Predicting the Performance of Wide Area Data Transfers

Sudharshan Vazhkudai [1,2] Jennifer M. Schopf [1,3] Ian Foster [1,4]
{vazhkuda, jms, foster}@mcs.anl.gov

[1] Mathematics and Computer Sciences Division, Argonne National Laboratory
[2] Department of Computer and Information Sciences, The University of Mississippi
[3] Computer Science Department, Northwestern University
[4] Department of Computer Science, The University of Chicago

Abstract

As Data Grids become more commonplace, large data sets are being replicated and distributed to multiple sites, leading to the problem of determining which replica can be accessed most efficiently. The answer to this question can depend on many factors, including physical characteristics of the resources and the load behavior on the CPUs, networks, and storage devices that are part of the end-to-end path linking possible sources and sinks.

We develop a predictive framework that combines (1) integrated instrumentation that collects information about the end-to-end performance of past transfers, (2) predictors to estimate future transfer times, and (3) a data delivery infrastructure that provides users with access to both the raw data and our predictions. We evaluate the performance of our predictors by applying them to log data collected from a wide area testbed. These preliminary results provide insights into the effectiveness of using predictors in this situation.

Keywords: *Grids, data transfer prediction, replica selection, information services.*

1. Introduction

As the coordinated use of distributed resources, or Grid computing, becomes more commonplace, basic resource usage is changing. Many recent applications use Grid systems as distributed data stores [10, 18, 21, 26, 28, 31, 38] where pieces of large data sets are replicated over several sites. For example, several high-energy physics experiments have agreed on a tiered Data Grid architecture [23, 24] in which all data (approximately 20 petabytes by 2006) is located at a single Tier 0 site; various (overlapping) subsets of this data are located at national Tier 1 sites, each with roughly one-tenth the capacity; smaller subsets are cached at smaller regional Tier 2 regional sites; and so on. Therefore, any particular data set is likely to have replicas located at multiple sites.

Different sites may have varying performance characteristics because of diverse storage system architectures, network connectivity features, or load characteristics. Thus, users (or brokers acting on their behalf) may want to be able to determine the site from which particular data sets can be retrieved most efficiently, especially as data sets of interest tend to be large (1–1000 MB). It is this *replica selection* problem that we address in this paper.

Since large file transfers can be costly, there is a significant benefit in selecting the most appropriate replica for a given set of constraints [6, 41]. One way a more intelligent replica selection can be achieved is by having replica locations expose performance information about past data transfers. This information can, in theory, provide a reasonable approximation of the end-to-end throughput for a particular transfer. It can then be used to make predictions about the future behavior between the sites involved.

We develop a predictive framework that combines three elements: (1) an integrated instrumentation of a high-performance data server (the Globus GridFTP [2] file transfer service) to record performance information for each file transfer that it performs, (2) predictors to estimate future transfer times, and (3) a data delivery infrastructure building off of the Globus Metacomputing Directory Service (MDS) [9] to provide users with access to both the raw data and our predictions. In the remainder of this paper, we describe these three aspects of our system and then present the results of experimental studies in which we evaluate the effectiveness of using predictors in this situation. We conclude with a discussion of future work.

2. Related Work

The goal of this work is to be able to obtain an accurate prediction of the time required to transfer a large file (10MB to 1GB) between a storage system and a client. Achieving this goal can be challenging since numerous devices are involved in the end-to-end path between the source and the client, and the performance of each (shared) device along the end-to-end path may vary in unpredictable ways.

One approach to replica selection is to construct performance models for each system component (CPUs at the level of cache hits and disk access, networks at the level of the individual routers, etc.) and then use these models to determine a schedule for all data transfers [35]. This approach is widely used in classical scheduling, where the resources are typically

CPUs, not network components, and the action is running a task, not transferring a file [1, 7, 8, 27, 29, 34, 37, 44]. In practice, however, system components are neither dedicated to our use nor under our full control. Hence, their behavior can vary in unpredictable ways as a result of competing traffic and load, unmodeled interactions, or the lack of available data [33].

A promising alternative to system characterization is to use observations of past application performance of the entire system, not on a component-by-component level, to construct predictions of future performance. The use of whole-system observation has three relevant properties for our purposes. First, we can construct such predictions without detailed knowledge of the underlying physical devices. Second, these predictions can, in principle, capture both evolution in system configuration and temporal patterns in load. Third, such predictions are for end-to-end behavior, which is typically what is of interest to the user. This technique is used by the Network Weather Service (NWS) [42] and by NetLogger [30] to predict network (for small file transfers) or CPU behavior, and by Downey [12] and Smith [36] to predict queue wait times.

We analyzed using NWS data for our own predictions. However, since the NWS was intended as a lightweight infrastructure and to have very little overhead, it generally uses small file sizes (the default is 64 KB with standard TCP buffer sizes) to determine network performance. In many cases this can be predictive of larger file transfers, but not for the setting we are examining with the large file sizes and possible parallelism associated with typical Grid data transfers.

Figures 1 and 2 illustrate the differences in performance that can arise between a small NWS probe and an actual file transfer using GridFTP, the file transfer service of the Globus Toolkit™ These two figures plot (in logarithmic scale) the performance measured in a set of controlled experiments with NWS probes and GridFTP transfers over a two-week period between two pairs of sites. Each figure shows approximately 1,500 NWS probes, conducted every five minutes, and approximately 400 GridFTP transfers at irregular intervals. This data is available at [15].

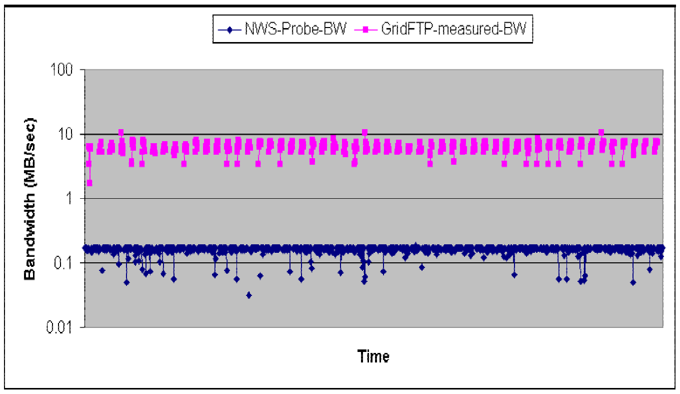

Figure 1: ISI-ANL GridFTP end-to-end bandwidth and NWS probe bandwidth

The NWS measurements indicate network bandwidth to be less than 0.3 MB/sec, while end-to-end GridFTP had a significantly higher transfer rate. More problematic, however, we see considerably greater variability in the GridFTP measurements, ranging from 1.5 to 10.2 MB/sec in both cases, showing that simple data transformations will not improve its predictive merits for this application. The NWS measurements are not the right tool to use, quantitatively or qualitatively, for accurate estimates of GridFTP costs.

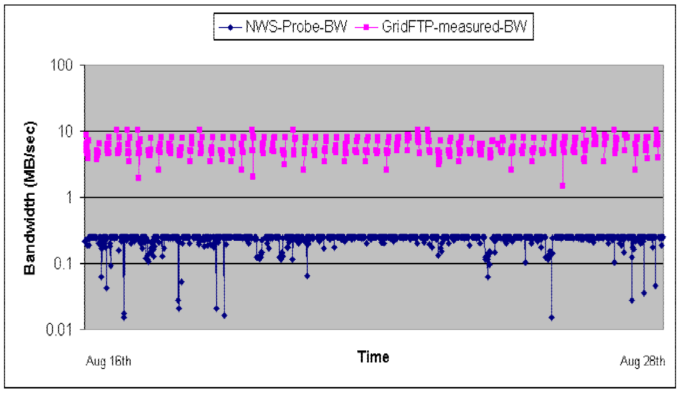

Figure 2: LBL-ANL GridFTP end-to-end bandwidth and NWS probe bandwidth

3. Monitoring GridFTP Performance

As stated earlier, the end-to-end data path consists of several shared devices, including networks, CPUs, and storage systems. However, using predictive techniques on large file transfers can be complicated by the fact that often storage systems are less amenable to "law of large numbers" arguments than are wide area networks or CPUs, for example. No longer does one additional flow or task have an insignificant effect on overall performance.

Because of this, we need to measure the entire transfer function, not just the transport as in other work. To this end, we instrumented the Globus Toolkit™ GridFTP server [6] to record the performance achieved for every data transfer in an end-to-end way. This information, along with metadata indicating the nature of the transfer, serves as the input to our predictors.

The use of actual data transfers to collect performance information, rather than synthetic probes, gives us more accurate data for the full function and imposes no additional probing overhead. The downside is that we have no control over the intervals at which data is collected. This situation can limit the predictive techniques that apply to the data, as detailed in Section 4. In principle, our system could be extended to perform file transfer probes at regular intervals for the sake of gathering data about the performance, and not for transferring useful data, but we do not consider that approach here.

Source IP	File Name	File Size (Bytes)	Volume	StartTime (Timestamp)	EndTime (Timestamp)	TotalTime (Seconds)	Bandwidth (KB/Sec)	Read/Write	Streams	TCP-Buffer
140.221.65.69	/home/ftp/vazhkuda/10 MB	10240000	/home/ftp	998988165	998988169	4	2560	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/25 MB	25600000	/home/ftp	998988172	998988176	4	6400	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/50 MB	51200000	/home/ftp	998988181	998988190	9	5688	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/100 MB	102400000	/home/ftp	998988199	998988221	22	4654	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/250 MB	256000000	/home/ftp	998988224	998988256	33	8000	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/500 MB	512000000	/home/ftp	998988258	998988335	67	7641	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/750 MB	768000000	/home/ftp	998988338	998988425	97	7917	Read	8	1000000
140.221.65.69	/home/ftp/vazhkuda/1 GB	1024000000	/home/ftp	998988428	998988554	126	8126	Read	8	1000000

Figure 3: Sample set from a log of file transfers between Argonne and Lawrence Berkeley National Laboratories. The bandwidth values logged are sustained measures through the transfer. The end-to-end GridFTP bandwidth is obtained by the formula *BW = File size / Transfer Time.*

GridFTP [3] is part of the Globus Toolkit™ [14] and is widely used as a secure, high-performance data transfer protocol [3, 6, 10, 18, 32]. It extends standard FTP implementations with several features needed in Grid environments, such as security on control and data channels, multiple data channels for parallel transfers, partial file transfers, and third party transfers. GridFTP consists of two modules: the control, or server, module and the client module. The server module manages connection, authentication, creation of control and data channels (separate control and data channels facilitate parallel transfers), and reading and writing data The client module is responsible for higher-level operations such as file get and put operations or partial transfers.

We instrumented the GridFTP server by adding mechanisms to log performance information for every file transfer, and this is available a service to any user of the Globus Toolkit™ 2.0 beta version. We added no new capabilities to GridFTP itself; we merely record the data and time the transfer operation. Log entries include source address, file name, file size, number of parallel streams, TCP buffer size for the transfer, start and end timestamps, total time consumed by the transfer, aggregate bandwidth achieved for the transfer, nature of the operation (read or write), and logical volume to and from which file was transferred. A sample log is shown in Figure 3.

The monitoring code is nonintrusive, major overhead being in timing routines, with a smaller percentage spent gathering the information mentioned above and performing a write operation. The entire logging process consumes on average approximately 25 milliseconds per transfer, which is insignificant compared with the total transfer time.

We log data to a standard location in the file system hierarchy and use a single log file for all transfers made to and from the server. Entries are logged in the Universal Logging Format (ULM) “Keyword=Value” format [40]. Each log entry is well under 512 bytes. Transfer logs can grow quickly in size at a busy site. Since old data has less relevance to predictions, we can trim logs based on a running window, as is done in the NWS. An alternative strategy used by NetLogger is to flush the logs to persistent storage (either disk or network) and restart logging. We are exploring these strategies for future GridFTP logs.

4. Prediction Techniques

Simply collecting the data from the GridFTP monitor is not sufficient to make a replica selection decision. In most cases, a prediction of future behavior is needed, not just a recitation of past behavior. In this section we briefly describe some of the predictors we developed, categorize possible approaches by basic mathematical techniques, context-insensitive filtering (using only the last five measurements, for example), and context-sensitive filtering (for example choosing only to use data for similarly sized file transfers). We detail the pros and cons of each technique and then describe the set of predictors we use for our data.

4.1. Basic Mathematical Functions

Mathematical functions for predictions are generally grouped into three categories: mean-based, median-based, and auto-regression model techniques. We use several variations of all of these models in our experiments.

Mean-based, or averaging, techniques are a standard class of predictors that use arithmetic averaging (as an estimate of the mean value) over some portion of the measurement history to estimate future behavior. The general formula for these techniques is the sum of the previous n values over the number of measurements. Mean-based predictors vary with the amount of history information used in their calculations and the amount of weight put on each value. For example, a total average uses the entire set of history data with each value weighted equally, but if more recent behavior has better predictive value, then a subset of the data is used. We discuss these variations in the subsequent section.

A second class of standard predictors is based on evaluating the median of a set of values. Given an ordered list of t values, if t is odd, the median is the $(t+1)/2$ value; if t is even, the median is half of the $t/2$ value added with the $(t+1)/2$ value. Median-based predictors are particularly useful if the measurements contain randomly occurring asymmetric outliers that are rejected. However, they lack some of the smoothing that occurs with a mean-based method, possibly resulting in forecasts with a considerable amount of jitter [20].

A third class of common predictors is auto-regressive models [17, 20, 42]. We use an Auto-regressive Integrated Moving Average (ARIMA) model technique that is constructed using the equation:

$$Y_t = a + bY_{t-1},$$

where Y_t is the prediction for time, t, Y_{t-1} is the previous data occurrence and a and b are the regression coefficients that are computed based on past occurrences of Y. The standard equation includes a shock term, which is not needed in this case.

This approach is most appropriate when there are at least 50 measurements and the data is measured with equally spaced time intervals, which we obviously do not have but we still examine this common technique. The main advantage of using an ARIMA model is that it gives a weighted average of the past values of the series thereby possibly giving a more accurate prediction. However it needs a larger data set than the previous techniques to achieve a statistically significant result, and can have a much greater computational cost.

4.2. Context-Insensitive Factors

More recent values are often better predictors of future behavior than an entire data set, no matter which mathematical technique is used to calculate a prediction. Hence, there are many different variants in selecting a set of recent measurements to use in a prediction calculation.

The fixed-length, or sliding window, average is calculated by using only a set number of previous measurements to calculate the average. The number of measurements can be chosen statically or dynamically depending on the system. We use only static selection techniques in this work. Options for dynamically selecting window size are discussed in [42].

The degenerative case of this strategy involves using only the last measurement to predict the future behavior. Work by Downey and Harchol-Balter [22] shows that this is a useful predictor for CPU resources, for example.

Instead of selecting the number of recent measurements to use in a prediction, we also consider using only a set of measurements from a previous window of time. Unlike other systems where measurements are taken at regular intervals [11, 42], our measurements can be spaced irregularly in time. Using temporal-windows for irregular samples can reflect trends more accurately than selecting a specific number of previous measurements because they capture recent fluctuations, thereby helping to ensure that recent (and, one hopes, more predictive) data is used. Much as the number of measurements included in a prediction can be selected dynamically, the window of time used can be decided dynamically.

As shown in Figure 4, we use fixed-length sets of the last 1 (last value), 5, 15, and 25 measurements. We use temporal-window sets of data of the last 5 hours, 15 hours, 25 hours, 5 days, and 10 days. We consider both mean-based and median-based predictors over previous *n* measurements; mean-based predictors over the previous 5, 15, and 25 hours; and an ARIMA model technique over the previous 5 and 10 days, since this function requires a much larger data set to produce accurate predictions than our other techniques.

4.3. Context-Sensitive Factors

Filtering a data set to eliminate unrelated values often results in a more accurate prediction. For example, a prediction of salary is more accurate when factors such as previous training, education, and years at the position are used to limit the data set of interest.

With the GridFTP monitoring data, initial results showed that file transfer rates had a strong correlation with file size. Studies of Internet traffic have also revealed that small files achieve low bandwidths whereas larger files tend to have high bandwidths [4, 5, 19]. This difference is thought to be primarily due to the startup overhead associated with the TCP start mechanism that probes the bandwidth at connection startup. Recent work has focused on class-based isolation of TCP flows [43] and on startup optimizations [45, 46] to mitigate this problem. As a proof of concept, we found 5–10 percent improvement on average when using file-size classification instead of the entire history file to calculate a prediction. This is shown in Figures 12 and 13, and discussed in Section 6.

For our GridFTP transfer data we ran a series of tests between our testbed sites to categorize the data sizes into a small number of classes. We categorized our data into four sets: 0–50 MB, 50–250 MB, 250–750 MB and more than 750 MB based on the achievable bandwidth. We note that these classes apply to the set of hosts for our testbed only; further work is needed to generalize this notion.

	Average based	Median based	ARIMA model
All data	AVG	MED	AR
Last 1 Value	LV		
Last 5 Values	AVG5	MED5	
Last 15 Values	AVG15	MED15	
Last 25 Values	AVG25	MED25	
Last 5 Hours	AVG5hr		
Last 15 Hours	AVG15hr		
Last 25 Hours	AVG25hr		
Last 5 Days			AR5d
Last 10 Days			AR10d

Figure 4: Context-Insensitive Predictors Used

4.4. Predictors Used

In our initial experiments presented in Section 6 we used a set of 30 predictors over our data sets: 15 predictors each over the entire data set ignoring the context-sensitive factor of data-transfer size, and the same 15 using previous data partitioned by file size. These predictors are summarized in Figure 4. Of course, many other variants for predictors are possible [11, 36, 42]. Also, rather than choosing just a single prediction technique, we could also evaluate a number of them and choose the most appropriate one on the fly, as is done by the NWS.

5. Delivery Infrastructure

Gathering the data is just the first step in building a service to provide predictions for replica selection. The second step, discussed in Section 4, is predicting future behavior based on past information. The third step, described in this section, is defining object classes, integrating this information with a resource provider, and then allowing this information to be discovered in the context of an information service.

Within the Globus Toolkit, the information infrastructure is handled by MDS-2 [9]. This service provides a configurable information provider component called a Grid Resource Information Service (GRIS) and a configurable aggregate directory component called a Grid Index Information Service (GIIS). These components interact with each other and higher-level services (or users) using two basic protocols: a soft-state registration protocol for identifying entities participating in the information service, and an inquiry protocol for retrieval of information about those entities. In brief, a GRIS uses the registration protocol to notify a GIIS (or other higher-level service) of its existence; a GIIS uses the inquiry protocol to obtain information from the known to that provider, which it merges into an aggregate view. This is shown in Figure 5.

Any information provided by a sensor, or information provider, can be used as part of this framework by communicating to a GRIS using a well-defined API. The GRIS and GIIS use the Lightweight Directory Access Protocol (LDAP) [25]. They publish information in LDIF and categorize it under different object classes (comprising multiple attributes and their associated values) as part of the defined schema.

5.1. GridFTP Information Provider

For the GridFTP monitoring data, we built an information provider that accesses the log data to advertise a set of recent measurements as well as some summary statistic data. To generate statistical information on transfers, we developed LDAP shell-backend scripts to filter the information in the logs. In addition, we developed schemas [16] for this data.

Figure 6 presents a fragment of the output from a GridFTP information provider (details include: prediction information, GridFTP server and port information, etc.). Combined, these enable a GridFTP performance information provider to process logs by building schemas and scripts to publish statistical information. Replica locations (sites running GridFTP servers) publish such performance information using GRIS servers.

From our preliminary experiments, a log of approximately 100 KB, around 700 log entries, took the information provider approximately 1 to 2 seconds to filter, classify the entries into object classes, and compute predictions.

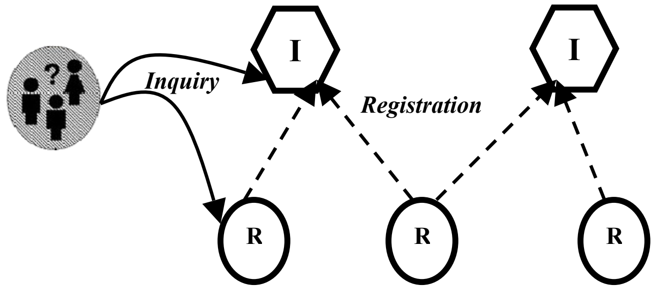

Figure 5: Depicts index servers, GIIS (I), with registered resources, GRIS (R). GRIS, R consists of various information providers (such as the GridFTP performance information provider) registered with it. Depicts user inquiries to GIIS on performance information.

GridFTP Information Provider Output

```
dn:"140.221.65.69,
hostname=dpsslx04.lbl.gov,dc=lbl,dc=gov,o=grid"
cn:"140.221.65.69"
hostname:"dpsslx04.lbl.gov"
gridftpurl:"gsiftp://dpsslx04.lbl.gov:61000"
minrdbandwidth:1462K
maxrdbandwidth:12800K
avgrdbandwidth:6062K
avgrdbandwidthtenmbrange:5714K
………………………………
```

Figure 6: A fragment of the output from the GridFTP performance information provider registered with the GRIS at LBL.

6. Experimental Results

We evaluated the thirty predictors (described in Section 4) on log files obtained from GridFTP transfers on a testbed of three sites: Argonne National Laboratory (ANL), the University of Southern California Information Sciences Institute (ISI), and Lawrence Berkeley National Laboratory (LBL). The results are presented in this section.

6.1. Log files Generation and Preprocessing

The datasets are derived from two GridFTP log files, each containing transfer data collected over a two-week period, one during August and the other during December 2001. Each data set examined transfers over two wide area links: LBL to ANL and ISI to ANL. Each log file contains approximately 350 to 450 transfers. Logs were generated using controlled GridFTP experiments that were performed daily from 6 pm to 8 am CDT, selecting a random file size from the set {1M, 2M, 5M, 10M, 25M, 50M, 100M, 150M, 250M, 400M, 500M, 750M, 1G} and randomly sleeping from 1 minute to 10 hours between file transfers. Figure 7 show how many values were obtained for each file classification size (as discussed in Section 4.3). Traces of log data can be obtained from [15].

Each data transfer was performed by using tuned TCP buffer settings and with parallel streams. GridFTP provides mechanisms to set TCP buffer sizes for transfers. In general, good buffer sizes can be calculated using the formula:

*RTT * "bottleneck bandwidth in the link"*

with RTT values obtained from ping, and bottleneck bandwidth obtained using a tool such as iperf [39]. For our experiments, we used a buffer size of 1MB and eight parallel flows.

For each data set and predictor, we used a 15-value training set; that is, we assumed that at the start of a predictive technique there were at least 15 values in the log file. This number does not imply, in the case of using context-sensitive information, that there were 15 relevant values, only that there were 15 values in the logs to begin to work with.

		August	December
All	LBL	450	365
	ISI	432	334
10 MB	LBL	168	134
	ISI	162	94
100 MB	LBL	112	82
	ISI	108	87
500 MB	LBL	112	82
	ISI	108	87
1 GB	LBL	58	67
	ISI	54	66

Figure 7: Total GridFTP transfers and transfers in terms of file size classification for August and December 2001 datasets.

6.2. Summary of Results

In this section, we analyze our predictions by computing absolute percentage error (difference in predicted and actual measured value), relative performance of predictors with reference to one another and compare error rates with and without file classification. Since there was no statistical significance between the two data sets, we illustrate all our results using the August 2001 datasets. A complete set of graphs, percentage error tables, and logs, for both the August and December 2001 datasets can be found at [15].

We calculate the prediction accuracy using the percentage error calculation:

$$((|Measured_{BW} - Predicted_{BW}|)/Measured_{BW}) *100$$

Figures 8 through 11 show bar charts of percentage absolute error for our 15 predictors for transfers between LBL-ANL and ISI-ANL with various file-size groupings. For each predictor the graph depicts the corresponding error in both LBL and ISI datasets.

The major result from these predictions is that even simple techniques are "at worst", off by about 25%, quite respectable for pragmatic prediction systems. More specifically, we see a marked improvement in predictions when we sort the data by file size and use a file of at least 100 MB. Figures 12 and 13 compare error rates of predictors in the context-sensitive and context-insensitive cases. In general, large file transfers seem to be more predictable than smaller file transfers.

We also measured the relative performance of these 15 predictors by computing the best and worst predictor for each data transfer. Figures 14 through 21 illustrate these results. On average, predictors that had high best percentage also performed poorly more often, thus nullifying any improvement, although median-based predictors seemed to vary more. Of note is the fact that the ARIMA models do not see improved performance for our data, although they are significantly more expensive. This is likely due to the irregular nature of our data. In addition, for our data sets, we did not see a noticeable advantage in limiting either average or median techniques by sliding window or time frames. This result, however, is likely due to the controlled experimental nature of our data.

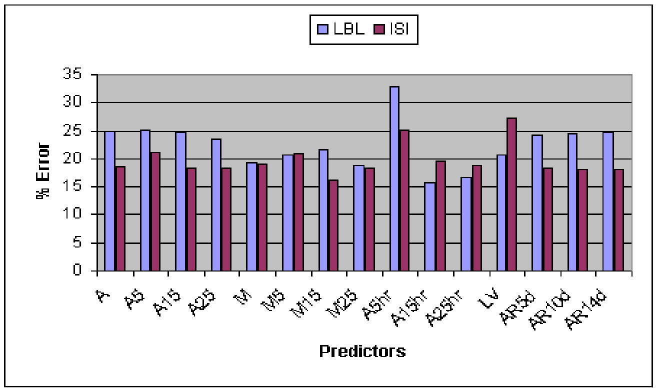

Figure 8: Percent error rates of predictors for LBL-ANL and ISI-ANL 10MB ranges.

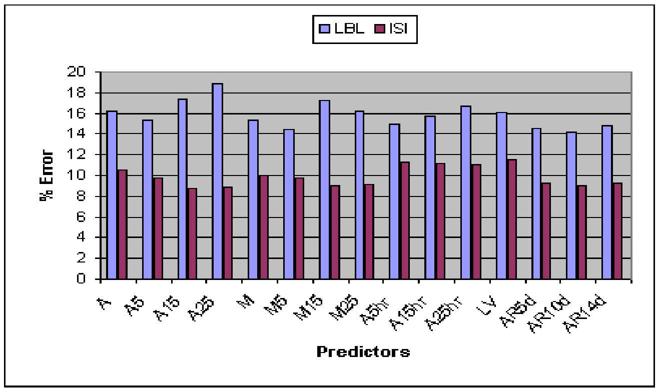

Figure 9: Percent error rates of predictors for LBL-ANL and ISI-ANL 100MB ranges.

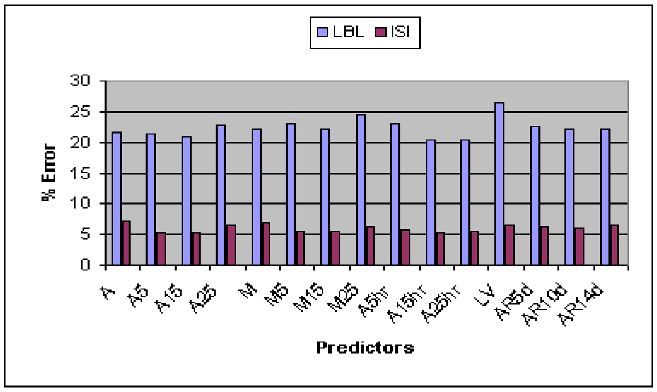

Figure 10: Percent error rates of predictors for LBL-ANL and ISI-ANL 500MB ranges.

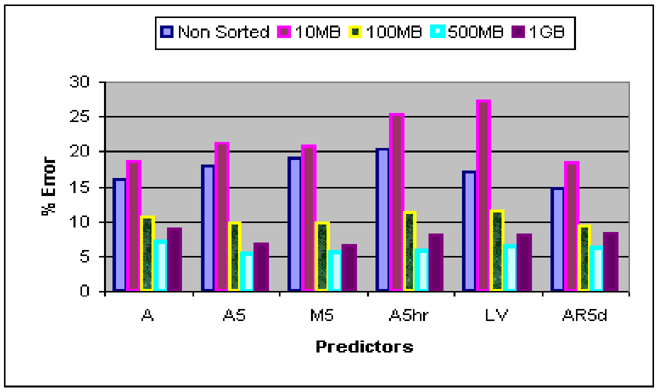

Figure 13: Impact of classification and the reduction in percent error rates for ISI-ANL.

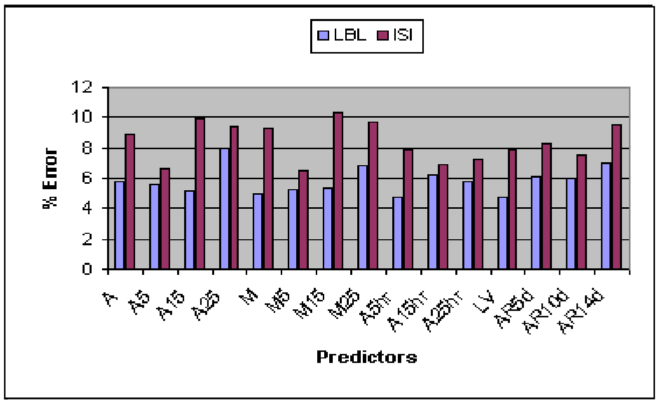

Figure 11: Percent error rates of predictors for LBL-ANL and ISI-ANL 1GB ranges.

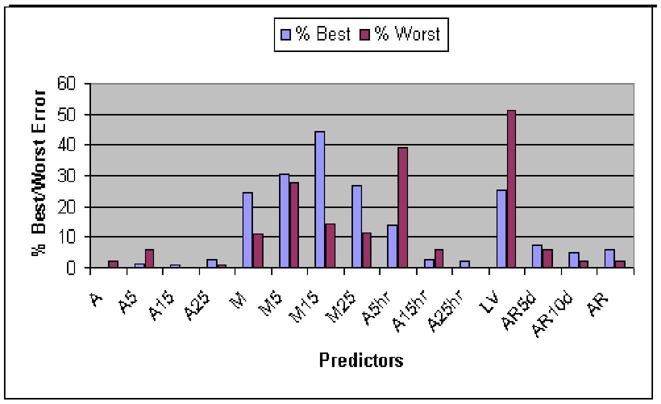

Figure 14: Relative performance of predictors for ISI-ANL 10MB ranges.

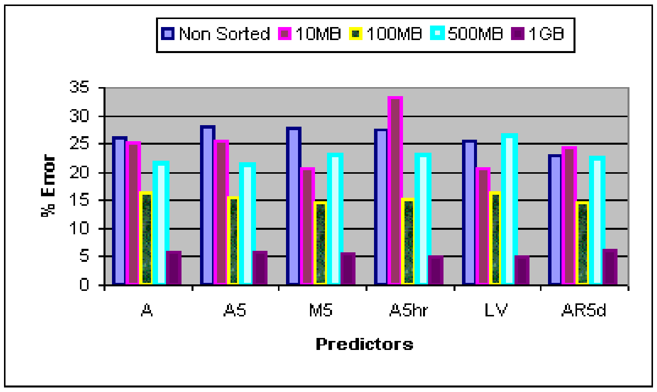

Figure 12: Impact of classification and the reduction in percent error rates for LBL-ANL.

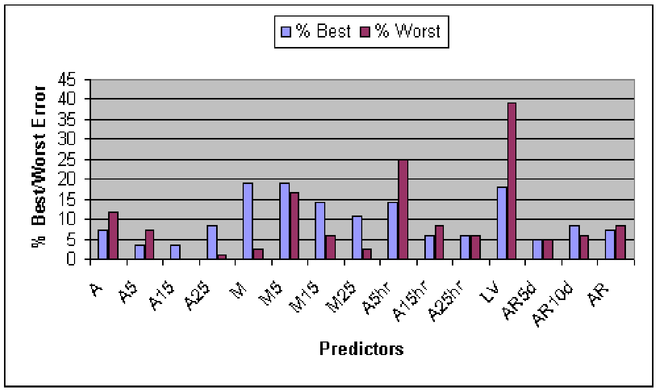

Figure 15: Relative performance of predictors for ISI-ANL 100MB ranges.

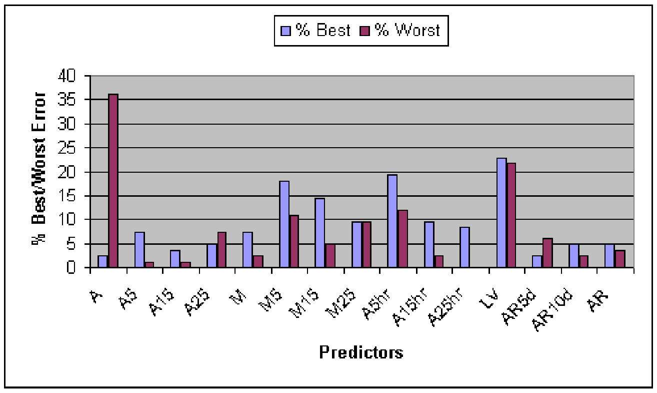

Figure 16: Relative performance of predictors for ISI-ANL 500MB ranges.

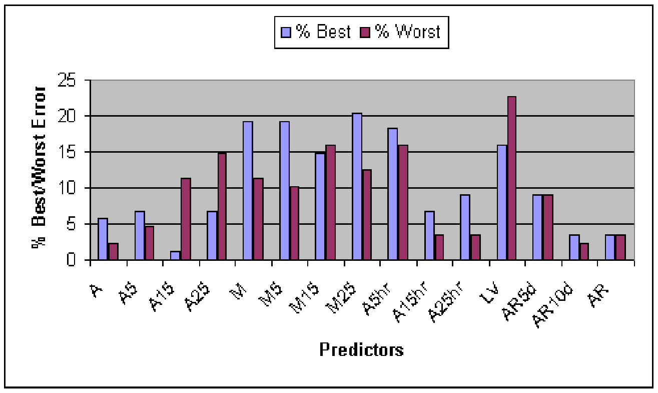

Figure 19: Relative performance of predictors for LBL-ANL 100MB ranges.

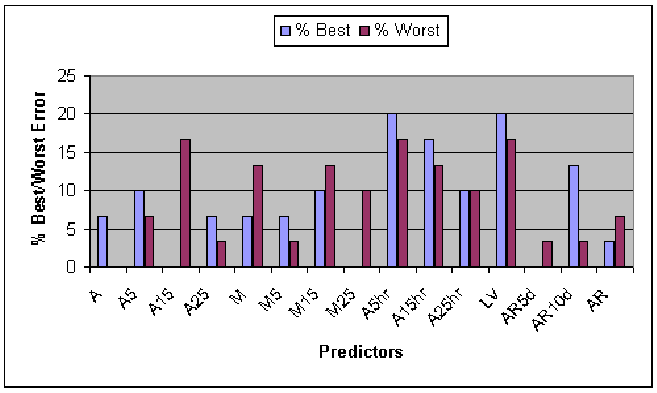

Figure 17: Relative performance of predictors for ISI-ANL 1GB ranges.

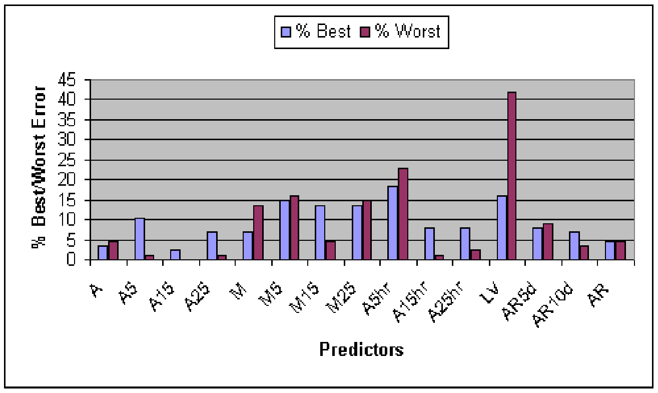

Figure 20: Relative performance of predictors for LBL-ANL 500MB ranges.

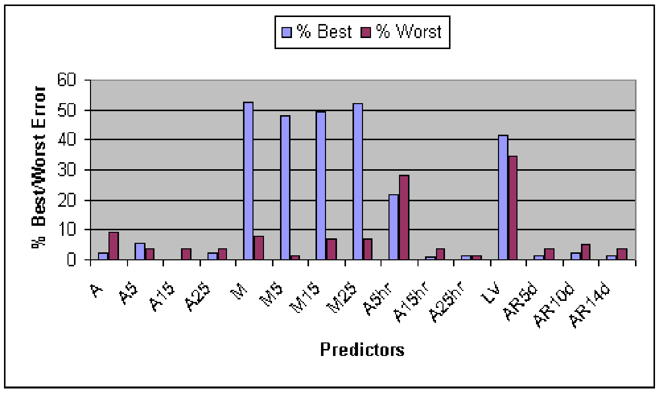

Figure 18: Relative performance of predictors for LBL-ANL 10MB ranges.

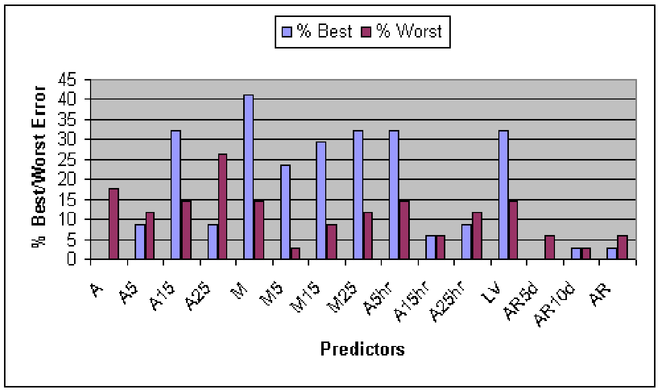

Figure 21: Relative performance of predictors for LBL-ANL 1GB ranges.

7. Conclusions and Future Work

In this paper we have described a technique that takes a step toward resolving the file replica selection problem. We detailed a monitor for GridFTP file transfer behavior, discussed several possible predictive techniques, and showed how data related to this is made accessible as part of the Globus Toolkit information service MDS by means of an information provider that uses GRIS/GIIS components.

Since our work with predictions was inconclusive, our future work will include using additional prediction techniques, as well as the possibility of using the NWS dynamic selection techniques. In addition, we plan to investigate using both basic predictions on the sporadic data combined with more regular NWS measurements and predictions for small regular data movement to overcome the drawbacks of each approach in isolation. Further, to extend the usability of these approaches, we plan to experiment with techniques that will let us extrapolate data when there is no previous transfer data between two sites [13], or to leverage off of other available data in these settings.

Acknowledgments

Many thanks to the Globus Project™ team, especially Bill Allcock. This work was supported in part by the Mathematical Information and Computational Sciences Division Subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract W-31-109-Eng-38, and by NSF Awards ACI-00933000 and PACI#763-04/ACI-9619019.

References

[1] V.S. Adve, Analyzing the Behavior and Performance of Parallel Programs, PhD Thesis, Technical Report TR1201, Department of Computer Science, University of Wisconsin, December 1993.

[2] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing, *IEEE Mass Storage Conference*, 2001.

[3] W. Allcock, I. Foster, V. Nefedova, A. Chevrenak, E. Deelman, C. Kesselman, A. Sim, A. Shoshani, B. Drach, and D. Williams, High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies, *Proceedings of Supercomputing (SC'01)*, November, 2001.

[4] S. Basu, A. Mukherjee, and S. Kilvansky, Time Series Models for Internet Traffic, Technical report GIT-CC-95-27, Georgia Institute of Technology, 1996.

[5] N. Cardwell, S. Savage, and T. Anderson, Modeling the Performance of Short TCP Connections, Technical report, Computer Science Department, Washington University, November 1998.

[6] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, 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, 2001.

[7] M. Cole, Algorithmic Skeletons: Structured Management of Parallel Computation, *Pitman/MIT Press*, 1989.

[8] M.E. Crovella, Performance prediction and tuning of parallel programs, PhD Thesis, University of Rochester, 1999.

[9] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, Grid Information Services for Distributed Resource Sharing, *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[10] The Data Grid Project, http://www.eu-datagrid.org.

[11] P. Dinda and D. O'Hallaron, Host Load Prediction Using Linear Models, *Cluster Computing*, Volume 3, no. 4, 2000.

[12] A. Downey, Queue Times on Space-Sharing Parallel Computers, *Proceedings of the 11th International Parallel Processing Symposium*, 1997.

[13] M. Faerman Alan Su, Rich Wolski, and Francine Berman, Adaptive Performance Prediction for Distributed Data-Intensive Applications, *Proceedings of the ACM/IEEE SC99 Conference on High Performance Networking and Computing*, Portland, Oregon, November 1999.

[14] I. Foster and C. Kesselman, The Globus Project: A Status Report, *Proceedings of IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4–18, 1998.

[15] GridFTP predictor Trace Data, http://www.mcs.anl.gov/~vazhkuda/Traces.

[16] GridFTP Information Provider Schema, http://www.mcs.anl.gov/~vazhkuda/Schema.

[17] N. Groschwitz and G. Polyzos, A Time Series Model of Long-Term Traffic on the NSFnet Backbone, *Proceedings of the IEEE Conference on Communications (ICC'94)*, May 1994.

[18] The GriPhyN Project, http://www.griphyn.org.

[19] L. Guo and I. Matta, The War between Mice and Elephants, Technical report BU-CS-2001-005, Computer Science Department, Boston University, May 2001.

[20] R. Haddad and T. Parsons, *Digital Signal Processing: Theory, Applications, and Hardware*, Computer Science Press, 1991.

[21] M. Hafeez, A. Samar, and H. Stockinger, A Data Grid Prototype for Distributed Data Production in CMS, *7th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT2000),* October 2000.

[22] M. Harchol-balter and A. Downey, Exploiting Process Lifetime Distributions for Dynamic Load Balancing, *Proceedings of the 1996*

Sigmetrics Conference on Measurement and Modeling of Computer Systems, 1996.

[23] K. Holtman, Object Level Replication for Physics, *Proceedings of 4th Annual Globus Retreat*, Pittsburgh, July 2000.

[24] W. Hoschek, J. Jaen-Martinez, A. Samar, and H. Stockinger, Data Management in an International Grid Project, *2000 International Workshop on Grid Computing (GRID 2000)*, Bangalore, India, December 2000.

[25] T.A. Howes and M.C. Smith, *LDAP Programming Directory-Enabled Application with Lightweight Directory Access Protocol.* Technology Series, Macmillan, 1997.

[26] The LIGO Experiment, http://www.ligo.caltech.edu/.

[27] V. W. Mak and S. F. Lundstrom, Predicting the Performance of Parallel Computations, IEEE Transactions on Parallel and Distributed Systems, pp. 106–113, IEEE Computer Society Press, July, 1990.

[28] D. Malon, E. May, S. Resconi, J. Shank, A. Vaniachine, T. Wenaus, and S. Youssef, Grid-enabled Data Access in the ATLAS Athena Framework, *Proceedings of Computing and High Energy Physics 2001 (CHEP'01) Conference*, 2001.

[29] Mark J. Clement and Michael J. Quinn, Analytical Performance Prediction on Multicomputers, *Proceedings of SuperComputing '93*, 1993.

[30] NetLogger: A Methodology for Monitoring and Analysis of Distributed Systems, http://www-didc.lbl.gov/NetLogger.

[31] H. Newman and R. Mount, The Particle Physics Data Grid, www.cacr.caltech.edu/ppdg.

[32] A. Samar and H. Stockinger, Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication, *IASTED International Conference on Applied Informatics (AI2001)*, Innsbruck, Austria, February 2001.

[33] J. M. Schopf and F. Berman, Performance Prediction in Production Environments, *Proceedings of IPPS/SPDP '98*, 1998.

[34] J.M. Schopf, Structural Prediction Models for High Performance Distributed Applications, *Proceedings of the Cluster Computing Conference (CCC '97)*, March 1997.

[35] X. Shen and A. Choudhary, A Multi-Storage Resource Architecture and I/O, Performance Prediction for Scientific Computing, *Proceedings of the 9th IEEE Symposium on High Performance Distributed Computing,* pp. 21–30. IEEE-Press, 2000.

[36] W. Smith, I. Foster, and V. Taylor, Predicting Application Run Times Using Historical Information, *Proceedings of the IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[37] A. Thomasian and P. F. Bay, Analysis Queuing Network Models for Parallel Processing of Task Systems, *IEEE Transactions on Computers c-35* 12, December 1986.

[38] I. Terekhov, R. Pordes, V. White, L. Lueking, L. Carpenter, H. Schellman, J. Trumbo, S. Veseli, and M. Vranicar, Distributed Data Access and Resource Management in the D0 SAM System, *Proceedings of HPDC 2000,* San Francisco, August 2000.

[39] A. Tirumala, J. Ferguson, Iperf 1.2 – The TCP/UDP Bandwidth Measurement Tool, http://dast.nlanr.net/Projects/Iperf/, May 2001.

[40] Universal Format for Logger Messages, http://www-didc.lbl.gov/NetLogger/draft-abela-ulm-05.txt.

[41] S. Vazhkudai, S. Tuecke, and I. Foster, Replica Selection in the Globus Data Grid, *Proceedings of the First IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001)*, pp. 106–113, IEEE Computer Society Press, May 2001.

[42] R. Wolski, Dynamically Forecasting Network Performance Using the Network Weather Service, *Journal of Cluster Computing*, Volume 1, pp. 119-132, January, 1998.

[43] S. Yilmaz and I. Matta, On Class-based Isolation of UDP, Short-lived and Long-lived TCP Flows, Technical report BU-CS-2001-011, Computer Science Department, Boston University, June 2001.

[44] M.J. Zaki, W. Li, and S. Parthasarathy, Customized Dynamic Load Balancing for Network of Workstations, *Proceedings of HPDC'96*, 1996.

[45] Y. Zhang, L. Qiu, and S. Keshav, Speeding Up Short Data Transfers: Theory, Architecture Support, and Simulation Results, *Proceedings of NOSSDAV 2000*, Chapel Hill, N.C., June 2000.

[46] Y. Zhang. L. Qiu, and S. Keshav, Optimizing {TCP} Start-up Performance, Technical report TR99-1731, Department of Computer Science, Cornell University, 1999.