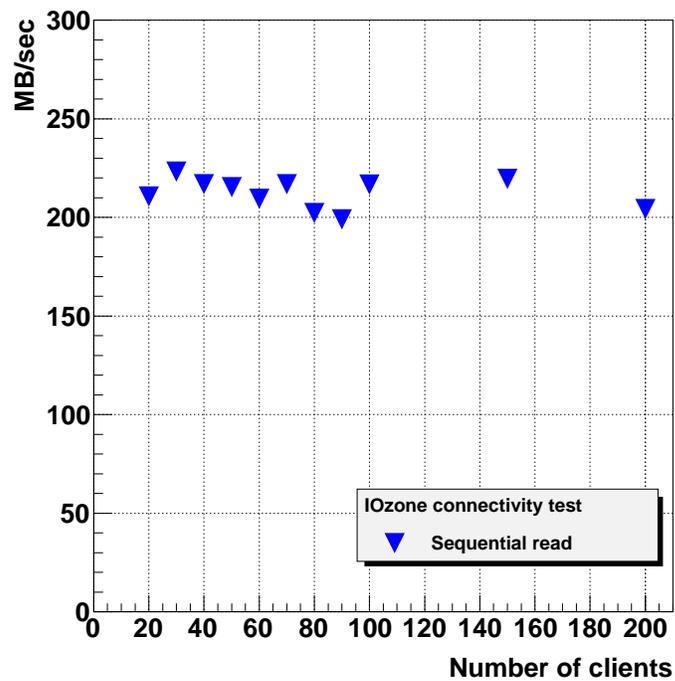


Figure 9: Results of the IOzone connectivity tests investigating dCache performance for many simultaneous clients (> 10). The aggregate data read rate is shown as function of number of clients for a fixed total amount of data (40GB) evenly distributed among the clients.

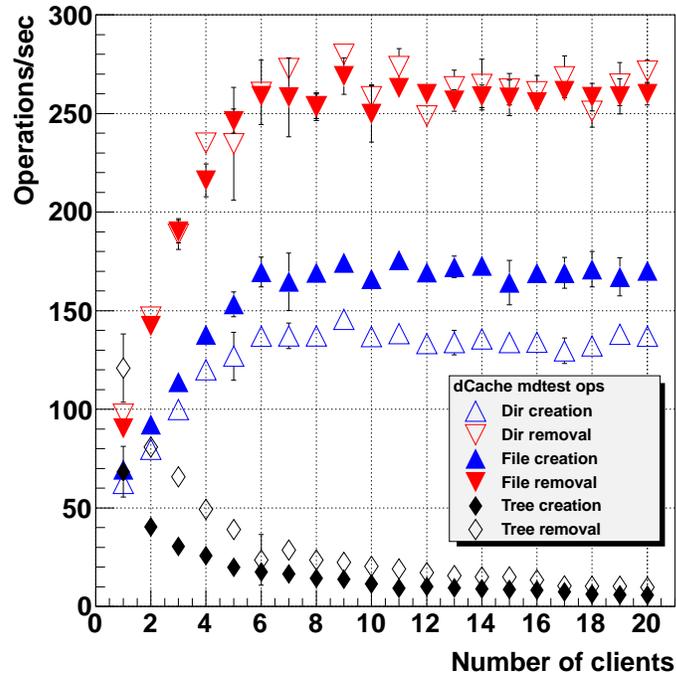


Figure 10: *mdtest* results for directory, file, and tree creation and removal.

4. Metadata Performance Measurements

We use the *mdtest* scalable I/O benchmark to evaluate the metadata performance of the test dCache system. We run *mdtest* in parallel with up to 20 concurrent MPI clients, each of which runs on a separate Fermicloud VM. Each client creates, stats, and removes 100 directories and 100 zero-byte sized files in separate directories and the rate of these operations is measured and reported by *mdtest*. We use the “read-your-neighbor” option in *mdtest* to avoid reading locally cached files by forcing stat operations to be performed on the files and directories created by other nodes. The measured rates are shown in Figs. 10 and 11 as a function of the number of clients.

We see from these plots that the rates for file and directory operations rise fairly linearly up to about 6 or 8 clients and plateau beyond that. When looking at these results, it is important to keep in mind that the Chimera namespace server used in dCache has a database backend, which is a PostgreSQL database in the current test setup. Since database access is provided by the Java DataBase Connectivity (JDBC) API with no database implementation specific binding, Chimera is not tied to any particular database implementation and can benefit from performance-enhancing techniques available on the different platforms. It is also not crucial to have the best possible metadata performance if dCache will be used primarily for the sequential I/O of large data files.

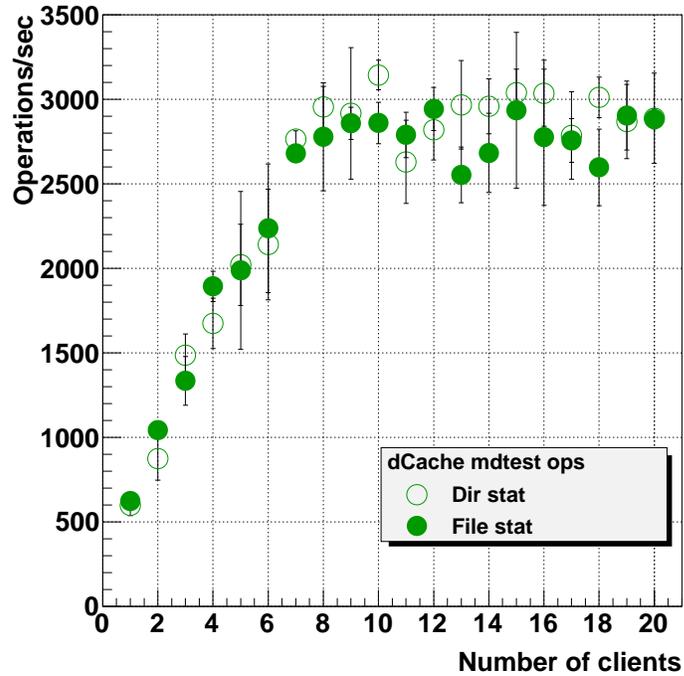


Figure 11: *mdtest* results for *stat* operations on directories and files.

Acknowledgements

The studies and results presented in this document would not have been possible without the generous help of many individuals. We particularly wish to thank Dmitry Litvintsev, Yujun Wu, and Terry Jones of the Data Movement and Storage (DMS) Department for setting up the test dCache system described in Section 2 and for their help and assistance throughout the course of our investigations. We also wish to thank Stan Naymola and Gene Oleynik of the DMS Department for their support. Finally we thank Steven Timm and Farooq Lowe of the Fermigrid department for their help in setting up Scientific Linux Fermi 6 Virtual Machines on Fermicloud and getting them to work.

A. Appendix

	head (dmsdca08)	pool-1 (dmsdca09)	pool-2 (dmsdca10)
CPU	Intel Xeon X3360 @ 2.83 GHz	Intel Xeon E5430 @ 2.66 GHz	Intel Xeon X3360 @ 2.83 GHz
BogoMips	5652.60	5333.33	5652.69
Memory	8 GB	16 GB	8 GB
Motherboard	Supermicro X7SB4/E (quad core)	SuperMicro X7DB8 (dual quad core)	Supermicro X7SB4/E (quad core)
Fibre-channel	QLogic Corp. ISP2422-based 4Gb Fibre Channel to PCI-X HBA		

Table 1: Hardware configuration of dCache nodes

	Fermicloud virtual machine properties
CPU	QEMU Virtual CPU version 0.9.1 @ 2.66GHz
BogoMips	5320.13
Memory	2 GB

Table 2: “Hardware” configuration of virtual machine clients in the Fermicloud cluster.

References

- [1] The dCache Project, <http://www.dcache.org>.
- [2] BlueArc Corporation (now part of Hitachi Data Systems), <http://www.bluearc.com>.
- [3] S. Shepler, M. Eisler, and D. Noveck. *Network File System (NFS) Version 4 Minor Version 1 Protocol*, Internet Engineering Task Force, RFC 5661 (2010); <http://www.ietf.org/rfc/rfc5661.txt>.
- [4] IOzone Filesystem Benchmark, <http://www.iozone.org>.
- [5] D. Chatterton *et al.*, *Performance Co-Pilot for IA-64 Linux User's and Administrator's Guide, Release 2.3*, Silicon Graphics Inc., Mountain View, CA (2002); Performance Co-Pilot, <http://oss.sgi.com/projects/pcp>.
- [6] W. E. Loewe *et al.*, LLNL's Parallel I/O Testing Tools and Techniques for ASC Parallel File Systems, UCRL-CONF-203489 (2004); Mdstest, https://computing.llnl.gov/?set=code&page=sio_downloads.