Profiling Grid Data Transfer Protocols and Servers

George Kola, Tevfik Kosar, and Miron Livny

Computer Sciences Department, University of Wisconsin-Madison
1210 West Dayton Street, Madison WI 53706
{kola,kosart,miron}@cs.wisc.edu

Abstract. The trend of data intensive grid applications has brought grid storage protocols and servers into focus. The objective of this study is to gain an understanding of how time is spent in the storage protocols and servers. The storage protocols have a variety of tuning parameters. Some parameters improve single client performance at the expense of increased server load, thereby limiting the number of served clients. What ultimately matters is the throughput of the whole system. Some parameters increase the flexibility or security of the system at some expense. The objective of this study is to make such trade-offs clear and enable easy full system optimization.

1 Introduction

The increasing trend towards data intensive grid applications [1] [2] [3] has brought grid data transfer protocols and storage servers into focus. We have done a full system profile of GridFTP [4] and NeST [5], two widely used data access and storage server and detailed how time is spent in each server.

Our profiling details server side CPU characteristics and shows the effects of concurrency level, number of parallel streams and protocol parameters like block-size on server load and transfer rate. This makes the trade-off between single client performance and server load clear. We also explain why certain parallelism level lowers the server load while increasing the transfer rate.

A good understanding of this helps computer architects to add processor features and operating system designers to optimize the operating system. It enables middleware and applications developers to optimize their software and helps grid deployers to choose appropriate machine configuration for their applications.

2 Methodology

We wanted to understand how time is spent in data access protocols and storage servers. We decided to study GridFTP and NeST, two widely used grid data access and storage servers. NeST server is interesting because it supports space reservation and a variety of interfaces: native chirp [5], NFS [6] and GridFTP.

Our desire to perform a full-system characterization including the path through the kernel while keeping the system perturbations minimal narrowed our choice of profiler to Oprofile [7], a Linux system-wide profiler based on Digital Continuous Profiling Infrastructure [8]. Oprofile uses the hardware performance counters on the Pentium family of processors.

For profiling, we setup two server machines: a moderate server, 1660 MHz Athlon XP CPU with 512 MB RAM, and a powerful server, dual Pentium 4 2.4 GHz CPU with 1 GB RAM. Both servers used Linux kernel 2.4.20. The moderate server had 100 Mbps connectivity while the powerful one had 1000 Mbps connectivity. We used three client machines, two of them were in local area and 1 was in wide area. The local area clients were dual 2.8 GHz Xeons and had 100 Mbps connectivity and were chosen randomly from a pool of 50 machines and the wide area client was quad 2 GHz Xeon with 100 Mbps connectivity. The powerful machines ensured that the clients were not the bottleneck and brought out the server characteristics.

We got full system profiles for both of GridFTP 2.4.3 and NeST servers using clients in the local area. For the extended study of GridFTP performance, we used clients both in local area and wide area.

Since we used real wide-area transfers we did not have any control over the loss rate. We did not trace it during the experiment because we felt such a packet trace collection at end hosts would interfere with our experiment. But we did periodic network traces and found that wide-area losses were negligible (less than 0.5%) at 100 Mbps. We have a 655 Mbps wide-area ATM connectivity and we found that the packet losses started showing up above 250 Mbps.

We tried out some commonly used options like parallel streams and concurrent number of file transfers in GridFTP and found the effect on server load.

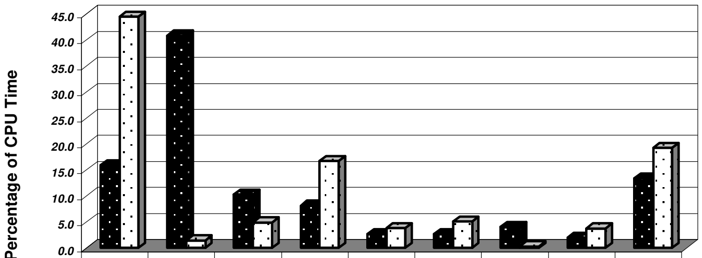
3 Full System Characterization

We studied how the time is spent on the server side and present the results in this section. This characterization details the fraction of time spent in the different system parts including the kernel. This is significant for data servers because most of the time is spent in the kernel and plain user-level server profile is not sufficient.

3.1 GridFTP

Figure 1 shows the GridFTP server CPU characteristic when a single local area client reads/writes a set of 100 MB files. The read and write clients achieved a transfer rate of 6.45 MBPS and 7.83 MBPS respectively.

In terms of server CPU load, reads from the server are more expensive than writes to the server. The extra cost is spent in interrupt handling and in the ethernet driver. The machine has an Intel Ether Express Pro 100 network interface card(NIC). We found that interrupt coalescing lowered the interrupt cost during write to server. The NIC transfers the received packets via DMA to main



	Idle	Ethernet Driver	Interrupt Handling	Libc	Globus	Oprofile	IDE	File I/O	Rest of Kernel
Read From GridFTP	15.9	40.9	10.3	8.1	2.7	2.7	4.0	2.0	13.4
Write To GridFTP	44.5	1.5	4.9	16.8	3.8	5.1	0.3	3.8	19.3

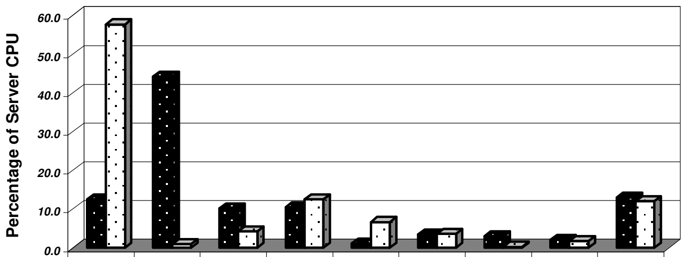
Fig. 1. GridFTP Server CPU Characteristics.

memory resulting in low CPU cost for writes to server. CPU is used to transfer output packets to the NIC resulting in high cost of read from the server. NIC with capability to DMA the output packets along with a driver capable of using that feature would reduce server read load considerably.

In the Libc, 65% of the time is spent in the getc function. The IDE disk has a greater overhead on reads compared to writes. Tuning the disk elevator algorithm may help here. The file I/O part includes the time spent in the filesystem. It is higher for writes because of the need for block allocation during writes. The rest of the kernel time is spent mostly for TCP/IP, packet scheduling, memory-copy, kmalloc and kfree.

3.2 NeST

Figure 2 shows the NeST server CPU profile when a single local area client reads/writes a set of 100 MB files using the chirp protocol. The read and write clients achieved a transfer rate of 7.49 MBPS and 5.5 MBPS respectively.



	Idle	Ethernet Driver	Interrupt Handling	Libc	NeST	Oprofile	IDE	File I/O	Rest of Kernel
Read From NeST	12.5	44.2	10.2	10.4	1.1	3.5	3.0	2.1	12.9
Write To NeST	57.7	1.0	4.3	12.6	6.7	3.7	0.3	1.7	12.0

Fig. 2. NeST Server CPU Characteristics.

NeST server has a 16% higher read transfer rate and a 30% lower write transfer rate compared to GridFTP server. The lower performance of writes to NeST server is because of the space reservation feature called 'Lots'. Before each write, NeST server verifies that the client has not exceeded the storage allocation, and at the end of write, it updates this meta-data persistently. This causes the slowdown. NeST allows turning off 'Lots' and doing that makes the write performance close to that of GridFTP server. This shows that space reservation while being a useful feature comes with a certain overhead.

4 Effect of Protocol Parameters

GridFTP allows us to use different block-sizes and multiple parallel streams. Further, clients can concurrently transfer multiple files to/from the server. We studied the effect of the above parameters and concurrency on transfer rate and CPU utilization.

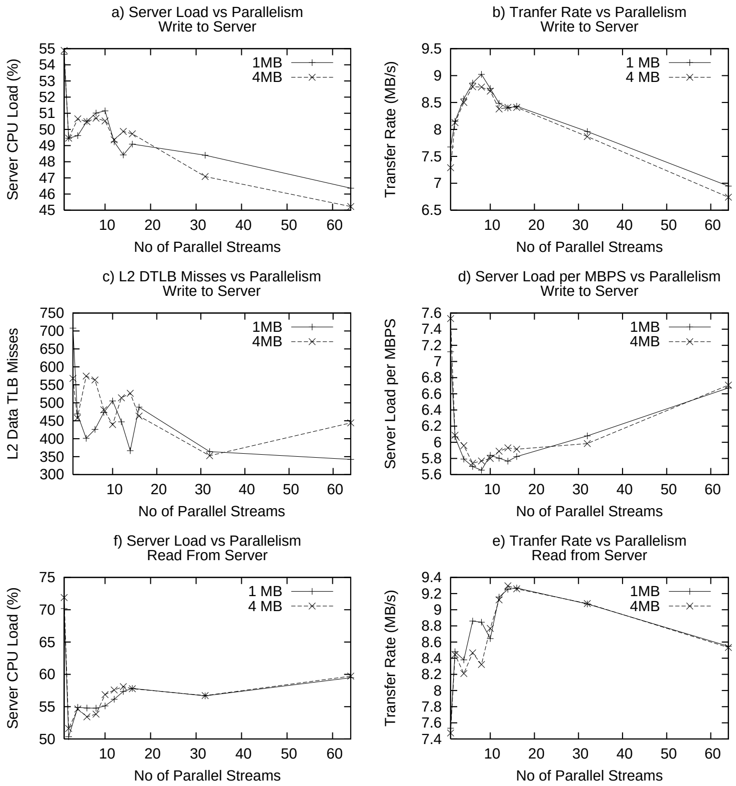


Fig. 3. The effect of block size and the number of parallel streams on GridFTP server load, transfer rate, and TLB misses.

The effect of using different block-sizes and parallelism while writing to the moderate GridFTP server is shown in Fig. 3. Interestingly, the server load drops and the transfer rate increases as we move from one stream to two streams. We repeated the experiment 20 times and got the same results. We analyzed further and decided to look at the Translation Look-Aside Buffer(TLB) misses. TLB is a cache that speeds up translating virtual addresses to physical addresses in the processor. As seen in Fig. 3c, the L2 Data TLB misses drops as the parallelism is increased from one to two. The drop in L2 DTLB misses explains the simultaneous decrease in server load and increase in transfer rate.

We went a step further and tried to find out what was causing the reduction in L2 DTLB misses and found that the Pentium processor family supports a large page size of 4 MB in addition to the normal page size of 4 KB. For data servers, using the large pages would be greatly beneficial. Unfortunately, the Linux kernel at present does not allow application to request such jumbo pages, but internally the kernel can use these large pages. We found that the internal kernel usage of jumbo 4 MB pages during use of parallel streams causes the drop in TLB misses. We also found that using a block size of 4 MB did not make the kernel use the jumbo page internally.

We tried the experiment with different machines and found that they had a different parallelism TLB miss graph. The variance in TLB misses was quite small till 10 parallel streams and starts rising after wards. Another interesting thing we found was that the TLB miss graph of a machine at different times was similar. At present, it appears that the Linux kernel usage of large pages internally depends mostly on the machine configuration. This requires a more thorough analysis.

Figure 3d shows the server load per MBPS transfer rate. Data movers may want to lower server CPU load per unit transfer rate and this graphs shows how they can use parallelism to achieve this.

The effect of block size when reading from the server is shown in Fig. 3e and 3f. We find that the optimal parallelism for reading from the server is different from that used to write to it.

We have studied the effects of different concurrency and parallelism levels on the transfer rate and CPU utilization. This study was done using the powerful server and the effect on write to server is shown in Fig. 4.

We observed that increasing the number of concurrent files being transferred results in a higher transfer rate compared to increasing the number of parallel streams (Fig. 4a and 4b). This is the result of multiple transfers being able to saturate the bandwidth better than single transfer with multiple streams. In wide area, both increasing the level of concurrency and parallelism improve the performance considerably (Fig. 4b). Whereas in the local area both have very little positive impact on the performance, and even cause a decrease in the transfer rate for slightly high concurrency and parallelism levels (Fig. 4a). As the file size increases, the impact of parallelism level on transfer rate increases as well (Fig. 4c). This study shows that increased parallelism or concurrency level does not necessarily result in better performance, but depends on many

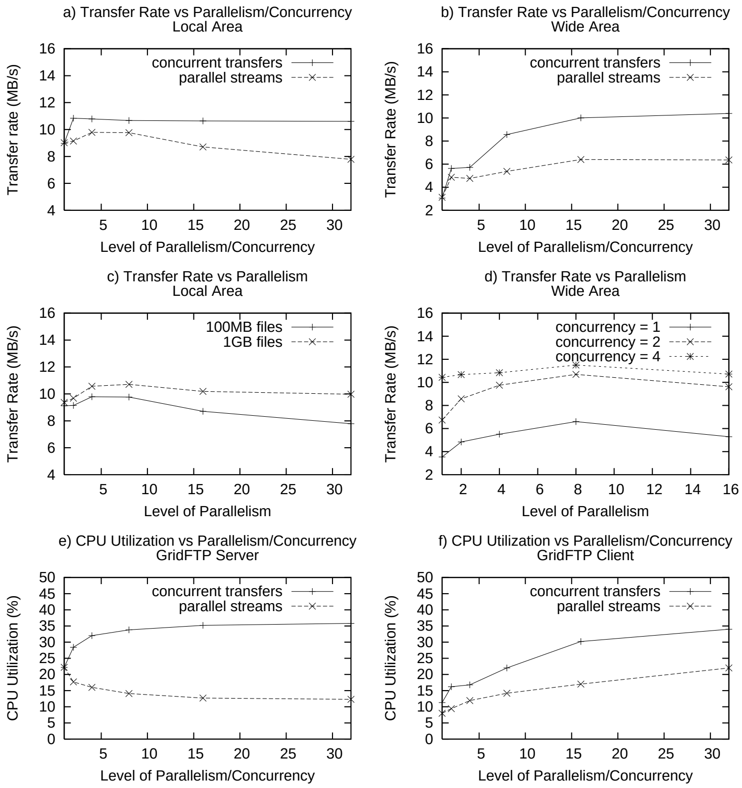


Fig. 4. The effect of concurrency and parallelism levels on the transfer rate and CPU utilization during write operations to the GridFTP server.

parameters. The users should select the correct parallelism or concurrency level specific to their settings.

Increasing the concurrency level also results in a higher load on the server (Fig. 4e), whereas increasing the number of parallel streams decreases server CPU utilization. On the client side, both increasing the concurrency and parallelism levels cause an increase in the CPU utilization of the client machine (Fig. 4f). We believe that using a combination of concurrency and parallelism can result in higher performance than using parallelism only or concurrency only (Fig. 4d). It can also help in achieving the optimum transfer rate by causing lower load to the server.

5 Related Work

CPU, memory and I/O characteristics of commercial and scientific workloads have been well studied [9] [10] [11] [12]. However, grid storage servers and data transfer protocols have not been profiled and characterized in detail.

Networked Application Logger (NetLogger) [13] toolkit enables distributed applications to precisely log critical events and thereby helps to identify system bottlenecks. It requires application instrumentation, which is difficult for complex and binary-only applications. It cannot be used to log frequent short events and kernel operations. The instrumentation may change the behavior of the program. Since, we wanted to perform a full system characterization that shows the time spent in kernel, we could not use NetLogger.

Vazhkudai et. al. [14] instrumented GridFTP [4] to log performance information for every file transfer and used it to predict the behavior of future transfers. They found that disk I/O takes up to 30% of the total transfer time and using disk I/O data improves end-to-end grid data transfer time prediction accuracy by up to 4% [15]. Our profiling gives a more complete picture of system performance and we believe that this information can be used to make more accurate predictions.

Silberstein et. al. [16] analyzed the effect of file sizes on the performance of local and wide-area GridFTP transfers and found that files sizes should be at least 10 MB for slow wide-area connections and 20 MB for fast local-area connection in order to achieve 90% of optimal performance, and small files do not benefit from multiple streams because of increased overhead of managing the streams. Our profiling work would help people find values for other parameters to achieve close to optimal performance.

6 Conclusion

In this work, we have done a full system profile of GridFTP and NeST, two widely used data access and storage servers and detailed how time is spent in these servers.

We have characterized the effect of concurrency and GridFTP protocol parameters block size and parallelism on data transfer rate and server CPU utilization. We have made clear the trade-off between single client performance and server load and shown how client performance can be increased and server load decreased at the same time and explained the reason behind this. This allows users to configure and optimize their systems for better end-to-end transfer performance and higher throughput.

We are planning to analyze our profiling data further in order to find better correlations between different parameters. This will help us to provide more useful information to users helping them to perform more sophisticated configuration and optimization.

Finally, we want to make our profiling dynamic and want to design and implement a feedback mechanism between our profiling model and higher level data movement schedulers like Stork [17]. This would allow systems to dynamically increase their knowledge, and enable schedulers to make intelligent scheduling decisions based on analysis of dynamically collected profiler data.

References

1. Sagal, B.: Grid Computing: The European DataGrid Project. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, Lyon, France (2000)
2. CMS: The Compact Muon Solenoid Project. http://cmsinfo. cern.ch/ (2004)
3. LIGO: Laser Interferometer Gravitational Wave Observatory. http://www.ligo.caltech.edu/ (2003)
4. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S.: Secure, efficient data transport and replica management for high-performance data-intensive computing. In: IEEE Mass Storage Conference, San Diego, CA (2001)
5. Bent, J., Venkataramani, V., LeRoy, N., Roy, A., Stanley, J., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., Livny, M.: Flexibility, manageability, and performance in a Grid storage appliance. In: Proceedings of the Eleventh IEEE Symposium on High Performance Distributed Computing (HPDC11), Edinburgh, Scotland (2002)
6. Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B.: Design and implementation of the Sun Network Filesystem. In: Proc. Summer 1985 USENIX Conf., Portland OR (USA) (1985) 119–130
7. Oprofile: A System Wide Profiler for Linux. http://oprofile.sourceforge.net (2003)
8. Anderson, J.M., Berc, L.M., Deanand, J., Ghemawat, S., Henzinger, M.R., Leung, S.A., Sites, R.L., Vandevoorde, M.T., Waldspurgerand, C.A., Weihl, W.E.: Continuous profiling: Where have all the cycles gone? In: Proceedings of the sixteenth ACM symposium on Operating systems principles. (1997)
9. Barroso, L., Gharachorloo, K., Bugnion, E.: Memory system characterization of commercial workloads. In: Proceedings of the International Symposium on Computer Architecture. (1998)
10. Lee, D., Crowley, P., Bear, J., Anderson, T., Bershad, B.: Execution characteristics of desktop applications on Windows NT. In: Proceedings of the 25th International Symposium on Computer Architecture. (1998)
11. Crandall, P., Aydt, R., Chien, A., Reed, D.: Input/output characteristics of scalable parallel applications. In: Proceedings of the IEEE/ACM Conference of Supercomputing, San Diego, CA (1995)
12. Kuo, S., Winslett, M., Cho, Y., Lee, J., Y.Chen: Efficient input and output for scientific simulations. In: Proceedings of I/O in PArallel and Distributed Systems. (1999)
13. Tierney, B., Gunter, D.: NetLogger: A toolkit for distributed system performance tuning and debugging. Technical Report LBNL-51276, LBNL (2002)
14. Vazhkudai, S., Schopf, J.M., Foster, I.: Predicting the performance of wide area data transfers. In: In Proceedings of the 16th Int. Parallel and Distributed Processing Symposium. (2002)
15. Vazhkudai, S., Schopf, J.M.: Using disk throughput data in predictions of end-to-end grid data transfers. In: Proceedings of the third International Workshop on Grid Computing, Baltimore, MD (2002)
16. Silberstein, M., Factor, M., Lorenz, D.: Dynamo - directory, net archiver and mover. In: Proceedings of the third International Workshop on Grid Computing, Baltimore, MD (2002)
17. Kosar, T., Livny, M.: Stork: Making Data Placement a First Class Citizen in the Grid. In: Proceedings of the 24th Int. Conference on Distributed Computing Systems, Tokyo, Japan (2004)