Adaptive Performance Prediction for Distributed Data-Intensive Applications*

Marcio Faerman† Alan Su† Richard Wolski‡ Francine Berman†

August 9, 1999

Abstract

The *computational grid* is becoming the platform of choice for large-scale distributed data-intensive applications. Accurately predicting the transfer times of remote data files, a fundamental component of such applications, is critical to achieving application performance. In this paper, we introduce a performance prediction method, **AdRM** (**Ad**aptive **R**egression **M**odeling), to determine file transfer times for network-bound distributed data-intensive applications.

We demonstrate the effectiveness of the **AdRM** method on two distributed data applications, SARA (Synthetic Aperture Radar Atlas) and SRB (Storage Resource Broker), and discuss how it can be used for application scheduling. Our experiments use the **Network Weather Service** [36, 37], a resource performance measurement and forecasting facility, as a basis for the performance prediction model. Our initial findings indicate that the **AdRM** method can be effective in accurately predicting data transfer times in wide-area multi-user grid environments.

1 Introduction

Ensembles of distributed computational, storage, and other resources, also known as *computational grids* [18], are becoming an increasingly important platform for applications which perform computations over large datasets. Such applications include image acquisition and processing computations, digital library searches, high performance massive data assimilation, distributed data mining and others [3, 6, 16, 17, 20, 25, 35]. Aggregating distributed resources on the grid presents the opportunity to employ or acquire data from massive datasets which are too large to be stored at a single site.

For many distributed data-intensive applications, data movement across the network is a critical determinant of application performance. In particular, in addition to being data-intensive, such applications may also be *network-bound*, with application performance heavily determined by the bandwidth available on network links used during data transfers. Examples of network-bound distributed data-intensive applications include JPL's Synthetic Aperture Radar Atlas (SARA) application [35], which allows the user to select and view images generated from a large, replicated, and distributed database of radar data, and SDSC's Storage Resource Broker (SRB) [6, 30], which provides a uniform interface for users to obtain data from a heterogeneous and distributed collection of data repositories.

Efficient execution of network-bound distributed data-intensive applications on computational grids can be challenging. Although grids offer considerable performance potential through aggregation of resources, application execution performance may be difficult to achieve in practice. In particular, the load and availability of shared resources such as networks may be hard to predict, affecting the ability of an application scheduler to develop performance-efficient application execution strategies.

In this paper, we present a method, Adaptive Regression Modeling (AdRM), for predicting the performance of data transfer operations in network-bound distributed data-intensive applications. Our technique predicts performance in production, multi-user distributed environments by employing small network bandwidth probes (provided by the Network Weather Service (NWS) [36, 37]) to make short-term predictions of transfer times for a range of file sizes. The NWS gathers performance probe data from a distributed collection of resources and catalogs that data as individual performance histories for each resource. It then applies lightweight time series analysis models to each performance history to produce short-term forecasts of future performance levels. In this paper, we make use of the performance probe data, but not of the performance forecasts generated by the NWS. Our approach is to combine NWS measurements (which are non-intrusive and relatively frequent) with instrumentation data taken from actual application runs (which are potentially intrusive but infrequent) to *predict the future performance of the application*.

*This research was supported in part by NASA Graduate Student Research Grant #NGT-2-52251, Department of Defense Modernization Contract 9720733-00, DARPA contract N66001-97-C-8531, National Science Foundation Grants ASC-9701333 and ASC-9619020, and CAPES/Brazil Grant BEX0488/95-2.

†Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093-0114 ({mfaerman,alsu,berman}@cs.ucsd.edu)

‡Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301 (rich@cs.utk.edu)

To capture the relationship between NWS probes and application benchmark data, we use regression models which calibrate application execution performance to the dynamic state of the system measured by the NWS. The result is an accurate performance model that can be parameterized by "live" NWS measurements to make time-sensitive predictions.

The development of performance methods such as **AdRM** is critical to achieving application performance for network-bound distributed data-intensive applications in multi-user computational grid environments. Accurate predictions of data transfer operations such as those provided by AdRM are used in compositional models of data-intensive applications. Such models, used by AppLeS schedulers [8], are often equational, precluding the use of relative ranking of resource performance to perform scheduling decisions. Models which provide an accurate ranking, although useful for some applications [31], may be insufficient for others. **AdRM** provides relatively accurate predictions and relies on observable performance measurements only, and thus can be continuously updated to adapt to current network conditions *automatically* and *in real-time*. We demonstrate the effectiveness of the **AdRM** method for two network-bound data-intensive applications with dissimilar data requirements: the SARA image acquisition application which generally targets relatively small files (1-3 MB), and the SRB query tool which is designed to handle much larger (16 MB or more) files.

This paper is organized as follows: In Section 2, we briefly describe the characteristics of network-bound distributed data-intensive applications, in particular, SARA and SRB, and discuss the importance of accurate predictions for application performance. Section 3 presents several performance models for predicting data transfer times for this application class. We present the **AdRM** prediction method and present experiments which demonstrate its effectiveness for both SARA and SRB in Section 4. In Section 5, we summarize and briefly discuss related and future work.

2 Network-Bound Distributed Data-Intensive Applications

We use the term **data-intensive applications** to denote computations which access and perform operations on numerous or massive datasets. Within this application class, we identify a subclass of **network-bound distributed** data-intensive applications for which a prime determinant of application performance is movement of data across the network.

In general, network-bound distributed data-intensive applications can be partitioned into computational and communication subcomponents that will be assigned over multiple distributed grid resources. Figure 1 represents a typical network-bound application, having

- a *data source* where the application data is stored,

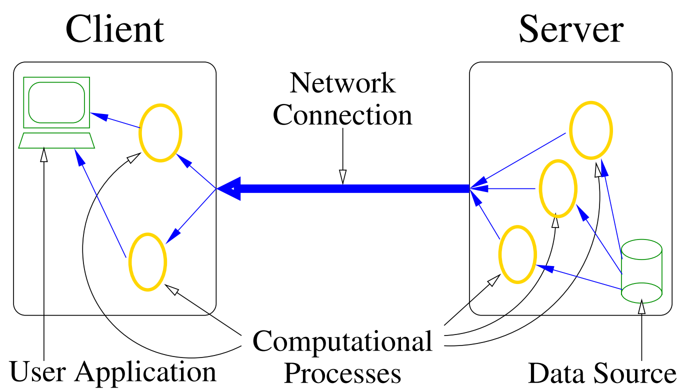


Figure 1: Network-bound distributed data-intensive applications generally involve retrieving data from a server to a client with various data manipulation operations performed at either the client or the server. The computational processes in the figure (the ellipses) are logical entities, representing very complex operations (e.g. data compression or image generation) to extremely simple operations (e.g. sending data to the network unchanged).

- a *user application*, running at the client site,
- several *computational processes* running at the server and client site, capable of performing varying data manipulation operations of varying degrees of complexity, and
- a *network connection*, over which data moves between the client and server (and a principal determinant of application performance for network-bound applications)

To illustrate the characteristics of network-bound distributed data-intensive applications we provide a brief description of JPL's Synthetic Aperture Radar Atlas application and SDSC's Storage Resource Broker.

The Synthetic Aperture Radar Atlas (SARA) [31, 35], developed at JPL and SDSC, is a Web-based distributed data-intensive application which allows users with access to the World-Wide Web to view images of the Earth's surface taken by a synthetic aperture radar. The SARA datasets are replicated across several high-capacity storage sites. Via a Java applet, users of the SARA system can request an image of an arbitrary sub-region with certain features of the data highlighted. A collection of distributed data servers and computational processes retrieve, filter, and convert the raw synthetic aperture data into an image file. SARA's Java user-interface applet allows the user to supply input parameters and view the resulting input file. During certain phases of SARA operation, files typically ranging in size from 1 to 3 MB are transferred between sites.

SDSC's Storage Resource Broker (SRB) [6, 30] is middleware that provides data-intensive applications with a uniform

API to access heterogeneous distributed storage resources systems including file systems, databases, and hierarchical and archival storage systems. SRB provides users the capability to access and aggregate massive quantities of data scattered across wide area networks. In addition to the file transfer semantic, the SRB has a meta-data catalog providing the user the capability of performing queries over the data according to a relational database discipline. Data-collections supported by the SRB include - the Elib Flora collection from UC, Berkeley [15], the Alexandria Digital Library from UC, Santa Barbara [1], the NARA (National Archives and Records Administration) [24] and the Art Museum Image Consortium (AMICO) collections [2].

In order to achieve performance, both SARA and SRB usually need to coordinate the data transfer operations as part of a larger application framework. In this case, file transfer time may be a component of a larger performance model. Accurate predictions of remote file transfer performance are critical in the scheduling of these applications over a distributed environment.

Note that there are several important differences between SARA and SRB. Most obviously, SARA files are typically smaller (1 to 3 megabytes) than SRB files, which can be quite large (tens to hundreds of megabytes). SARA is specifically designed and implemented to be an interactive satellite image processing Web application, whereas the SRB is general-purpose middleware providing facilities for a wide range of distinct distributed data-intensive applications. The differences in their designs, implementations, and the way that they are typically used result in different performance behavior for SRB and SARA when performing file transfers, particularly in the presence of varying network performance.

In this study, we chose to compare SARA and SRB under different usage models. Since SARA is designed to be an interactive visualization tool, we implemented the SARA client to make files accesses relatively frequently – approximately once every five minutes. We hypothesized that large-scale SRB transfers, however, would be relatively infrequent due to their potentially lengthy durations (up to tens of minutes per transfer). As such, we programmed the interval between SRB transfers to range from 2 to 25 hours. These differences provide a wider experimental framework to validate the effectiveness our proposed prediction method, since not only are the applications different, but they access the network on different time scales as well.

2.1 Predicting Application Performance

In order to execute network-bound distributed data-intensive applications, the computation, communication and data transfer components of the application must be mapped to resources. To achieve performance in multi-user grid environments, adaptive scheduling of these components is critical. As part of the AppLeS project, we have found that we can achieve good schedules by using dynamic system resource information in application models which predict application performance [8, 26, 27, 28, 29].

Even for the simple client-server model in Figure 1, effective adaptive scheduling may be non-trivial. For example, for the model shown, the scheduler must determine how these operations can be split between client and server. To illustrate, suppose the user of an SRB geographical image server application requests data about Los Angeles. Due to the nature of the raw data to which the server has access, filtering operations on the relevant data (e.g. highlighting specific characteristics of the dataset) may be necessary. For instance the Los Angeles information might be stored in a large file which also contains data about many other California cities - which however are not relevant to the user. Such filtering operations can be performed either at the server site, at the client site, or partially executed on both sites.

The decision of where and how the filtering will be conducted deeply affects performance. If the requested information is extracted at the server site, then less information will need to be transferred across the network and processed at the client site. However, if the server site is heavily loaded and the network connection between server and client is relatively fast, scheduling the filtering operation on the client location may yield better overall application performance. Conversely, if the network connection is particularly slow then it might be more efficient to use data compression at the server and decompression at the client. Although this strategy would consume more CPU power and time at both sites, the application may perform better overall by avoiding the large data transfer. Andresen et. al. describe additional workload allocation and performance trade-offs issues on distributed data-intensive scenarios in [4].

As the example above illustrates, the decision of how to partition application operations over the resources of a distributed environment may deeply affect performance. Moreover, on a multi-user distributed environment the challenge is further complicated since the deliverable performance of each individual resource can vary dynamically. To accurately forecast application behavior the prediction model must reflect load and performance characteristics of both the application and its target environment. However, developing prediction models which reflect dynamic computational and communication phenomena is difficult, albeit important. In Section 3, we will describe several models for predicting file transfer performance in dynamic grid environments.

2.2 Using Performance Prediction Models

In order to support scheduling, predictions are generally used in two ways: to *rank* alternative candidate schedules or resources according to a given cost function, and as an estimate of the *absolute* performance of components of a larger application model, for coordinating workload allocation over resources. Note that both these uses impose different constraints on what it means for predictions to be "good".

For the "ranking" scenario, it is important for the prediction model to provide *relative* predictions which should be ranked the same as the actual execution under the chosen performance criteria – execution time, throughput, etc. For example, if the application executes in less time (under the same load conditions) with schedule A than with schedule B, a good ranking prediction model would yield execution time predictions P_A and P_B such that $P_A \leq P_B$. Note that we are not necessarily concerned whether P_A is a close match with the actual execution time for the application under schedule A or P_B for schedule B, since the prediction model and cost function only serve to rank alternatives.

However, certain applications or situations such as workload allocation may require a higher level of precision and accuracy regarding *absolute* predictions. Errors in the prediction of the deliverable performance of subcomponents of larger compositional models may aggregate so that the overall application performance model itself might become inaccurate. In particular, we want $P_A \approx T_A$ (where T_A is the actual execution time) when schedule A is part of a larger compositional application performance model. In this case, we are more interested in the error between the actual execution time and the predicted execution time, rather than simple ranking.

To illustrate, a scheduler for the geographical data application described in the previous section requires predictions of the effective computational capacity on the computation sites as well as the deliverable data transfer throughput between the two sites at run time. Having these predictions on hand, the scheduler can find the best allocation of the filtering operation over the client and server computers, coordinating it with the data transfer across the network.

By contrast, consider a "server selection" scenario in which a data consumer must choose between various data servers which contain portions of a replicated dataset. In this case, predictions of file transfer are used to choose between data servers storing a desired replicated data file. In particular, *ranking* predictions are sufficient in this situation, as described in [31] in the context of the SARA application. In contrast, applications which use SRB for data access require reasonably accurate file transfer time estimates in order to make scheduling decisions when the prediction of data transfer time is part of a larger application execution performance model.

This paper focuses on a method to provide accurate *absolute* predictions of remote file transfer performance. In the next section we discuss performance models concerning the characterization and prediction of remote file transfer performance.

3 Performance Models

It would be reasonable to expect that a simple performance model of remote file transfer time for a network-bound distributed application would suffice for scheduling and application execution. In particular, the straightforward model **RBW** (**R**aw **B**and**W**idth model)[1] shown below:

$$FileTransferTime = \frac{DataSize}{AvailableBandwidth}$$

could be used to estimate the time for transferring files between various servers based on the value of $AvailableBandwidth$ between the client and each of the servers. This value could be supplied by a network monitor such as the Network Weather Service (NWS) [36, 37], which measures and forecasts the load and availability of system resources (including network bandwidth). The **RBW** model can be used to predict the file transfer performance of distributed data-intensive applications, like SARA and SRB, in a wide-area grid environment, if the $AvailableBandwidth$ measure accurately represents the effective bandwidth an application would experience.

Our initial experiments with SARA [31] show that the **RBW** model using NWS forecasts can be used effectively to rank alternative candidate schedules. The NWS generated $AvailableBandwidth$ forecasts, based on standard NWS 64 kilobyte data transfers rather than the 1 to 3 megabyte file transfers that SARA uses. Although this scheduler was able to *rank* the servers, the resulting *absolute* transfer time predictions were quite inaccurate. The NWS measurement probes can be parameterized to use arbitrary transfer sizes, however each probe consumes network bandwidth. To obtain an accurate value of $AvailableBandwidth$ for SARA, we would have had to configure the NWS to probe the network using the same message sizes that SARA uses thereby dramatically increasing the intrusiveness of the monitoring. Moreover, this solution would not necessarily guarantee an accurate $AvailableBandwidth$ measure for applications with different characteristics, such as SRB, which typically has larger transfer sizes than SARA.

In addition, **RBW** fails to consider application computational and internal message buffering overheads and overlap between communication and computational operations which may have non-trivial effects on performance for even the most network-bound applications. It is not surprising then that **RBW** model predicts performance relatively inaccurately, as shown in Table 1. Note that while the 64 kilobyte probes do not yield an absolute measure of available bandwidth, the peaks and valleys as predicted by the NWS probes visually correlated with observed file transfer behavior (Figures 2 and 3). Our work in this paper attempts to exploit the correlation between non-invasive probes and actual application be-

[1] In the scope of this paper we consider files large enough such that initial transfer latency is negligible compared to the whole file transfer time.

SRB – NMAE		
site	RBW	AdRM
U.C. Davis	54.10%	11.09%
NCSA	105.20%	9.20%
W.U. St. Louis	96.31%	11.95%
Rutgers	101.56%	1.15%

SARA – NMAE		
site	RBW	AdRM
Utah	34.40%	9.98%
UIUC	36.08%	11.10%
Caltech	15.80%	11.67%

Table 1: **Normalized Mean Absolute Errors (NMAE)** for file transfer throughput forecasts obtained directly from bandwidth measurements **RBW** and from the **Ad**aptive **R**egression **M**odeling **(AdRM)** forecaster.

havior to yield a performance prediction with good absolute accuracy.

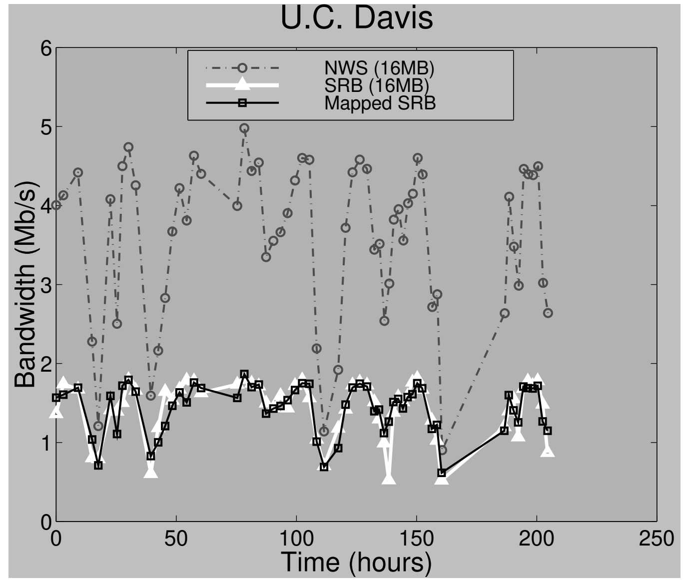


Figure 2: Linear Regression mapping NWS (16 MB messages) to SRB (16 MB file transfers)

3.1 Regression Modeling

Although a sophisticated and detailed performance model could possibly represent almost all the intricacies present in the performance behavior of remote file transfers, the complexity level of this mechanistic model might be so high that its dynamic parameterization at run-time might prove infeasible. Regression modeling is a simple and lightweight method for establishing a functional relationship among variables [9, 11, 14]. In order to achieve more accurate predictions, we considered the use of a linear regression model to address the discrepancy between the performance behavior of small NWS probes and larger data transfers exhibited in the **RBW** model. We developed two linear models that map NWS bandwidth measurements to the observed file transfer behavior of the network-bound distributed data-intensive application within a specified time-frame.

To determine the upper bound on absolute accuracy, we started by regressing large-datafile transfer times with 16MB NWS bandwidth probes. This probe size is too large to be practical for the NWS in general, but it allowed the NWS to more accurately mimic the actual network load during file transfer. Figure 2 shows the results of using this technique to model file transfer times from the University of California at Davis to the University of California at San Diego. The experiments show this regression model parameterized by data from 16MB file transfers and 16MB NWS message size probes. The results were very impressive. We observe in the graph how the modeled datafile transfer performance closely tracks the actual throughput measurements. Our next step was to investigate a less intrusive approach to determine if the technique could be implemented practically.

3.2 A Practical Approach — NWS with 64 kB Probes Only

In order to decrease overhead, we considered a new regression model which uses NWS inputs with 64 kB probes. This is the probe size that is commonly used by the Network Weather Service to perform bandwidth measurements and forecasts. Using small probes, the NWS is able to maintain a low level of intrusiveness on the network [36, 37].

Representative results using the low overhead model, are shown in Figure 3. Note that the model still tracks the actual datafile transfer measurements very closely. This regression model with 64 kB probes can be used successfully to form the basis for an efficient application performance prediction method.

In the next section, we describe the development and use of a performance prediction method which we call **Ad**aptive **R**egression **M**odeling **(AdRM)**.

4 Using AdRM to Dynamically Predict Execution Performance of SARA and SRB

In this section we describe an adaptive regression modeling (**AdRM**) method to **predict** the execution performance of the SRB and SARA grid applications at run-time. Since linear

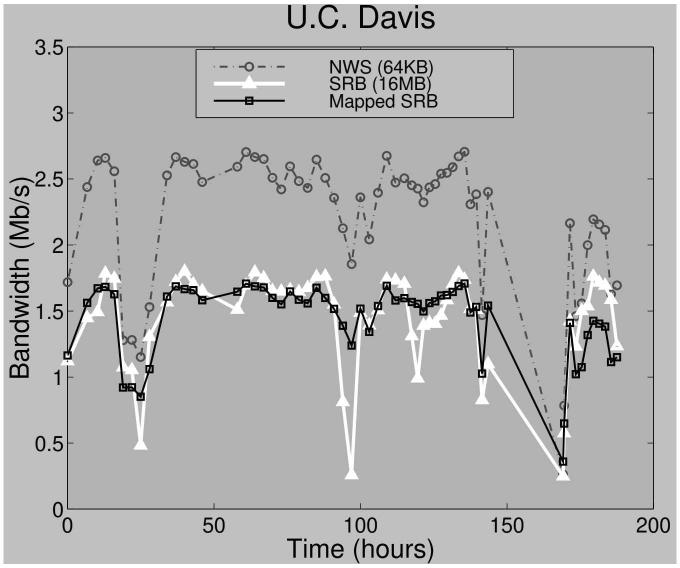


Figure 3: Linear Regression mapping NWS (64 kB messages) to SRB (16 MB datafile transfers)

regression is cheap to compute,[2] we start by deriving an initial regression model from historical application and NWS performance data. As the application executes, we monitor its performance and add the monitored values to the performance history of the program. When a prediction is required, we recalculate the regression coefficients "on-the-fly" using the original performance history, the most recent performance measurements, and the corresponding NWS data for the most recent time frame. In this way, the prediction model evolves and adapts in response to changing performance conditions.

We term the initial set of samples required to "heat-up" the regression model the *Start-up Window*. Having an initial regression model derived from the *Start-Up Window*, we obtain a new bandwidth sample from NWS, and use this value in the regression model to generate a prediction of the first datafile transfer throughput value.

While the application executes, the regression function is updated at each new application file transfer. The new file transfer sample and the network bandwidth value (from the NWS) are incorporated as a new pair within history of network and datafile transfer samples that will be used in the calculation of the next regression model. Then, the updated regression model is used to forecast the next file transfer, and so on.

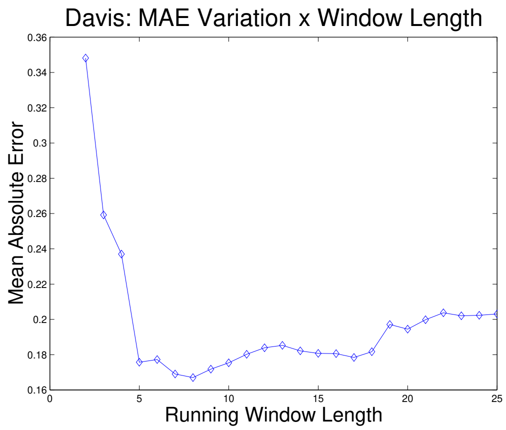
We refer to the set of samples used to generate each new regression model at execution time as the *Running Time Window*. The *Running Time Window* slides over the past history of network and application measurements at each new application transfer. The regression model is updated at each new sample assuring that the forecaster will adapt to the most recent application resource requirements and observed network behavior. At each update, the oldest sample pair of the *Running Time Window* is discarded at each new transfer. In this way, we filter out of the regression model past history that is no longer relevant for new trends of system behavior. The pseudo-code of the **AdRM** prediction method is provided in Appendix A of this paper. Note that **AdRM** generally resembles the ARX (Auto Regression with with eXogeneous variables) modeling approach [22] in the sense that the ARX model structure is also based on regression. The **AdRM** method refits the regression function at each new performance sampling, adapting to changes in the system over time.

The sizes of the *Start-up Window* and *Running Time Windows* are free parameters for the **AdRM** method. While we do not, at present, have an automatic method for determining these parameters from the data, we investigated the effect of varying the size of the *Start-up Window* and *Running Time Window* respectively. After simulating the forecaster for several *Start-Up Window* and *Running Time Window* lengths, we found that the the modeling technique remains accurate for a wide range of window sizes. The algorithm is least sensitive to the length of the *Start-Up Window*, since its effect is felt only initially. Figures 4 and 5 show plots of the variation of mean absolute errors of **AdRM** predictions as the *Running Time Window* lengths range from 2 to 25 past history samples, for the sites analyzed in this work [3]. The *Start-up Window* length is fixed at 2 initial samples. Notice that for small window sizes the error is quite large for the majority of the sites. The error curves flatten at low error values for window sizes larger than 5. For some windows longer than 15 samples, the **AdRM** method starts to exhibit a slightly higher error. We suspect that excessive past history, which is no longer relevant for the future trend of the performance behavior, is being incorporated into the prediction. Furthermore, longer windows are undesirable since they would also require a larger computational cost. Usually a *Running Time Window* length between 8 and 20 samples will yield good predictions for the environments we have investigated. For completeness we have included graphs which show how the mean squared error varies with *Running Time Window* lengths[3] in Appendix B.

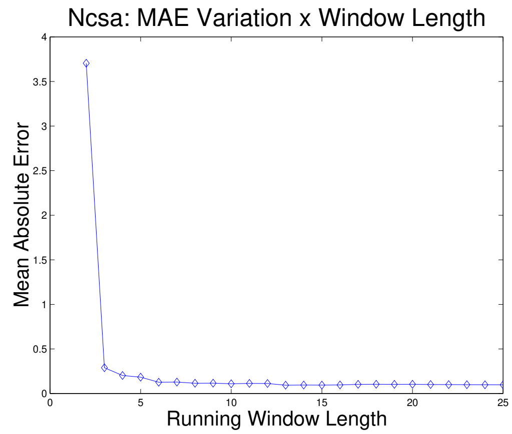
Figures 6 and 7 are representative of a comprehensive set of experiments executed to predict file transfer times using the **AdRM** method for SRB and SARA respectively. In the experiments, SARA transferred files of 3MB, at every 5 minutes from data servers running at the University of Utah, University of Illinois Urbana-Champaign and Caltech. The SRB

[2] We ran our linear regression kernel, written in C, for a total of 8000 iterations (in sets of 1000 iterations at various times during the day) with a dataset of 15 samples. The average execution time for the regression was 13.72 ms on a Sun SparcStation-20 with light to moderate loads from other users. Furthermore, analysis of the linear regression technique shows that the algorithm's asymptotic running time is linear in the dataset length.

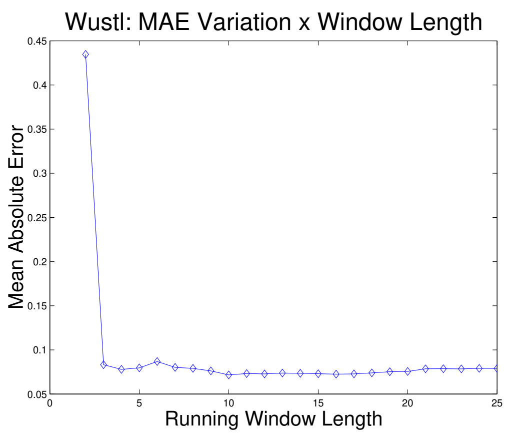
[3] In general small window lengths generated high error levels. To facilitate the visualization of the error variation over the window length range, we omit from some graphs small window size data points which generated very high errors.



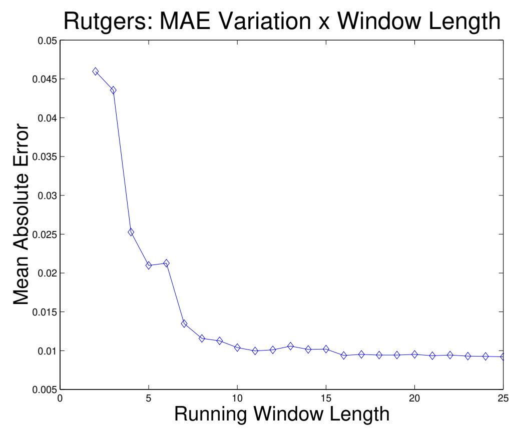
(a) mahler.cipic.ucdavis.edu



(b) vor.ncsa.uiuc.edu



(c) brainmap.arl.wustl.edu



(d) bionic.rutgers.edu

Figure 4: Variation of Mean Absolute Errors of AdRM predictions with *Running Time Window* length for data transfers from SRB sites.

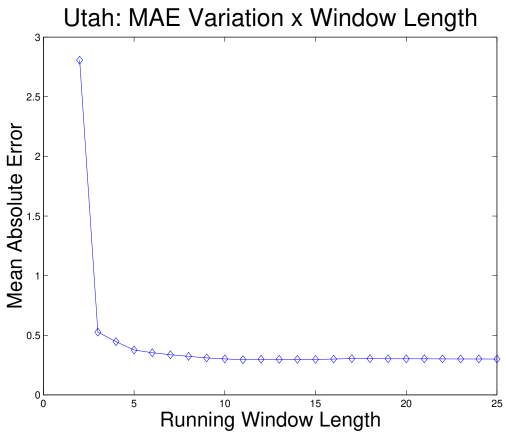
client transferred 16MB files at intervals ranging from 2 to 25 hours from the University of California at Davis, University of Washington at St. Louis, NCSA and Rutgers. Both clients ran at the University of the California, San Diego. In the figures, the differences (the error) between predicted (*in black*) and actual execution times (*in white*) are represented by the vertical distance between each pair of points. The dashed lines are predictions using the **RBW** model. In Table 1, we summarize the error results for this data.

4.1 Analysis

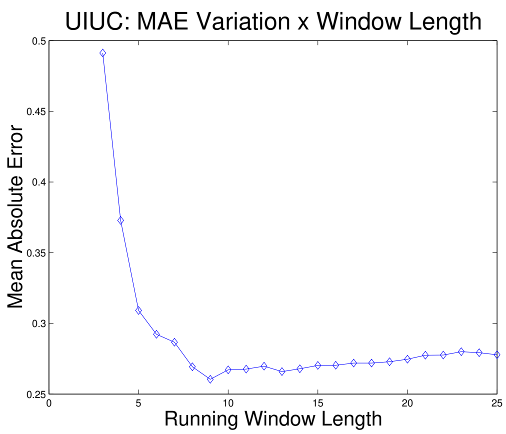
To determine prediction accuracy, we use the *Normalized Mean Absolute Error* **(NMAE)**, given by

$$\sum_{i=1}^{N} \frac{|EstimatedPerf_i - MeasuredPerf_i|}{N \times \overline{MeasuredPerf}} \times 100\%$$

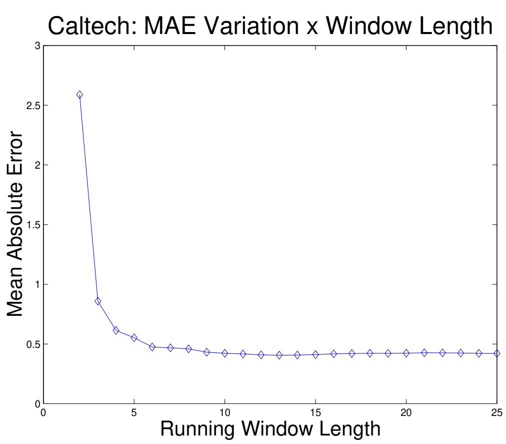
where N is the total number of predicted measurements and $\overline{MeasuredPerf}$ is the mean measured datafile transfer performance for a particular site. The concept behind the *Normalized Mean Absolute Error* is to calculate the mean prediction error, then normalize it by the mean datafile transfer throughput which is given by $\overline{MeasuredPerf}$. In other words, the error is given as a proportion or percentage of the average performance perceived by an application. In this way we provide



(a) perigee.chpc.utah.edu



(b) sitar.cs.uiuc.edu



(c) spin.cacr.caltech.edu

Figure 5: Variation of Mean Absolute Errors of AdRM predictions with *Running Time Window* length for data transfers from SARA sites.

a metric of effectiveness comparable between different applications and sites with distinct performance characteristics.

Observe that the **AdRM** predictions very closely track the actual measured performance of both SARA and SRB applications. Table 1 numerically confirms the efficacy of this method. As it can be observed, the highest relative errors (NMAE) for **AdRM** predictions are on the order of only 10%, whereas the errors for predictions using the **RBW** model reach up to 100%. The largest difference occurs for the Rutgers data server, where the **RBW** model resulted in an error of 101.56%, while the **AdRM** forecasting method predicted with an error of just 1.15%.

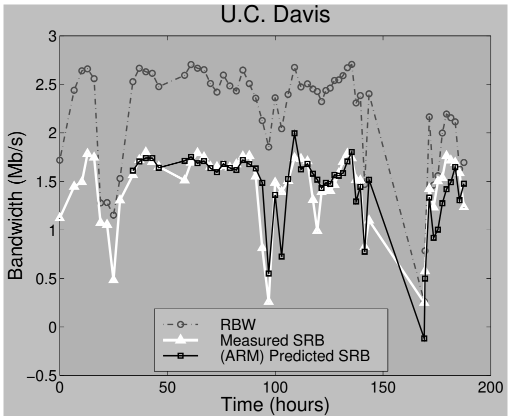
Furthermore, it is important to emphasize that we were able to successfully obtain this high level of prediction accuracy with a low level of intrusiveness on the system — using only small 64 kB messages to probe the network behavior.

In Table 2 we show the mean squared errors (**MSE**) for **RBW** and **AdRM**. The mean squared error calculation emphasizes large errors and attenuates low error samples. Again, we can see that **AdRM** had overall better performance when considering the (**MSE**) metric. Table 3 shows the mean datafile transfer throughput for each data server.

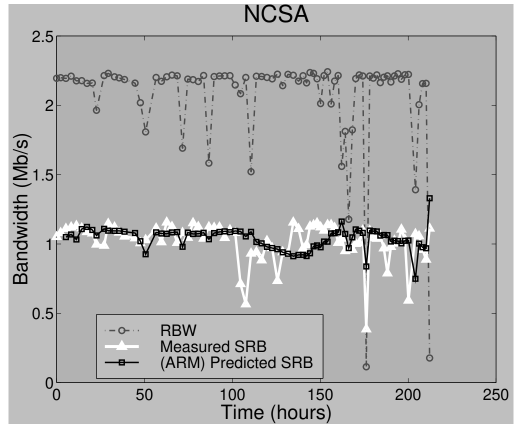
4.2 Using AdRM for Scheduling

The **Appl**ication-**Le**vel **S**cheduling (AppLeS) approach incorporates both application-specific system requirements and dynamic resource performance information to schedule distributed applications in multi-user distributed environments [8, 31, 38]. AppLeS application-level schedulers use a performance model based on the application's communication and computational needs. Performance models can be represented by mathematical equations, in which numeric values for resource performance forecasts are variables [26]. The scheduling agent can compare candidate schedules by evaluating the value of the performance equation for different resource mixes, and choose the resource combination that maximizes application performance. An AppLeS scheduler delays evaluation of the model until run-time, at which point the model parameters are supplied by dynamic performance predictions of various system components, such as remote file transfer throughput and CPU load.

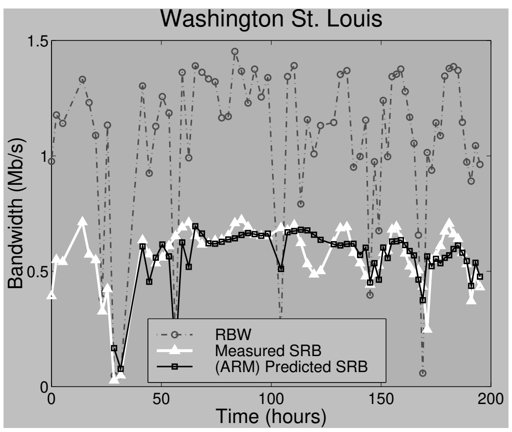
The **AdRM** method uses dynamic information provided by the Network Weather Service(NWS) [36, 37] and past application performance history to generate predictions of future file transfer performance. For the general distributed data-intensive application scenario illustrated in Figure 1, the AppLeS agent would optimize overall application performance, coordinating the computational workload between the client and the server sites with the data transfer operations over the network. **AdRM** would provide file transfer performance predictions which would be incorporated in the compositional performance model used by AppLeS for its scheduling decisions. These file transfer predictions would be inte-



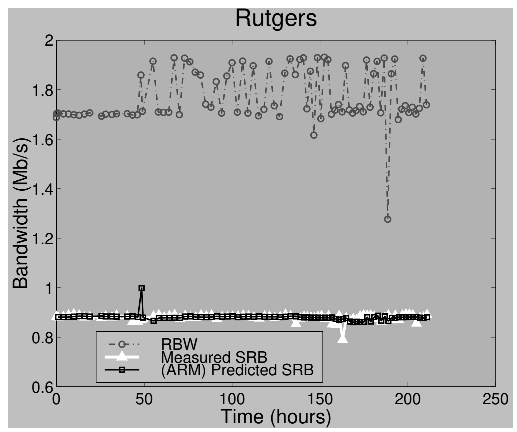
(a) mahler.cipic.ucdavis.edu



(b) vor.ncsa.uiuc.edu



(c) brainmap.arl.wustl.edu



(d) bionic.rutgers.edu

Figure 6: Forecasting SRB datafile transfer behavior with the **AdRM** method.

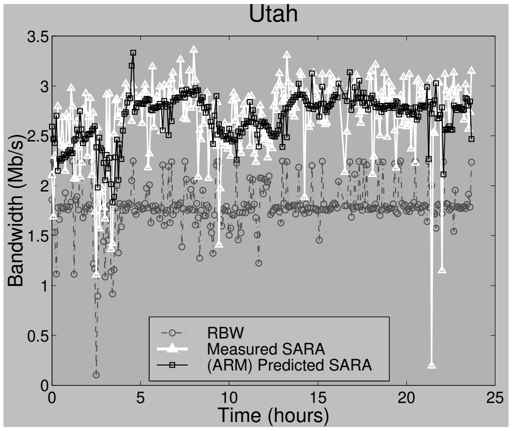
grated with other dynamic system forecasts (e.g. processor load) to parameterize the AppLeS performance model. Resource performance forecasts would be provided by systems such as the Network Weather Service [36] or [13] developed by Dinda et. al .

The initial AppLeS schedulers we have developed [5, 8, 29, 31, 38] have focused on the development of adaptive custom schedules for individual grid applications. We are currently developing AppLeS templates for scheduling structurally similar classes of grid applications. We are focusing on several application classes, including network-bound distributed data-intensive applications.

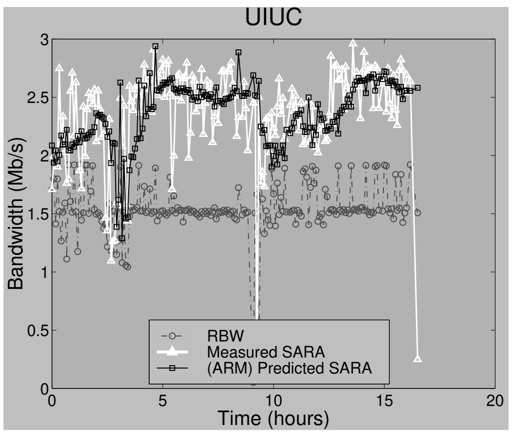
5 Conclusions and Future Work

This paper presents Adaptive Regression Modeling (**AdRM**), a dynamic forecasting method to predict the performance of data transfer operations for network-bound distributed data-intensive applications. Our method achieves a high level of accuracy for exemplar applications SARA and SRB. **AdRM** is a useful and effective application performance prediction method with the following characteristics:

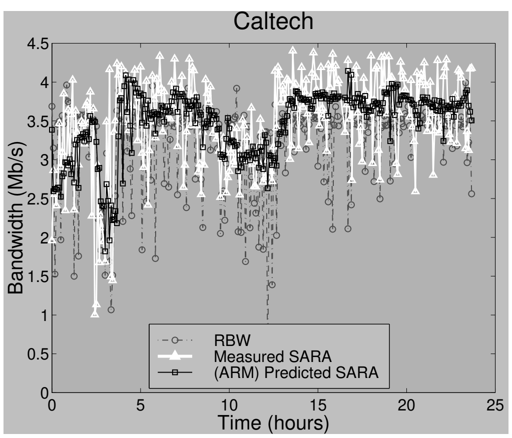
- The prediction model is derived automatically and in real-time, by using a regression model to map measurements of system behavior to observed application performance.
- **AdRM** dynamically adapts in time to changes in the environment by accounting for changes in the workload and



(a) perigee.chpc.utah.edu



(b) sitar.cs.uiuc.edu



(c) spin.cacr.caltech.edu

Figure 7: Forecasting SARA file transfer behavior with the **AdRM** method.

SRB – MSE		
site	RBW	AdRM
U.C. Davis	0.7536	0.0534
NCSA	0.3320	0.0195
W.U. St. Louis	1.2147	0.0112
Rutgers	0.8129	0.0004

SARA – MSE		
site	RBW	AdRM
Utah	0.9652	0.1449
UIUC	0.8690	0.1524
Caltech	0.5128	0.2975

Table 2: **Mean Squared Errors (MSE)** for file transfer throughput forecasts obtained directly from bandwidth measurements **RBW** and from the **Ad**aptive **R**egression **M**odeling **(AdRM)** forecaster. These values are not normalized.

SRB – Mean Throughput	
site	Mbits/s
U.C. Davis	1.4399
NCSA	1.0218
W.U. St. Louis	0.5659
Rutgers	0.8792

SARA – Mean Throughput	
site	Mbits/s
Utah	2.6825
UIUC	2.3577
Caltech	3.4825

Table 3: Mean Measured File Transfer Throughput values for SRB and SARA

system reconfigurations, which are used to parameterize frequent updates of the model.

- Predictions can be computed with low overhead costs – the linear regression is applied to a sliding window of a small number of sample pairs – making it possible to adapt quickly to changing system conditions.

- Internal details of the underlying system are hidden. Basically no knowledge about communication protocols, network topology, or file system configurations was necessary to achieve good predictions. Predictions are based on the measured effect of the system on application execution performance, rather than on explicit knowledge about the system intricacies. For example, even though the SARA application exhibits higher file throughput than the NWS measured bandwidth and exactly the opposite happens for the SRB transfers, the forecaster is able in both instances to make accurate predictions.

Several groups are working on research related to our work. Performance analysis and scheduling of data-intensive applications are described by the ADR group from University of Maryland in [34] and by Thakur in [32]. However, they focus on parallel data servers running over local area networks. Lowekamp also employs statistical techniques considering network and application performance information to generate data transfer predictions at application-level, though his focus regards local area network transfers instead of remote file access operations over wide area networks [23]. Dinda et. al. have examined linear prediction models in many regimes and found such models to accurately predict host load performance over a wide range of system conditions [13]. Andresen et. al. study compositional models and performance trade-offs for distributed data-intensive applications in [4]. Performance monitoring and forecasting of wide-area networks is discussed in works such as the NWS [36, 37], GloPerf [19], ReMoS [12] and [7, 10]. The Netlogger system [33] presents a profiling framework for distributed storage systems.

Many open problems remain. Some preliminary investigation and experimental results indicate that the aggregation of 64-kilobyte network samples collected at short-term intervals increases the correlation level with long file transfers. This method allows a compromise between prediction accuracy and sampling intrusiveness. Furthermore, for low level of correlation between time series, the **AdRM** method behaves as a sliding moving window predictor taking into account mostly past application benchmarking information. We want to investigate further the implications of this scenario.

We intend to extend this work to account for data transfers performed on hierarchical storage systems such as HPSS [21], including systems with tertiary storage (tapes). We also hope to be able to use the **AdRM** method to generate network forecasts for varied communication protocols and configurations and accurate relative ranking among several data servers. In extending the scope of this work to new scenarios, we also intend to look at the possibility of on-demand refinement of the regression model to include additional factors, such as disk or tape behavior and server load, in order to do a better job of assessing end-to-end file transfer, and hence application, performance.

Acknowledgements

The authors would like to thank NPACI researchers Reagan Moore, Chaitanya Baru, Arcot Rajasekar and Michael Wan for their invaluable help and for providing us access to the SRB infrastructure, their constant support, and insightful comments. The SARA application was developed by Roy Williams at CalTech and George Kremenek at SDSC, both of whom helped immensely in our experiments with SARA. We are also very grateful for the important ideas resulting from discussions with our colleagues in the AppLeS group. Essential help was also provided by Jim Hayes in adding new capabilities to the NWS, necessary to run our experiments. Finally, we would like to thank the NPACI sites that provided the data servers for this work, as well as the anonymous reviewers who provided helpful and substantive comments.

References

[1] Alexandria Digital Library Project webpage at `http://alexandria.sdc.ucsb.edu/`.

[2] Art museum image consortium (amico) webpage at `http://www.amico.org/home.html` and `http://www.npaci.edu/DICE/AMICO/`.

[3] A. Amoroso, K. Marzullo, and A. Ricciardi. Wide-Area Nile: A Case Study of a Wide-Area Data-Parallel Application. In *ICDCS'98 - International Conference on Distributed Computing Systems*, 1998.

[4] D. Andresen, T. Yang, O. H. Ibarra, and Ömer Eğecioğlu. Adaptive Partitioning and Scheduling for Enhancing WWW Application Performance. *Journal of Parallel and Distributed Computing*, 1998.

[5] AppLeS webpage at `http://www-cse.ucsd.edu/groups/hpcl/apples`.

[6] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC Storage Resource Broker. In *IBM CASCON '98*, 1998.

[7] S. Basu, A. Mukherjee, and S. Klivansky. Time Series Models for Internet Traffic. *IEEE Comput. Soc. Press*, 1996.

[8] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.

[9] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis — Forecasting and Control*. Prentice Hall, 1994.

[10] R. L. Carter and M. E. Crovella. Dynamic Server Selection Using Bandwidth Probing in Wide-Area Networks. Technical Report TR-96-007, Computer Science Department, Boston University, 1996.

[11] S. Chatterjee and B. Price. *Regression Analysis by Example*. John Wiley & Sons, Inc., 1991.

[12] T. DeWitt, T. Gross, B. Lowekamp, N. Miller, P. Steenkiste, J. Subhlok, and D. Sutherland. ReMoS: A resource monitoring system for network-aware applications. Technical Report CMU-CS-97-194-REVISED, School of Computer Science, Carnegie Mellon University, December 1998.

[13] P. A. Dinda and D. O'Hallaron. An evaluation of linear models for host load predictions. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, 1999.

[14] A. L. Edwards. *An Introduction to Linear Regression and Correlation*. W. H. Freeman and Company, 1984.

[15] UC Berkeley Digital Library Project webpage at `http://elib.cs.berkeley.edu/`.

[16] U. Fayyad and R. Uthurusamy. Data Mining and Knowledge Discovery in Databases. *Communications of the ACM*, 1996.

[17] R. Ferreira, B. Moon, J. Humphries, A. Sussman, J. Saltz, R. Miller, and A. Demarzo. The Virtual Microscope. In *Proc. of the 1997 AMIA Annual Fall Symposium*, 1997.

[18] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[19] GloPerf webpage at `http://www-fp.globus.org/details/gloperf.html`.

[20] R. Grossman, S. Kasif, R. Moore, D. Rocke, and J. Ullman. Data Mining Research: Opportunities and Challenges - A Report of three NSF Workshops on Mining Large, Massive, and Distributed Data, 1998. Available at `http://www.ncdm.uic.edu/M3D-final-report.htm`.

[21] High Performance Storage System webpage at `http://www.sdsc.edu/hpss/hpss1.html`.

[22] L. Ljung. *System Identification - Theory for the User*. Prentice Hall, second edition, 1999.

[23] B. Lowekamp, D. O'Hallaron, and T. Gross. Direct network queries for discovering network resource properties in a distributed environment. In *Proc. 8th IEEE Symposium on High-Performance Distributed Computing (HPDC-8)*, Redondo Beach, CA, Aug. 1999.

[24] NARA Project webpage at `http://www.sdsc.edu/NARA/`.

[25] B. R. Schatz. High-Performance Distributed Digital Libraries: Building the Interspace on the Grid. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, 1998.

[26] J. Schopf and F. Berman. Performance prediction in production environments. *Proceedings of the IPPS/SPDP Conference*, April 1998.

[27] J. M. Schopf. *Performance Prediction and Scheduling for Parallel Applications on Multi-User Clusters*. PhD thesis, University of California, San Diego, 1998. Also available as UCSD CS Dept. Technical Report, Number CS98-607, `http://www.cs.nwu.edu/~jms/Thesis/thesis.html`.

[28] J. M. Schopf and F. Berman. Stochastic scheduling. In *Supercomputing '99 (to appear)*, 1999. Also avilable as Northwestern University, Computer Science Department Technical Report CS-99-3, or `http://www.cs.nwu.edu/~jms/Pubs/TechReports/sched.ps`.

[29] N. Spring and R. Wolski. Application level scheduling of gene sequence comparison on metacomputers. In *Proc. 12th ACM International Conference on Supercomputing*, Jul 1998.

[30] SDSC's **Storage Resource Broker** project webpage at `http://www.npaci.edu/DICE/SRB/index.html`.

[31] A. Su, F. Berman, R. Wolski, and M. M. Strout. Using AppLeS to schedule simple SARA on the computational grid. *International Journal of High Performance Computing Applications*, 13, 1999.

[32] R. Thakur, W. Gropp, and E. Lusk. A Case for Using MPI's Derived Datatypes to Improve I/O Performance. In *Proc. of SC98: High Performance Networking and Computing*, 1998.

[33] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, J. Lee, D. Gunter, and S. Kim. The NetLogger Methodology for High Performance Distributed Systems Performance Analysis. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, 1998.

[34] M. Uysal, T. Kurc, A. Sussman, and J. Saltz. A Performance Prediction Framework for Data Intensive Applications on Large Scale Parallel Machines. In *Proc. of 4th Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers*. 1998.

[35] R. Williams. Caltech's **Synthetic Aperture Radar Atlas** project webpage at `http://www.cacr.caltech.edu/~roy/sara/index.html`.

[36] R. Wolski. Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, August 1997.

[37] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems (to appear)*, 1999. available from `http://www.cs.utk.edu/~rich/publications/nws-arch.ps.gz`.

[38] D. Zagorodnov, F. Berman, and R. Wolski. Application Scheduling on the Information Power Grid. *to appear in International Journal of High-Performance Computing*, 1998.

A AdRM Pseudo-Code

AdRM_Predictor (Win_{Start}, Win_{Run})

If (Initialization) Then
$Samples_{BW}$ $\Leftarrow$ Get_NWS_Bandwidth_History(t_0 : $t_0 + Win_{Start}$)
$Samples_{App}$ $\Leftarrow$ Get_Application_Performance_History(t_0 : $t_0 + Win_{Start}$)
t_{l_s} $\Leftarrow$ Win_{Start} /* Time of Last Sampling */
Else /* Running Time */
$Samples_{BW}$ $\Leftarrow$ Get_NWS_Bandwidth_History($t_{l\_s} - Win_{Run}$: t_{l_s})
$Samples_{App}$ $\Leftarrow$ Get_Application_Performance_History(t_{l_s} - Win_{Run} : t_{l_s})
End If

$\mathbf{F_{Regression}}$ $\Leftarrow$ Calculate_Linear_Regression_Function($Samples_{BW}$, $Samples_{App}$)

```
/* Use previously calculated linear regression model to *
 * predict the file transfer throughput performance     *
 * perceived at application-level                       */
```

New_Bandwidth_Sample = NWS_Get_Bandwidth_Sample()
Predicted_Performance $\Leftarrow$ $\mathbf{F_{Regression}}$(New_Bandwidth_Sample)

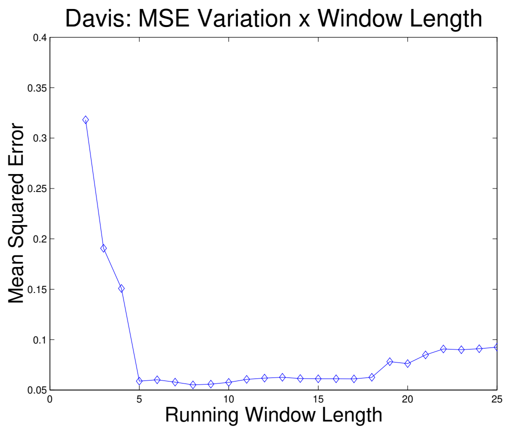
```
/* Prepare for next prediction by storing a bandwidth   *
 * sample and updating the "last sample" timestamp.     *
 * Application history is stored by the AppLeS          *
 * scheduler, which is calling AdRM_Predictor.           */
```

Store_NWS_Bandwidth_History(New_Bandwidth_Sample)
t_{l_s} $\Leftarrow$ t_{now}

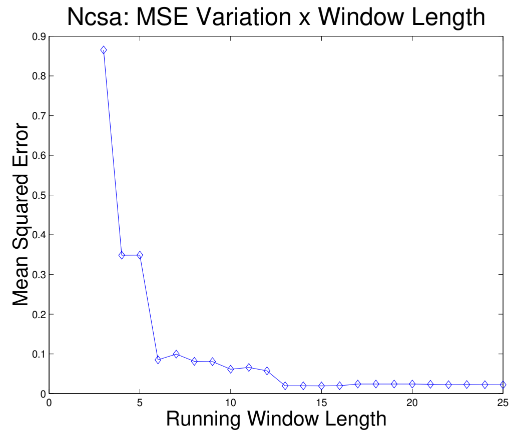
Return(**Predicted_Performance**)

END

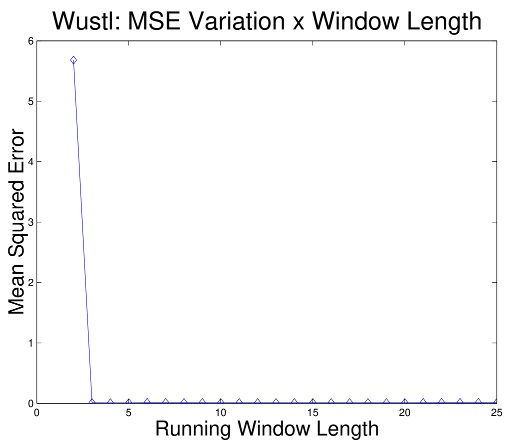
B Variation of MSE with *Running Time Window*



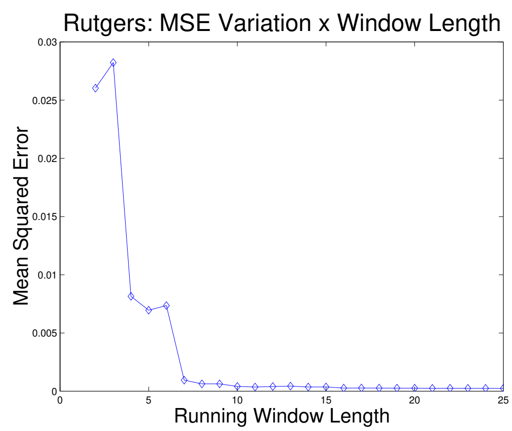
(a) mahler.cipic.ucdavis.edu



(b) vor.ncsa.uiuc.edu

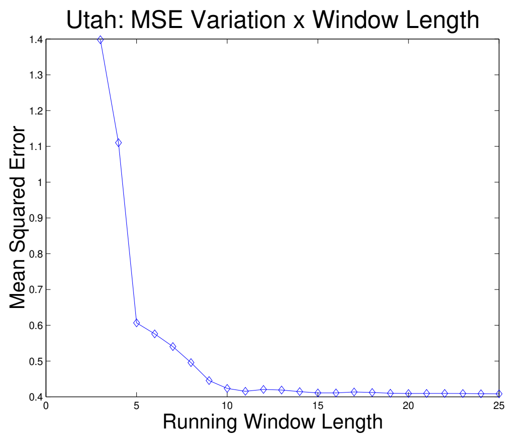


(c) brainmap.arl.wustl.edu

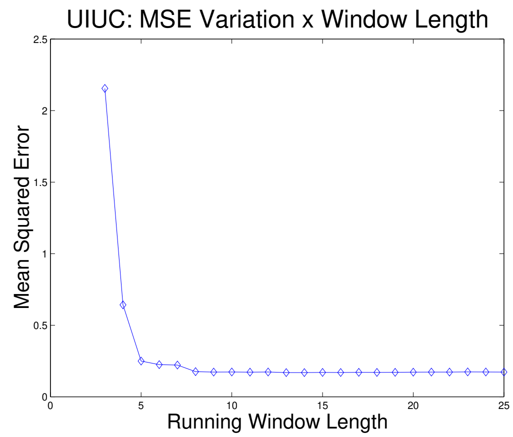


(d) bionic.rutgers.edu

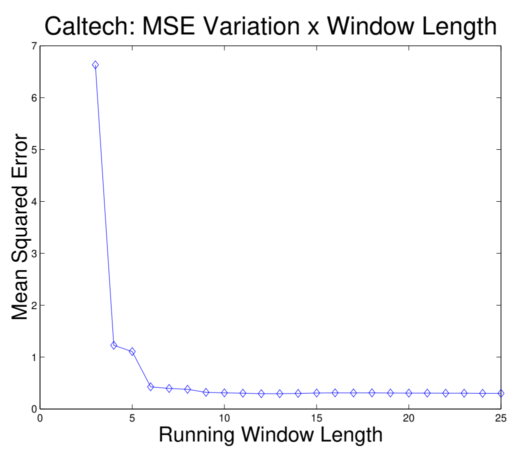
Figure 8: Variation of Mean Squared Errors of AdRM predictions with *Running Time Window* length for data transfers from SRB sites.



(a) perigee.chpc.utah.edu



(b) sitar.cs.uiuc.edu



(c) spin.cacr.caltech.edu

Figure 9: Variation of Mean Squared Errors of AdRM predictions with *Running Time Window* length for data transfers from SARA sites.