Performance Evaluation of Distributed Data Access Optimization Approaches: Experiments versus Simulations

Vinicius de Freitas Reis [1] **and Rodrigo Fernandes de Mello** [1]

[1]Department of Computer Science – Institute of Mathematics and Computer Sciences
University of São Paulo – São Carlos – SP – Brazil

vini.reis@gmail.com, mello@icmc.usp.br

Abstract. *This work was motivated by the gap in between simulation and experimental approaches when evaluating distributed data access operations. The performance of file replication techniques is evaluated and compared under both approaches to investigate whether simulation results are observable in real-world scenarios. In that sense, this paper presents a modular support to implement data access optimization policies in the Gfarm distributed file system and compare it to OptorSim, a broadly used Data Grid simulator. We have concluded that, in some situations, even qualitative results differ significantly in both approaches, raising important considerations when evaluating simulation results and designing real-world scenarios.*

1. Introduction

Until the last decade, three paradigms were explored in science: empirical, based solely on experimentation; theoretical, which attempts to explain facts by equations; and computational, which addresses complex analytical models by considering simulations. However, recently there has been a huge increase in the data produced by instruments or generated by simulations, drastically changing techniques, technologies and the whole pipeline of scientific exploration. This change, for some authors, can further characterize a new scientific paradigm [Hey et al. 2009].

Examples of applications under this new paradigm are the SKA (Square Kilometer Array) Project, which may produce data at a rate of 4.1 Pbits/s [Dewdney et al. 2009], Pan-STARSS (Panoramic Survey Telescope & Rapid Response System), which captures terabytes of raw data a night [PAN-STARRS 2010] and the LHC (Large Hadron Collider), which is estimated to require in between 50 and 100 PB of storage per year [Hey et al. 2009]. The large volume of data captured by all those projects must be preprocessed before storage, simplifying further analysis.

Grid computing [Foster et al. 2001] was proposed as the computational infrastructure to meet these requirements by coordinating the sharing of heterogeneous resources (such as telescopes and other complex instruments) among organizations at geographically distributed locations. In the same way, Chervenak et al. [Chervenak et al. 1999] proposed the concept of Data Grids, a specialization of Grids focused on providing reliable and robust large-scale data sharing.

Several application domains, such as high-energy physics, bioinformatics and climate analysis, require such type of infrastructure. Those domains face problems mainly related to storage size, data throughput, application performance as well as the costs involved in all that scenario. Another common problem is that data is heavily generated at

only one site and accessed by remote computers. Thus, data files must be stored in a way to reduce access costs, such as latency, consequently improving application performance (not only the application generating data, but also others which access the information). Besides, data must not be concentrated, as hotspots and single points of failure would be generated.

File replication is the approach commonly considered to address many of those problems. This approach creates multiple file copies at different grid sites in order to: 1) increase file availability, making a file accessible even if some of the replicas are offline or a grid site fails; 2) balance the data servers workload, once file accesses can be distributed among replicas; and 3) decrease file access costs, as every process can use the nearest replica (considering both latency and bandwidth). This scenario brings new problems, as the maintenance of consistency, which creates a tradeoff in between the data distribution benefits (such as performance, availability, etc) and the access of updated data.

Ragananthan & Foster [Ranganathan and Foster 2001] evaluated how file replication affects Grid performance using their own developed simulator. Three access patterns were considered, against five dynamic replication strategies. Results showed reductions in the network bandwidth consumption as well as in the file access time. In a similar direction, Bell et al. [Bell et al. 2003a] proposed the OptorSim simulator as a tool to study different replication strategies on Data Grids. The modular approach of OptorSim made it broadly considered [Bell et al. 2003b, Elghirani et al. 2009, Rahman et al. 2008] and also employed in the CERN's EU DataGrid Project.

From a more practical point of view, Douceur & Wattenhofer [Douceur and Wattenhofer 2001] evaluated file replica approaches in the Farsite distributed file system in order to increase file availability. A distributed hill climbing approach was used, and networks with more than 50.000 nodes were simulated. Tatebe et al. [Tatebe et al. 2003] studied the time spent on file replication, considering parallel transfers. Experiments were conducted on networks connecting USA and Japan. Abdullah et al. [Abdullah et al. 2008] considered a peer-to-peer Data Grid model and discussed nearest neighbor-based distributed replication strategies. A simulator was implemented using the PARSEC simulation language and experiments evaluated peer-to-peer domain measurements, such as success rate and flood response time. The results confirmed significant benefits.

By analyzing the previous works, we observe there is a gap in between simulation and experimental approaches. Simulations are very useful in order to point out relative tendencies of replication and consistency strategies, however, they neither confirm the actual amount of resources used nor the final performance improvements. On the other hand, experimental approaches are costly and hard to implement, demanding considerable time. Another important concern is how to ensure that simulation results are indeed observable in real-world scenarios. In that sense, this paper proposes a modular support to implement data access optimization policies in the Gfarm distributed file system [Tatebe et al. 2002]. Gfarm was selected due to its stability and robustness, verified after a trial with several distributed file systems [1].

[1]Ceph [Weil et al. 2006], Starfish [Starfish 2009], Gfarm [Tatebe et al. 2002], HDFS [Hadoop 2010], Cloudstore [Cloudstore 2010] and PVFS [Carns et al. 2000] were evaluated, as detailed in Section 3

The support provides a well-defined interface, allowing researchers and developers to design and test new optimization approaches. Besides this proposal, we also implemented and evaluated the performance of the most commonly considered data access optimization approaches, which are available in the OptorSim simulator. OptorSim was chosen for comparison as it is one of the most used data grid simulators [Bell et al. 2003b, Elghirani et al. 2009, Rahman et al. 2008]. By analyzing the results, we confirm the tendency of OptorSim results. However, the percentages of improvements are very distinct for some situations, which raises important considerations when evaluating simulation results and the design of real-world scenarios.

This paper is organized as follows: Section 2 presents some related work; Section 3 describes the architecture of the proposed support as well as implementation details; Section 4 explains how experiments were prepared, presenting results of the execution of those experiments on a real environment and on OptorSim. Finally, conclusions and references are presented.

2. Related Work

Related work on data access optimization methods can be divided into two basic approaches: simulation and experimental ones. While simulations are very popular and useful to relatively compare different optimization strategies and point out tendencies, inaccurate results may limit real-world conclusions. On the other hand, the inherent difficulty in building environments, designing and implementing systems has limited experiments in real scenarios. Some of the main works on data access optimization are presented next. As the reader may observe, they are strongly based on simulation methods.

Ranganathan & Foster [Ranganathan and Foster 2001] studied the effect of many replication strategies by considering different file access patterns. Results, based on their own simulator, analyzed both the bandwidth consumption and the time consumed to retrieve remote files. Local files were supposed to have zero response time, and file writing was not allowed, avoiding consistency problems. Three file access patterns were considered: one purely random, one with small temporal locality and one with both temporal and geographical locality. Five different replication strategies were studied, and a replacement strategy hybrid in between file popularity and access time was used. No details about network topology and size were given. Results compared replication strategies among themselves, but no comparison was made to an environment without replication. It was possible to observe an approximately 50% reduction in response time for some scenarios, as well as an increase in those times in other cases. Bandwidth consumption and response time were analyzed.

Elghirani et al. [Elghirani et al. 2009] explored the synergy in between process scheduling and data management, aiming to increase the overall application performance. They also proposed a framework to replicate and estimate the proper file copy placement. The framework provides file distribution algorithms based on Tabu Search [Glover 1986], Genetic Algorithms [Holland 1975] and Ant Colony Optimization [Dorigo and Di Caro 1999]. Those techniques consider information such as access patterns, dataset popularity, network latency and bandwidth. Simulations were conducted using the proposed framework in conjunction with OptorSim [OptorSim 2009] and the European DataGrid Testbed 1 was evaluated (this is a typical OptorSim testbed)

[Bell et al. 2002]. Tests considered a single file access pattern, with varying number of jobs, and analyzed the job execution time.

Rahman et al. [Rahman et al. 2008] modeled the dynamic data replication problem by using operations research. The replica distribution problem is modeled as the resolution of a p-center or p-median problem [Hakami 1963], or a combination of both, which is achieved by a multi-objective optimization algorithm. The solution is approximated by using Lagrangian relaxation [Fisher 2004]. The OptorSim simulator was used to validate the replication strategies, using the European DataGrid Testbed 1. This work did not consider the time consumed in operations; it only evaluates the approach by counting the number of local and remote accesses. Moreover, it does not compare the proposed approach to others available in OptorSim.

Sato et al. [Sato et al. 2008] proposed a data replication strategy considering application dynamics. The replication problem was modeled as a combinatorial optimization problem, which aims to minimize the replication transfer time. The authors approach the problem using an integer linear programming solver [GLPK 2009], fact that created noticeable delay according to them. A prototype was implemented in the Gfarm distributed file system [Tatebe et al. 2002], however experiments in real environments were limited only to network throughput estimates. File size distribution was based on the files used by BLAST biological sequence alignment application [Altschul et al. 1990]. The main evaluations of the proposed approach were drawn from simulations, however no details were given about how simulations were conducted or which tools were used, which does not allow reproducibility.

Furthermore, the objective and metrics vary among the presented initiatives. Sato et al. [Sato et al. 2008] and Elghirani et al. [Elghirani et al. 2009] consider the total application execution time as performance metric. On the other hand, Rahman et al. [Rahman et al. 2008] evaluate the data transfer time and the number of local and remote accesses. Ranganathan & Foster [Ranganathan and Foster 2001] analyze bandwidth consumption and response time. Sato et al. [Sato et al. 2008] also study the replication workload, by evaluating the free storage space. Ranganathan & Foster [Ranganathan and Foster 2001] discuss some algorithms storage usage, but do not present any data. Another issue is that the algorithms proposed, except for Ranganathan & Foster [Ranganathan and Foster 2001], are not distributed, requiring a central entity to coordinate all operations. All of them also assume the existence of a graph indicating the network topology, latencies and bandwidths with accurate measurements, an assumption that is very restrictive considering dynamic environments.

As previously presented, we have observed a vast list of relevant works on data access approaches, however most of them are based on simulation approaches. That was the main motivation for this work, which considered an experimental strategy and compared it to one of the main simulators currently proposed, called OptorSim [OptorSim 2009]. This comparison allowed confirming sensible differences in simulation and real-world scenarios, influencing decisions in practical situations.

3. Proposed Support

This work was motivated by the fact that most data access optimization studies are conducted based on simulations, which can produce inaccurate results when compared to

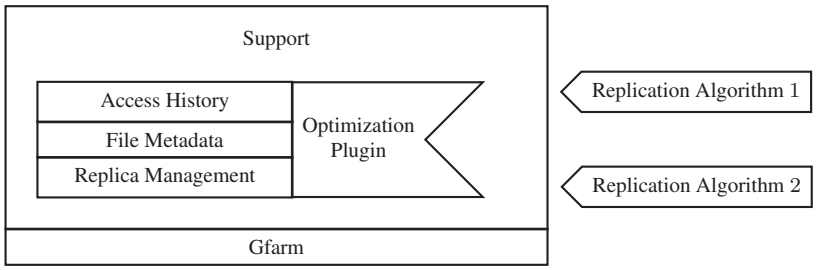


Figure 1. Optimization support overview. The blocks inside the support show examples of services provided to optimization plugins.

real situations. It proposes a support to design and evaluate distributed data access optimization techniques in real-world scenarios. The support was developed over the Gfarm [Tatebe et al. 2002] distributed file system, and allows optimization strategies to be plugged, easily developed and tested. Figure 1 shows the basic architecture of the system: the support, built over the Gfarm API, offers Replica Management services, File Metadata and Access History retrieval for optimization plugins, which can be tested and interchanged.

Before designing it, a distributed file system was selected to take advantage of the support services. In such a situation, we defined the following list of criteria: 1) the file system should be licensed under an open source model, making it available for other researchers, developers and users, and have the source code available for study; 2) the file system should be stable, allowing for reproducible and reliable experiments; 3) the file system should support both read and write operations, enabling its use for scientific applications; 4) the file system should not have any single point of failure.

Since then, we defined a set of file systems to be evaluated: Ceph [Weil et al. 2006], Starfish [Starfish 2009], Gfarm [Tatebe et al. 2002], Hadoop's HDFS [Hadoop 2010], Cloudstore [Cloudstore 2010] and PVFS [Carns et al. 2000]. NFS was designed for local networks, therefore it was not considered in this set. Ceph was the first file system to be tested. It satisfied some of the criteria: it supports read-and-write operations, and has distributed storage and distributed metadata. However, even simple file sharing tests in between two machines revealed the existence of bugs. Emails exchanged with Ceph's author confirmed that the file system was not ready to be used in production environments. The second file system tested was Cloudstore, which also presented stability problems. The third system tested, Gfarm, was selected to support this work. Although it did not fully meet all criteria (it presents a single point of failure in the metadata server), it was the best available. Regarding the other three systems, Starfish could not be used because of patent problems, PVFS replication support required high-cost solutions, such as a SAN (Storage Area Network), and HDFS was designed in write-once approach.

The Gfarm distributed file system is open source software distributed under GNU/GPL [GPL 2009] and part of the Asia Pacific Grid project (ApGrid) [Tatebe et al. 2002]. It was designed to manage file accesses in dynamic networks consisting of thousands of computers storing large amounts of data. Gfarm provides the clients a single file system image, transparently available to applications via a FUSE (File system in Userspace) module. Gfarm itself is composed of several data servers and a metadata server which is, unfortunately, a single point of failure to the system. Data servers communicate directly with clients to provide data and the metadata server manages the

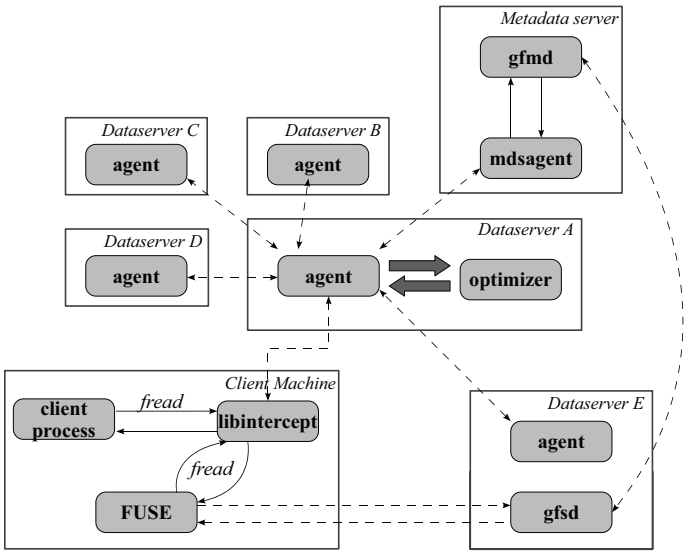


Figure 2. Interprocess communication at the proposed support.

replica catalog (it provides the association among files and data servers), file permissions and other relevant data.

In the next step, the support was designed. We decided to have three main components: one in the clients, one in data servers and one in the metadata server. The client component is responsible for monitoring the application and triggering optimization requests; the data server component runs the optimization techniques and replicates files, and the metadata server optimization component is responsible for removing file replicas. We also had to serialize replica removal requests, ensuring that at least a single file replica would always exist. Due to the distributed nature of the file system, two different data servers may request the removal of a single replica of the same file at the same time.

Figure 2 illustrates the interprocess communication during the optimization. The `agent` process is the optimization process which runs at data servers; `optimizer` is the process which implements the optimization technique; `libintercept.so` is an interception library linked to the client process at runtime; `gfsd` is the Gfarm data server process; `gfmd` is the Gfarm metadata server process; `mdsagent` is the process responsible for receiving `agent` requests to remove file replicas.

The client component was implemented by intercepting client processes, allowing the support to capture process data beyond file accesses, such as system calls. This interception is performed using both `Dlsym` library and the dynamic linker. The Linux environment variable `LD_PRELOAD` allows the process functions from the C standard library to be overwritten by the interception library. After executing the desired code, the original functions from GNU Libc are called using the `Dlsym` library. This technique has a small overhead (around 1 and 2 percent considering the process total execution time [Ishii 2010]).

When a file has been frequently accessed, an optimization request is sent to a data server. To decide whether or not to trigger this request, an EWMA (Exponentially Weighted Moving Average) of access times is used, as averages closer to the current time indicate a burst of file accesses. For example, when the client issue five file reads in less than a second, this average will be very close to the current time, triggering the

request. On the other hand, if only a single request occurs, it will not change the average significantly and the optimization will not be triggered.

When the difference in between this average and the current time is below some threshold, the optimization request is sent to the nearest data server in terms of RTT (Round-trip time). This is possible because clients keep an updated list of online data servers with measurements of their RTT and free disk space. The request contains information such as every I/O operation parameter and timestamp, available file replicas and file sizes.

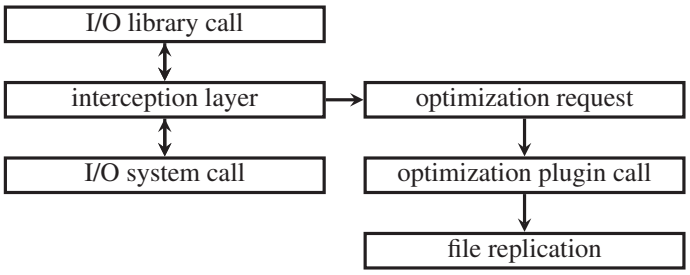


Figure 3. Main steps involved in the optimization process.

As the support needs to be modular, each optimization technique is a separate program, which receives information from this request through its standard input and writes the resulting actions back to its standard output (file replicas can be created or removed). This makes the development of the optimization technique easier, as any programming language can be used.

4. Experiments

4.1. Environment Configuration

The next step in the project development was to evaluate the proposed interface and compare data access optimization strategies under real-world scenarios. An approach based on virtual machines was chosen, as it allowed emulating different network topologies and system scales. The developed software could be used, with no change, to conduct experiments in real-world grid computing environments. One of the goals of these experiments was to evaluate how simulation (using OptorSim [Bell et al. 2002]) results are related to the ones obtained under those virtual environments.

The support prototype was built using VirtualBox [VirtualBox 2010] virtual machines, but this approach soon revealed too expensive even to create medium-sized networks, as each virtual machine used a large amount of resources. The solution found was to switch to OpenVZ [OpenVZ 2010] virtual machines, which in fact proved to be much more scalable. VirtualBox uses an approach called hardware-level virtualization [Sugerman et al. 2001] (or full virtualization), in which each virtual machine has its own simulated hardware devices with different instances of operating systems running. On the other hand, OpenVZ uses an approach called operating system-level virtualization [Soltesz et al. 2007], in which a single kernel is shared among all virtual machines (now called containers) and all processes run natively. This approach is capable of providing isolation among containers and also presents less overhead than other virtualization techniques, such as complete virtualization (VirtualBox, VMWare) and paravirtualization (Xen) [Chaudhary et al. 2008].

After creating the containers, it was necessary to configure the network among them. Each connection in between the machines was defined as a kernel bridge containing two virtual network interfaces. The kernel module `netem` (Network Emulation) provided control over the connection latency, and the module `htb` (Hierarquical Token Bucket) provided control over the connection bandwidth. A script was used to automatically create the containers, the network topology (with the given latencies and bandwidths) and the routing tables, which were calculated using the Floyd-Warshall algorithm applied to the input graph. Several dynamic routing software, including BIRD [BIRD 2010] and Quagga [Quagga 2010], were tested, but none of them worked out on the virtual network created.

The results of the emulation environments were compared to the ones obtained with OptorSim [Bell et al. 2002]. OptorSim is a broadly used simulator, designed to study the dynamics of data replication strategies in Grid environments. It simulates the Grid as a network of sites, each one having its own set of SEs (Storage Elements) and CEs (Computing Elements). Each site also has a Replica Manager service, which takes replica optimization decisions, and a Global Resource Broker, which schedules jobs among CEs. An arbitrary network topology can be specified and every job can be configured to use a different subset of files.

Two file replication techniques implemented in OptorSim are considered in this work: Least Frequently Used (LFU) and Least Recently Used (LRU). Those techniques are based on the well-known cache replacement policies. The link in between caching and file replication is the assumption that, when a file is first used, it is always advantageous to create a its local replica considering there is enough space. When there is no disk space available, a replica must be removed in order to create a new one. LFU removes the least frequently used replica in some predefined time window and LRU removes the least recently used replica.

The order in which a job requests files determines its access pattern. Four different access patterns were implemented in OptorSim and also used in this work: Sequential, Unitary Random Walk, Gaussian Random Walk and Flat Random. The first file read is always random, with uniform distribution among files. Let i be the identifier of the file currently read. In the Sequential pattern, next reads follow the sequence $(i + 1, i + 2, i + 3, \ldots)$. In Random Walk patterns, the next file read is chosen using some probability distribution centered at i: in Unitary Walks this distribution is uniform (files $i + 1$ or $i - 1$ can be chosen), and in Gaussian Walks this distribution is normal. Finally, in the Flat Random pattern, every file has the same probability (uniform distribution) of being chosen in all reads.

All simulations and experiments were conducted using the CMS Testbed 2002 network topology available in OptorSim, as shown in Figure 4. This network is based on the CMS (Compact Muon Solenoid) experiment, part of the LHC (Large Hadron Collider) project at CERN (Conseil Européen pour la Recherche Nucléaire), and is composed of 20 data servers and 6 routers distributed among research centers. Such a testbed topology was considered in OptorSim simulations and also emulated in the experimental scenario using OpenVZ. The machine used in the experiments was Intel(R) Core(TM) 2 Quad CPU $Q9550$ at 2.83GHz with 4GB of RAM running CentOS 5.4.

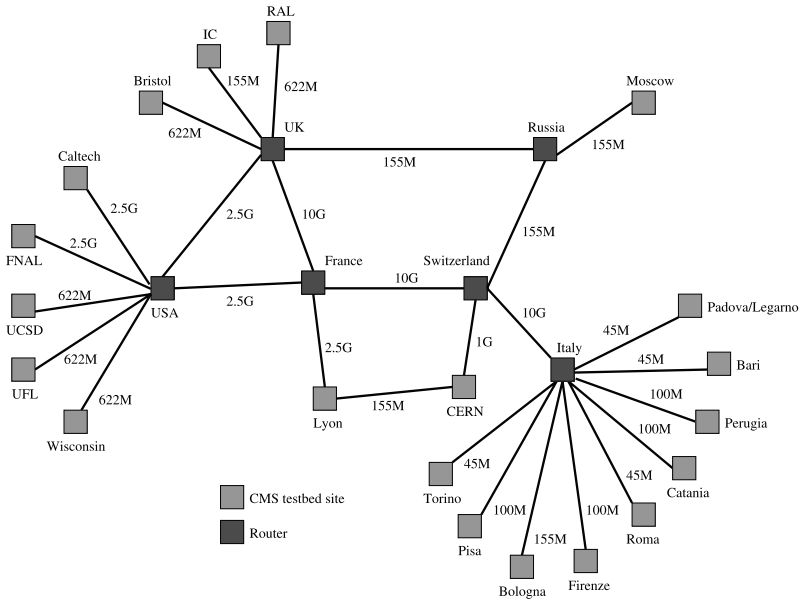


Figure 4. CMS Testbed 2002 network topology [Bell et al. 2003b].

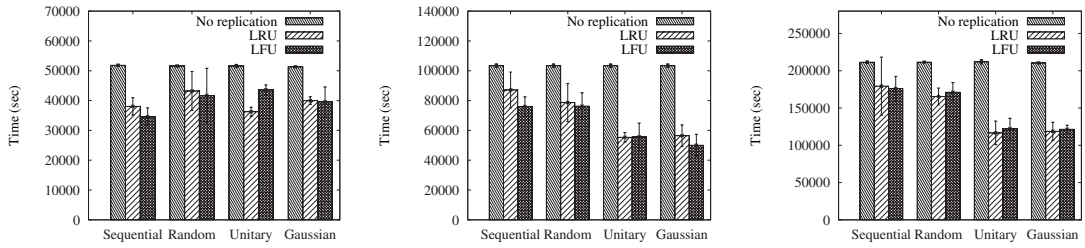
We created 100 files, initially located at FNAL site (Figure 4). All files had the same size, which can also be seen as multiple fixed-size chunks of the same file. The metadata server was located at CERN. The graph used by OptorSim [Bell et al. 2002] did not specify connection latencies, which define the nearby regions for the optimizer. The empirical equation $latency = \lfloor \frac{5000}{x} \rfloor$, where x is the link bandwidth (in Mbps), was used to model those missing latencies. Using such an equation, we had latencies distributed in between 0msec and 110msec.

4.2. Results

Figures 5 and 6 present the experimental (using the emulated environment) and simulation (using OptorSim) results, considering different file access patterns. A single job was run on every grid site, reading 30 whole files using 4KB blocks. Although we considered only one job per site, we can indeed see it as one entity summarizing the behavior of multiple jobs running at the same location. Notice that the plots show the sum of the execution time of all jobs, such that the global system behavior is depicted. Any reduction in this time represents a real performance improvement, although some jobs may run slower due to overhead introduced by the support.

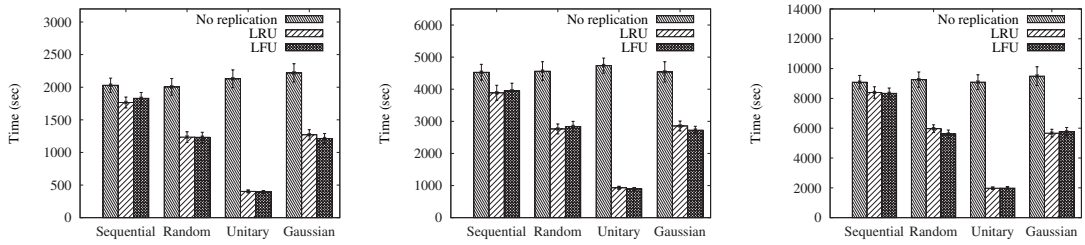
The free disk space on every data server was enough to store 5 file replicas in all scenarios, avoiding all files being replicated to every grid site. Due to the time required, real experiments were repeated 5 times, while every scenario was executed 30 times on OptorSim (simulations were run more times due to their considerably shorter execution time). Error bars show the confidence interval of 95% for the average time necessary to conduct all file access operations under different scenarios. It is important to notice that the confidence intervals found are small enough to be statistically relevant, despite the relatively small number of repeated experiments (5 times).

Notice that the scales in Figures 5 and 6 are quite different, an expected fact as the emulated environment was run on a single physical machine. However, the relative results between the two environments should be equivalent (and they were not).



(a) 128MB files (b) 256MB files (c) 512MB files

Figure 5. Emulated environment results.



(a) 128MB files (b) 256MB files (c) 512MB files

Figure 6. OptorSim simulation results.

Figure 6 clearly shows that, on OptorSim, every replication technique has the same relative results independently of the file size chosen. On the other hand, Figure 5 shows a quite different behavior comparing the 128MB situation to the 256MB and 512MB scenarios. The performance decrease observed at 128MB for the Unitary and Gaussian patterns can be explained by the optimization support overhead: the first file access at every client will trigger an optimization request, which will trigger a latency measurement at the client, causing a significant lag when the job execution time is determined by 128MB files. The time to take this measurement does not lengthen as the file size increases, making it less perceived under other scenarios.

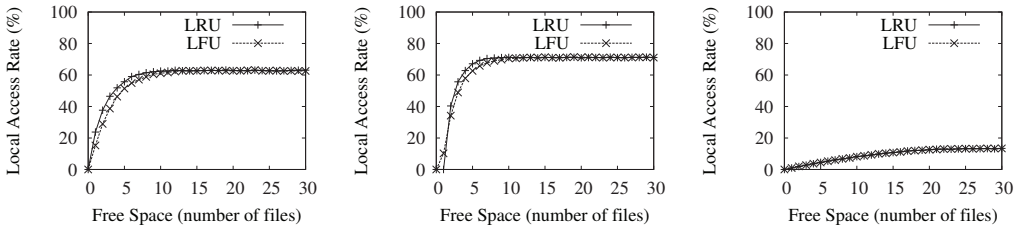
Another important analysis is the speedup of replication techniques (compared to the situation with no replication), as it makes possible to compare the relative results between the two environments. In OptorSim experiments, using 256MB files under the Unitary Random Walk access pattern, the LRU replication offers 5.07 of runtime speedup against only 1.87 observed in the real scenario. On the other hand, comparisons with other access patterns are more consistent. For example, considering the Gaussian Walk access pattern, speedups are 1.59 for OptorSim and 1.83 in the real environment. All those results are summarized in Table 1.

This type of analysis is an important achievement, because it shows that simulations conducted in OptorSim, even in a qualitative manner, may provide very inaccurate results. For example, Sato et al. [Sato et al. 2008] report that their proposed replication technique provided a speedup equal to 200, a very doubtful result even considering a different network topology and file access patterns. Those authors did not give details on the simulation technique used, which makes it impossible to reproduce experiments.

Analyzing the experimental results, we can conclude they are consistent: both Flat Random and Sequential access patterns have compatible performance, which is expected since the probability of a file being read twice is almost zero for Flat Random pattern, and

Table 1. Speedup comparison.

	LRU		LFU	
	OptorSim	Real	OptorSim	Real
Sequential	1.16	1.18	1.15	1.36
Flat Random	1.65	1.31	1.60	1.36
Unitary Walk	5.07	1.87	5.30	1.86
Gaussian Walk	1.59	1.83	1.66	2.06



(a) Gaussian Random Walk (b) Unitary Random Walk (c) Flat Random

Figure 7. Monte Carlo simulation results.

zero for the Sequential one.

A study was also conducted to analyze the behavior of the system as the free space, on every data server, is increased. Considering an environment with only two machines – a server, containing all files, and a client, that issued requests and had some local disk free space to store replicas – a simple Monte Carlo simulation was conducted to calculate the probability of a local file access, given an access pattern and the free space at the client. Each simulation was repeated $3,000$ times, and Figure 7 shows the average of local file access rate for the Unitary Random Walk, Gaussian Random Walk and Flat Random access patterns. The Sequential pattern was not considered because it would always have rates equal to zero.

We observed that after 10 files of free space, the performance of the optimization techniques became stabilized (Figure 7). At this point, no replica removal occurs, therefore the performance of LRU and LFU (and every other file replication technique that creates replicas in the first access and uses some other strategy to remove replicas when the local data server becomes full) is exactly the same.

Another important consideration is that the free space used in previous experiments (5 files) is very close to the optimal performance, meaning that even if the free space were changed in those experiments, no significant performance changes would occur. In fact, considering that the access times are 4 seconds for local files and 230 for remote ones (these were the average times observed in real experiments for 256MB files), we used the local access rate to estimate the optimal performance achieved when free space was unlimited. Results for an unlimited approach are presented by horizontal dashed bars in Figure 8.

It may seem strange that some real results are below the bars, but it can be explained by the fact that a replicated file does reduce the file access cost on all nearby servers (if they read the file), which is an effect not considered on the simulation proposed. However, this type of simulation was very useful because it showed the need for the development of file replication techniques that proactively create replicas and did not rely only on file requests from a single job. Every file replication technique that does not

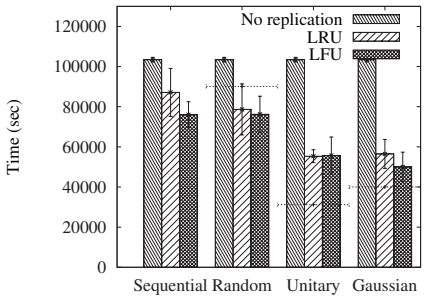


Figure 8. Integrated results. Dashed bars show the expected performance if every server had unlimited disk space.

follow this approach is bounded by the performance limits shown in Figure 8.

5. Conclusions

This work investigates the differences in between experimental and simulation approaches when evaluating distributed data access optimization algorithms. A support for distributed data optimization using the Gfarm distributed file system was implemented and a virtualized environment was used as a testbed for this support. Several experiments were conducted to evaluate some classic algorithms, and these same experiments were reproduced in the OptorSim simulator. A comparative analysis of those results was performed, showing that OptorSim can produce inaccurate results even in relative terms for some scenarios. This result raises important concerns when studying the performance of file replication algorithms, since most works are based on simulations [Ranganathan and Foster 2001, Sato et al. 2008, Bell et al. 2003b, Elghirani et al. 2009, Rahman et al. 2008, Douceur and Wattenhofer 2001, Abdullah et al. 2008].

As future work, we intend to evaluate some real-world applications running in this virtualized environment, which is an important step as the performance improvement is highly dependent on the access pattern considered. We also expect to develop and test a new replica placement algorithm and evaluate its performance in both environments.

6. Acknowledgments

This paper is based upon work supported by FAPESP (São Paulo Research Foundation), Brazil. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of FAPESP.

References

Abdullah, A., Othman, M., Ibrahim, H., Sulaiman, M., and Othman, A. (2008). Decentralized replication strategies for p2p based scientific data grid. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 3, pages 1 –8.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

Bell, W. H., Bell, W. H., Cameron, D. G., Cameron, D. G., Capozza, L., Capozza, L., Millar, A. P., Millar, A. P., Stockinger, K., Stockinger, K., Zini, F., and Zini, F. (2003a). Optorsim - a grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing Applications*, 17:2003.

Bell, W. H., Cameron, D. G., Capozza, L., Millar, A. P., Stockinger, K., and Zini, F. (2002). Simulation of dynamic grid replication strategies in optorsim. In *Journal of High Performance Computing Applications*, pages 46–57. Springer-Verlag.

Bell, W. H., Cameron, D. G., Carvajal-schiaffino, R., Millar, A. P., Stockinger, K., and Zini, F. (2003b). Evaluation of an economy-based file replication strategy for a data grid. In *In Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003 (CCGrid 2003*. IEEE Computer Society Press.

BIRD (2010). http://bird.network.cz/ – retrieved 03/06/2010.

Carns, P. H., Ligon III, W. B., Ross, R. B., and Thakur, R. (2000). PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA. USENIX Association.

Chaudhary, V., Cha, M., Walters, J., Guercio, S., and Gallo, S. (2008). A comparison of virtualization technologies for hpc. In *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, pages 861 –868.

Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. (1999). The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200.

Cloudstore (2010). http://kosmosfs.sourceforge.net/ – retrieved 01/02/2010.

Dewdney, P., Hall, P., Schilizzi, R., and Lazio, T. (2009). The square kilometre array. *Proceedings of the IEEE*, 97(8):1482 –1496.

Dorigo, M. and Di Caro, G. (1999). The ant colony optimization meta-heuristic. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, Londres.

Douceur, J. and Wattenhofer, R. (2001). Optimizing file availability in a secure serverless distributed file system. *Reliable Distributed Systems, IEEE Symposium on*, 0:0004.

Elghirani, A., Subrata, R., and Zomaya, A. Y. (2009). Intelligent scheduling and replication: a synergistic approach. *Concurrency and Computation: Practice and Experience*, 21(3):357–376.

Fisher, M. L. (2004). The lagrangian relaxation method for solving integer programming problems. *Manage. Sci.*, 50(12 Supplement):1861–1871.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549.

GLPK (2009). http://www.gnu.org/software/glpk/ – retrieved 25/05/2010.

GPL (2009). http://www.gnu.org/copyleft/gpl.html – retrieved 25/05/2010.

Hadoop (2010). http://hadoop.apache.org/ – retrieved 06/12/2010.

Hakami, S. (1963). Optimum location of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12:450–459.

Hey, T., Tansley, S., and Tolle, K., editors (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Mich.

Ishii, R. P. (2010). *Optimization input output operations aiming at reduce execution time of distributed applications which handle large amount of data*. PhD thesis, ICMC-USP.

OpenVZ (2010). http://openvz.org/ – retrieved 25/05/2010.

OptorSim (2009). http://sourceforge.net/projects/optorsim/ – retrieved 25/05/2010.

PAN-STARRS (2010). http://pan-starrs.ifa.hawaii.edu/public/ – retrieved 22/11/2010.

Quagga (2010). http://www.quagga.net/ – retrieved 01/06/2010.

Rahman, R., Barker, K., and Alhajj, R. (2008). Replica placement strategies in data grid. *Journal of Grid Computing*, 6(1):103–123.

Ranganathan, K. and Foster, I. (2001). Identifying dynamic replication strategies for a high-performance data grid. In *In Proc. of the International Grid Computing Workshop*, pages 75–86.

Sato, H., Matsuoka, S., Endo, T., and Maruyama, N. (2008). Access-pattern and bandwidth aware file replication algorithm in a grid environment. In *GRID*, pages 250–257. IEEE.

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287.

Starfish (2009). http://wiki.digitalbazaar.com/en/starfish_distributed_filesystem – retrieved 25/05/2010.

Sugerman, J., Venkitachalam, G., and Lim, B.-H. (2001). Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA. USENIX Association.

Tatebe, O., Morita, Y., Matsuoka, S., Soda, N., and Sekiguchi, S. (2002). Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, pages 102–110.

Tatebe, O., Sekiguchi, S., Morita, Y., and Matsuoka, S. (2003). Worldwide fast file replication on grid datafarm. In *Computing in High Energy and Nuclear Physics (CHEP03*.

VirtualBox, S. (2010). http://www.virtualbox.org/ – retrieved 01/02/2010.

Weil, S., Brandt, S. A., Miller, E. L., Long, D. D. E., and Maltzahn, C. (November 2006). Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI 06)*, pages 307–320.