Improving Parallel Data Transfer Times Using Predicted Variances in Shared Networks

Lingyun Yang[1] Jennifer M. Schopf[2] Ian Foster[1,2]
[1]*Department of Computer Science, University of Chicago, Chicago, IL 60637*
[2]*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439*
lyang@cs.uchicago.edu [jms, foster]@mcs.anl.gov

Abstract

The increasingly common practice of using multiple distributed storage systems as a distributed data store within which large datasets may be replicated has led to the problem of how to access replicated data efficiently. Multiple-source parallel transfers can improve data throughput time by transferring data from several replicas in parallel. However, we then face the problem of deciding how to distribute the data load among different storage resources. We propose a Tuned Conservative scheduling technique that uses predicted mean and variance network information to make data distribution decisions. This stochastic scheduling technique uses a tuning factor to adjust the amount of the data assigned to a link in accordance with the variability of the network performance. We apply our technique to the GridFTP implementation in the Globus Toolkit and demonstrate that the technique can produce data transfer times that are significantly faster and less variable than those of other techniques.

1. Introduction

In an increasing number of scientific disciplines, large data collections are emerging as important community resources. Furthermore, increases in network speed make it feasible and useful to distribute large data sets across geographically distributed computer and storage resources that may be located in different parts of a country or even in different countries.

For example, scientific experiments such as CMS [8] and ATLAS [11] involve data collections with a total size that will soon reach multiple petabytes. While these experiments may maintain a master copy of their data at a single central site, various (overlapping) subsets of this data are also distributed at national Tier 1 sites, each with roughly one-tenth the capacity; and smaller subsets are cached at national, regional, and university sites. Any particular file is likely to have multiple replicas located at different sites.

Given such multiple replicas, data retrieval can potentially be accelerated by downloading different parts of the file from different sources in parallel. Various file distribution systems and tools (e.g., DPSS [15], BitTorrent [9], GridFTP [2], I2-DSI [3])have been developed to support parallel transfers of distributed or replicated data. The time required for such systems to complete a transfer is strongly influenced by the amount of data fetched from each source, particularly in a heterogeneous and dynamic network environment. Thus arises the problem investigated in this paper: how to determine the amount of data to be fetched from each of several sources.

A simple approach is to request the same amount of data from each source. However, such a scheme is unlikely to result in an efficient data transfer because it does not consider the heterogeneous network connections with sources. While adaptive data decomposition techniques can be used to address heterogeneous resource capabilities, temporal variations in those capabilities have seldom been considered. An alternative approach, which we focus on here, is to use stochastic information about the performance (mean and variance) of past transfers to predict the performance of future transfers.

More specifically, we present a *Tuned Conservative scheduling* technique that uses predicted mean and variance network information to adjust the amount of data allocated to multiple network links. The basic idea is straightforward: we seek to allocate more data to network links that we expect to deliver higher performance. However, we often see that a link with a larger capacity will also show a higher variance in performance and therefore will more strongly influence the transfer time than will a link with less variance. (For example, we observed that during a five-minute period, the bandwidth of a network link from the University of California, San Diego, to the University of Tennessee averaged 1.52 Mb/s, with a variance of 0.12, while the bandwidth of a network link *within* the University of California, San Diego, averaged 30.99 Mb/s with a variance of 31.53). Intuitively, the resource with high variance is less “reliable” and should be allocated less work than resources with less variance in performance. The Tuned Conservative scheduling method uses a tuning factor to adjust the amount of the data assigned to a link in accordance with the variability of the network performance. This technique addresses both the dynamic and the heterogeneous nature of shared network environment.

We evaluate the effectiveness of this scheduling technique by implementing it within the Globus Toolkit® implementation of the GridFTP parallel FTP standard. These results demonstrate that we can achieve significant improvements in both mean transfer times and the variance of those times when compared to nonadaptive schemes in heterogeneous, dynamic environments.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 describes the problem. Section 4 describes our Tuned Conservative scheduling policy. Section 5 presents our experimental results. Section 6 summarizes our work.

2. Related work

Significant previous work aims at providing efficient access to distributed data. The Distributed Multi-Storage Architecture [13] satisfies both performance and capacity requirements of data intensive applications by storing different data sets of one application in different storage resources that may be distributed over heterogeneous networks. However, this system uses only user-provided data and historical performance information to choose the storage resource, ignoring dynamic changes in system performance.

The Internet2 Distributed Storage Initiative (I2-DSI) [3] project is a replicated hosting platform for Internet content and services. Content is distributed at the network edges, improving latency and reducing bandwidth consumption. However, the best replica is selected based only on simple criteria such as data availability and proximity of the server.

Distributed Parallel Storage Server (DPSS) [15] aims to provide image streams fast enough to permit multi-user and real-time applications by using the network to aggregate the server output. Large collections of disks seek in parallel, and all servers send the resulting data to the application in parallel. The performance of the system heavily depends on the data organization, which is determined by the application as a function of data type and access patterns.

Bittorrent [9] is a P2P application that enables efficient access to large amounts of distributed data by enforcing cooperation among clients to elevate file server load and improve data transfer performance using swarming techniques. The amount of data retrieved from each data source is determined by the specification of the data provider and the download speed of the client itself.

Recent work on network performance predication allows the use of predicted information when making data allocation decisions. The Network Weather Service (NWS) [17] provides measurements and a one-step-ahead prediction for network capability by sending out small probes (normally 64 kB) at regular intervals. In the nonstochastic setting, this single *point* value is used to estimate the data transfer time, which is used to help select the best replica. However, single point values are often inaccurate or insufficient representation for characteristic that change over time. Vazhkudai and Schopf [16] also use a point value prediction of file transfer times, but they use GridFTP log files, NWS data, and I/O data and a regression technique.

In our previous work [19], we proposed a conservative scheduling policy able to achieve efficient execution of data-parallel computations in heterogeneous and dynamic environments. This policy uses information about the expected mean and variance of future CPU capabilities to define computing workload mappings appropriate for dynamic resources. In this work, we extend those techniques to evaluate network status, and we use expected mean and variance of network information to guide data allocation decisions among different network links. The result is an approach that exploits predicted variance in network performance information to define a time-balancing scheduling strategy that achieves better data transfer times with smaller variance.

3. Problem statement

Efficient data retrieval in a distributed system can require, in the general case, mechanisms for the *discovery* of source machines that have data replicas, the *selection* of an appropriate subset of those sources, and the *mapping* of the required data onto selected sources. For the purpose of this article, we assume that the target set of sources is fixed, and the dominant factor in data retrieving is data communication. We ignore the disk I/O time and focus on the data allocation problem for multiple link parallel data transfers using network capability information. We do *not* assume that the network links in this resource set have identical or even fixed capabilities. Within this context, our goal is to achieve data assignments that balance load between network links so that each link finishes transferring at roughly the same time, thereby minimizing the total transfer time. This form of load balancing is also known as *time balancing.*

Time balancing is generally accomplished by solving a set of equations, such as the following, to determine the data assignments:

$$T_i(D_i) = T_j(D_j) \qquad \forall i, j \tag{1}$$
$$\sum D_i = D_{Total} \ ,$$

where

- D_i is the amount of data assigned to resource i;
- D_{Total} is the total amount of data need to be transferred ;
- $T_i(D_i)$ is the time needed to transfer D_i data from *i*th data source to destination. It can be calculated by using the following function:

$$T_i(D_i) = \text{EffectiveLatency}_i + D_i/\text{EffectiveBW}_i \qquad (2)$$

To proceed, we need mechanisms for obtaining some measure of future network capability, and translating this measure into an effective network capability that is then used to guide data mapping. As we discuss below, two measures of future resource capability are important: the expected value and the expected variance of that value. One approach to obtaining these two measures would be to negotiate a service level agreement (SLA) with the resource owner under which the resource owners would contract to provide the specified capability [5]. Alternatively, we could use observed historical data to generate a prediction for future behavior [6,12,14,16-18]. We focus in this article on the latter approach and present techniques for predicting the future capability. However, we emphasize that our results for data mapping are also applicable in the SLA-negotiation case, as our techniques can be used to determine how best to adapt to a set of SLAs once they are negotiated.

4. Stochastic time balancing

NWS applies a collection of one-step-ahead prediction strategies to a time series of network or computations resource data and chooses the prediction strategy used for the "next" time step dynamically according to which strategy has been most accurate over recent measurements. However, this one-step-ahead prediction is not sufficient for our purposes. Our transfers may take a significant time, during which network performance may change, unlike the conditions experienced by a simply 64K probe.

What is needed for better data distribution and scheduling is an estimate of the *average* network capability that the data transfer will experience during the entire transfer period, rather than the network information at a single future point in time.

In dynamic environments, we find that a link with a larger capacity typically shows a higher variance in performance and therefore can more strongly influence the transfer time than a link with smaller variance. Thus, the variation of the future capability should also be considered in the scheduling policy.

With these considerations in mind, we have developed a *stochastic scheduling policy* that uses predicted variation information to tune the predicted average network capability and to adjust the data allocated to the network links based on run-time information. To allow for the use of stochastic information, we do not use the one-step-ahead prediction information to calculate how much data should be allocated by the time-balancing formula. Instead, we define the effective bandwidth of a link as

$$\text{Effective BW} = \text{BWMean} + \text{TF} * \text{BWSD}, \qquad (3)$$

where

- BWMean is the predicted mean bandwidth of the network link the data will encounter during transfer,
- BWSD is the predicted variation of bandwidth of the network link the data will encounter during transfer, and
- TF is a per link *Tuning Factor* used to determine the number of the standard deviations to add to the mean value of bandwidth on the network link. This factor is used to determine how conservative the data allocation policy should be. For links with higher variation, we prefer a more conservative scheduling policy.

Notice that *EffectiveLatency* is also a variable in Formula 2. We focus only on the bandwidth because, in our experiment, the latency is only a very small portion of the total data transfer time (< 0.1% for network links within one domain, and <1% for network links across domains). Hence, we ignore the influence of latency when calculating the total data transfer time.

4.1. Predicting mean and variance

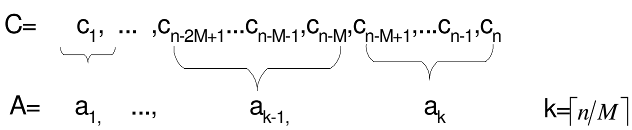
We now describe how to determine the predicted stochastic value—mean and standard deviation of bandwidth—by extending the NWS predictors. We then define the algorithm used to select the Tuning Factor based on the run-time conditions.

4.1.1. Bandwidth mean prediction. Instead of predicting the value at a single future time point, we want to be able to predict the network capability over the time interval of the data transfer. Since the network traffic exhibits a high degree of self-similarity [10], averaging values over successively larger time scales will not produce time series that are dramatically smoother. Thus, to calculate the predicted average network bandwidth the data will encounter during its transfer, we need to first aggregate the original network bandwidth time series into an interval network bandwidth time series and then run predictors on this new series to estimate its future value.

Aggregation, as defined here, consists of converting the original time series into an interval time series by combining successive data over a nonoverlapping larger time scale. The aggregation degree M is the number of original data points used to calculate the average value over a time interval. This value is determined by the resolution of the original time series and the execution time of the applications, and need be only approximate. For example, if the resolution of the original time series is 0.1 Hz (i.e., measured every 10 seconds) and the estimated data transfer time is about 100 seconds, the aggregation degree M can be calculated by

M = transfer time * frequency of original time series (4)
= 100 * 0.1
= 10.

Hence, the aggregation degree is 10, meaning that 10 data points from the original time series are used to calculate one aggregated value over 100 seconds. The process of aggregation is as follows:



where

$C=c_1,c_2,\ldots,c_n$ is the original preceding network bandwidth time series measured at constant-width time interval;

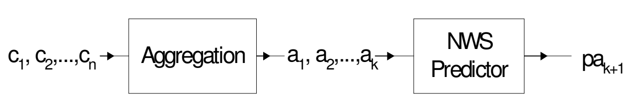
M is the aggregation degree, calculated by Equation 4; and

$A = a_1,a_2,\ldots,a_k$ ($k=\lceil n/M \rceil$) is the interval network bandwidth time series, calculated by Equation 5:

$$a_i = \frac{\sum_{j=1..M} c_{n-(k-i+1)*M+j}}{M} \qquad i=1..k. \qquad (5)$$

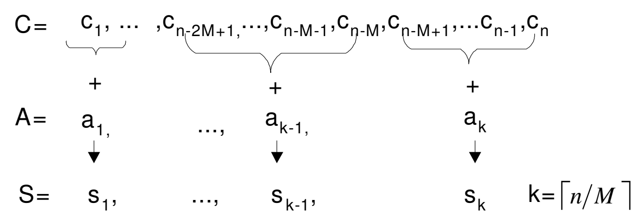
Each value in the interval time series "a_i" is the average network bandwidth over the time interval that is approximately equal to the data transfer time.

After creating the aggregated time series, we use the one-step-ahead NWS predictor on the aggregated time series to predict the mean interval network bandwidth.



The output pa_{K+1} is the predicted value of a_{k+1}, which is approximately equal to the average network bandwidth the data will encounter during its transfer.

4.1.2 Bandwidth variance prediction. To predict the variation of network bandwidth, we use the standard deviation during the data transfer. We need to calculate the standard deviation time series using the original bandwidth time series *C* and the interval bandwidth time series *A* (defined in Section 4.1.1):

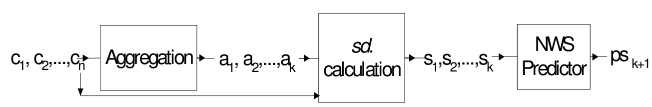


Assuming the original bandwidth time series is $C=c_1,c_2,\ldots,c_n$, the interval bandwidth time series is $A=a_1,a_2,\ldots,a_k$($k=\lceil n/M \rceil$), and an aggregation degree of M, we can calculate the standard deviation bandwidth time series $S=s_1,s_2,\ldots,s_k$:

$$s_i = \sqrt{\frac{\sum_{j=1..M} (c_{n-(k-i+1)*M+j} - a_i)^2}{M}} \qquad i=1..k \qquad (6)$$

Each value in standard deviation time series "s_i" is the average difference between bandwidth and the mean bandwidth over the interval.

To predict the standard deviation of the network bandwidth, we use the one-step-ahead NWS predictor on the standard deviation time series. The output ps_{k+1} will be the predicted value of s_{k+1}, or the predicted bandwidth variation for the next time interval.



4.2. The tuning factor

The goal of this work is to achieve better data assignment on different contended network links in order to reduce the total data transfer time. To this end, we define a conservative scheduling method that uses predicted means (defined in Section 4.1.1) and variances (defined in Seciton 4.1.2) for network capacity information to make data-mapping decisions. To take into account in our data-scheduling decisions the variability of the network capability, we define a *Tuning Factor (TF)*, which represents the variability of the bandwidth as a whole and, as such, provides the "knob" to tune to make use of the scheduling policy more or less conservative. The basic idea behind our approach is to assign less data on network links with a larger variability in performance, which is considered less "reliable". Hence, for a link with more variable bandwidth, effective bandwidth will be smaller.

We calculate EffectiveBandwith as a formula based on the base predicted mean bandwidth value, the TF, and the standard deviation information. Specifically, we vary the number of the standard deviation added to the base bandwidth mean value using the TF. So for a link with a high variance, to calculate a smaller effective bandwidth, we set the TF to be small (adding a smaller number of standard deviation to the base bandwidth mean value).

In previous work [19], we used a similar technique for CPU data. However, unlike CPU load, which usually has a small variance, bandwidth can have a large variation, which sometimes can be as twice large as the mean of the bandwidth. Thus, the Tuning Factor is also needed to limit the influence of the standard deviation on the mean.

So we required a TF value that (1) is inversely proportioanl to the value of variance of the network bandwith. For a link with a larger variance in its bandwidth, we want a smaller TF value, thus a more conservative scheduling policy, vice versa; (2) is able to limit the the number of the standard devation added to the mean, expecially when the variance is large. With these considerations in mind, we use the algorithm in Figure 1 to calculate the Tuning Factor.

```
N=SD/Mean
If (N>1)
   TF=1/(2*N);
Else
   TF=1-N/2;
```

Figure 1.The algorithm to compute our Tuning Factor.

This alogrithm will give a Tuning Factor that has the following characteristics:

(1) TF = 0 to ½ when SD/Mean > 1. Because the higher variation the network link has in its capability, the higher N value it will have. When the standard deviation is larger than the mean of the bandwidth (SD/Mean>1), the network is considered to be high variable and less reliable. We want a smaller TF and thus a smaller effective bandwidth value in this case. A TF value less than ½ can also limit the influence of the standard deviation on the mean when the variance is high.

(2) TF = ½ to 1 when SD/Mean <= 1. When the standard deviation is smaller than the mean of the bandwidth (N <= 1), the network link is considered to be low variable and more reliable. We want a larger TF value and thus a larger effective bandwidth value.

(3) In both cases, the Tuning Factor is inversely proportional to N.

To illustrate our idea more clearly, we calculate the value of TF by our algorithm, while fixing the mean bandwidth value equal to 5 Mb/s and changing the standard deviation of bandwidth from 1 to 15. The results are shown in Figure 2.

We can see from Figure 2 that the TF is inversely proportional to the bandwidth standard deviation (and N), given a fixed mean value. For network links with higher variation, we will have a smaller TF and effective bandwidth value and thus a more conservative data scheduling decision.

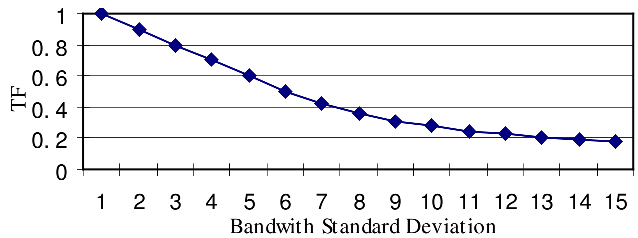


Figure 2. TF value as a function of bandwidth standard deviation, when standard mean is equal to 5 Mb/s.

Note that there are many ways to calcualte the TF value as long as the result meets our basic requirements. The algorithm given in Figure 1 is only one possible approach. The validity of the tuning factor and the tuned conservative scheduling method is evaluted in the next section. However, we admit that there may exist other approaches to calculate the TF value that may further improve the efficiency of the tuned conservative scheduling method.

5. Experiments

To validate our work, we conducted experiments on the GrADs [4] test bed, which comprises workstation clusters at universities across the United States, including the University of Chicago, University of Illinois at Urbana Champaign, University of Tennessee, University of California at San Diego, University of Houston, and University of South California's Information Sciences Institute.

5.1. Experimental methodology

To show the validity of our technique, we define five scheduling policies that we compare in the following experiments:

(1) Best One Scheduling policy (**BOS**): Retrieve data from the network link with the highest predicted mean bandwidth.

(2) Equal Allocation Scheduling policy (**EAS**): Retrieve the same amount of data from each source.

(3) Mean Scheduling policy **(MS)**: Allocate data according to the time balancing formula and use the interval bandwidth prediction, described in Section 4.1.1, for the effective bandwidth. This is equivalent to the Tuning Factor equal to 0. The formula is EffectiveBW = predicted BWMean.

(4) Non-tuned Stochastic Scheduling policy (**NTSS**): Allocate data according to the time balancing formula and use non-tuned bandwidth variability to adjust the value of effective bandwidth. This is equivalent to the Tuning Factor equal to 1. The

formula is EffectiveBW = predicted BWMean + predicted BWSD.

(5) Tuned Conservative Scheduling policy (**TCS**): Allocate data according to the time balancing formula, and use the Tuning Factor as described in Section 4.2 to decide how conservative the scheduling policy should be. For links with higher variability, we estimate more conservative effective bandwidth and thus allocate less data. For this strategy EffectiveBW = predected BWMean + TF * predicted BWSD. The value Tuning Factor adapts from 0 to 1 according to the variation in bandwidth, using the algorithm given in Figure 1.

We implemented multiple-link parallel data transfers using the partial data transfer function provided by GridFTP, part of the Globus Toolkit [7]. We measured the parallel data transfer time achieved for the five scheduling policies described above. We performed experiments on different sets of machines so as to evaluate a range of different network configurations. Every set includes three source machines and one destination machine. We set up our experiments using a single destination machine that received data from three source machines in parallel. Each machine has a replica of the file and provides part of the data, with the amount transferred from each source determined by the scheduling policy. Each pair of source and destination links opens one TCP socket. The networks may encounter contending load from other users during our experiments.

To compare the policies fairly, we alternate scheduling policies for the same data transfers so that any two adjacent runs experience similar load and variation in the environment. For each set of experiments we performed approximately 100 runs, but the experimental data is consistent with larger runs on similarly loaded platforms. For method 1 and 3-5, the effective bandwitdh and the data allocation scheme are recalculated before each run using the run-time information.

5.2. Experimental results

We show results from four representative experiments in Figures 3–6. The resource configurations for the experiments are summarized in Table 1. We performed experiments on different sets of source machines to verify our strategy on different network configurations. We find that network links within a single domain often have larger bandwidth and larger variance, while network links across domains often have smaller bandwidth and smaller variance. Network status is monitored by NWS during experiments. The mean and average standard deviation of bandwidth that each link experienced over the entire run are also shown in Table 1. Note that although the average standard deviation was less than the mean of the bandwidth over the whole period of every experiment, during some particular runs, the standard deviation was larger than the mean value. So both N>1 and N<1 cases were evaluated. We varied the size of the data to be transferred to make every run finish in a reasonable time on different network configurations.

Table 1.Resource configurations used for experiments.

Exp	Destination Machine	Source Machines	BWMean (Mb/s)	BWSD	Data (M)
Fig.3	mystere. ucsd.edu	cirque.ucsd.edu,	3.51	1.44	200
		nouba.ucsd.edu	51.02	23.79	
		dralion.ucsd.edu	19.43	12.76	
Fig.4	mystere. ucsd.edu	torc1.cs.utk.edu	1.51	0.04	200
		nouba.ucsd.edu	47.77	15.77	
		dralion.ucsd.edu	39.75	12.98	
Fig.5	mystere. ucsd.edu	torc1.cs.utk.edu	1.54	0.025	150
		msc01.cs.utk.edu	1.70	0.032	
		dralion.ucsd.edu	48.99	11.59	
Fig.6	mystere. ucsd.edu	torc1.cs.utk.edu	1.56	0.051	100
		msc01.cs.utk.edu	1.72	0.012	
		mckinley.cs.uh.edu	2.71	0.174	

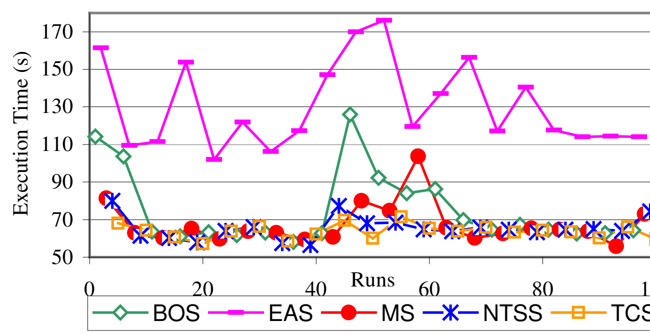


Figure 3. Comparison of the Best One, Equal Allocation, Mean, Non-tuned Stochastic Scheduling, and Tuned Conservative Scheduling policies on the resource set of three source machines in the same domain as the destination machine.

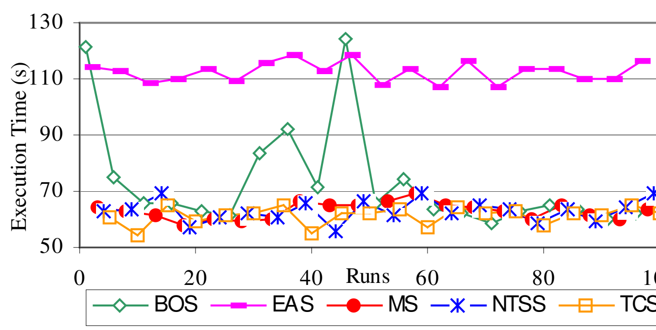


Figure 4. Comparison of the Best One, Equal Allocation, Mean, Non-tuned Stochastic Scheduling, and Tuned Conservative Scheduling policies on the resource set of two source machines in same domain as the destination machine, the third source machine in another domain.

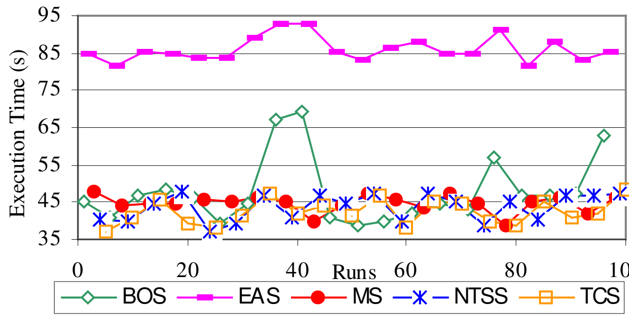


Figure 5. Comparison of the Best One, Equal Allocation, Mean, Non-tuned Stochastic Scheduling, and Tuned Conservative Scheduling policies on the resource set of one source machine in the same domain as the destination machine, the other two machines in another domain.

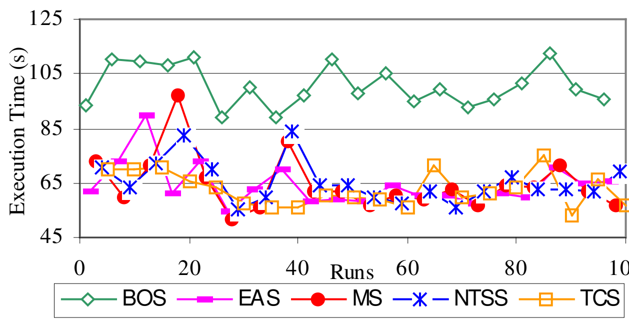


Figure 6. Comparison of the Best One, Equal Allocation, Mean, Non-tuned Stochastic Scheduling, and Tuned Conservative Scheduling policies on the resource set of all three source machines in different domains from that on the destination machine.

The experimental results shown in Figure 3-6 indicate that none of the scheduling polices considered performs constantly best. To compare these policies, we used three metrics: an absolute comparison of transfer times, a relative measure of achievements, and a statistical analysis of the significance of the improvement of our strategy. The first metric involves an average mean and an average standard deviation for all transfer times of each scheduling policy as a whole, as shown in Table 2. This metric gives a rough evaluation of the performance of each scheduling policy over a given interval of time. We can see from the results in Table 2 that over the entire run, the Tuned Conservative Scheduling policy exhibited 3%-51% less overall transfer time than the Best One Scheduling and Equal Allocation Scheduling policies (presumably because it takes load balancing into account) and 2%-7% less overall transfer time than Mean and Non-Tuned Stochastic Scheduling policy (presumably because it takes network performance variability into account). We also see that considering load balancing and variation information in the scheduling policy results in more predictable behavior: The Tuned Conservative Scheduling policy exhibited a 1% - 84% smaller standard deviation in transfer time than the Best One, Equal Allocation, and Non-tuned Stochastic Scheduling policies.

Table 2: Average mean and average standard deviation for entire set of runs for each scheduling policy. The best in each experiment is shown in shade.

Exp	BOS		EAS		MS		NTSS		TCS	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Fig.3	74.82	19.74	130.37	22.89	67.28	10.92	65.38	6.09	**63.70**	**3.69**
Fig.4	73.07	18.97	112.52	3.61	63.02	**2.96**	63.04	3.91	**61.33**	3.03
Fig.5	48.89	9.31	86.04	3.33	45.47	**2.72**	43.65	3.58	**42.34**	3.31
Fig.6	100.61	7.54	64.19	7.92	64.68	10.13	65.37	7.64	**62.58**	**6.33**

The second metric we used, *Compare*, is a relative metric that evaluates how often each run achieves a minimal transfer time. We consider a scheduling policy to be "better" than others if it exhibits a lower transfer time than another policy in five adjacent runs. Five possibilities exist: *best* (best transfer time among the five policies), *good (*better than three policies but worse than one), *average* (better than two policies and worse than two), *poor* (better than one policy and worse than three), and *worst (*worst transfer time of all five policies).

Table 3. Summary statistics using *Compare* to evaluate five scheduling policies, with the largest value in each case shown in shade.

Exp	Policy	Best	Good	Avg	Poor	Worst
Fig. 3	BOS	1	**6**	4	**9**	0
	EAS	0	0	0	0	**20**
	MS	5	3	**10**	2	0
	NTSS	**7**	**6**	2	5	0
	TCS	**7**	5	4	4	0
Fig. 4	BOS	3	3	3	**9**	2
	EAS	0	0	0	2	**18**
	MS	4	**6**	**7**	3	0
	NTSS	5	**6**	5	4	0
	TCS	**8**	5	5	2	0
Fig. 5	BOS	5	3	1	**11**	0
	EAS	0	0	0	0	**20**
	MS	2	6	6	6	0
	NTSS	6	4	**8**	2	0
	TCS	**7**	**7**	5	1	0
Fig. 6	BOS	0	0	0	0	**20**
	EAS	6	5	4	5	0
	MS	5	5	6	4	0
	NTSS	2	4	**7**	**7**	0
	TCS	**7**	**6**	3	4	0

These results are given in Table 3, with the largest value in each case shown shaded. We see that Tuned Conservative Scheduling using predicted mean and tuned variation is more likely to have a "best" or "good" transfer time than the other approaches. This fact suggests that appropriately taking account of the average and variation network information during the period of data transfer in the scheduling policy can significantly improve the transfer time.

Notice that the Equal Allocation Scheduling policy is always “worst” relative to the other approaches in the experiments shown in Figures 3–5. The reason is that in these three experiments, network capabilities are highly heterogeneous. Thus, the EOS strategy of allocating an equal amount of data to all sources results in “unbalanced” workload allocation and poor performance. In contrast, the Best One Scheduling policy performs worst in the experiment shown in Figure 6. During this experiment, network capabilities are similar, and thus load balancing strategies that distribute load over multiple links tends to perform better than the Best One strategy of selecting a single “best” link.

The third metric uses the T-test to evaluate the significance of the improvement of our strategy over other strategies. The T-test is a statistical method used to assess whether the means of two groups are significantly different from each other [1]. The result of a T-test is a set of P-values that indicate the possibility that the differences could have happened by chance: a lower P-value means a more significant difference between two groups, so for our experiments smaller numbers are better. T-tests can be paired or unpaired; a paired T-test is used when the two groups are not independent, and an unpaired test is used when the two groups are independent. For our experiments, we calculate both paired and unpaired T-tests because it was not always clear whether the groups should be considered independent of one another. In addition, T-tests can be one-tailed, which is used when one group is expected to always be less than (or greater than) the other and we know that direction, or two-tailed, which is used only to show a difference that can sometimes be less and sometimes be greater. Since our strategy should always be better than the other strategies, we use a one-tailed test.

The results of the paired and unpaired one-tailed T-tests comparing the Tuned Conservative strategy with the other four strategies are shown in Table 4 and Table 5, respectively, with P-values smaller than 10% shown shaded. These results indicate that the possibility of the improvement happening by chance is small. Thus, we conclude that our Tuned Conservative scheduling policy achieves significant improvements relative to the other three strategies in most cases.

Table 4. Paired one-tailed T test value for the Tuned Conservative scheduling policy relative to each of the other four policies.

Exp	BOS	EAS	MS	NTSS
Fig. 3	**0.92%**	**<0.01%**	**8.26%**	**8.27%**
Fig. 4	**0.78%**	**<0.01%**	**6.09%**	**7.05%**
Fig. 5	**0.30%**	**<0.01%**	**0.37%**	**4.74%**
Fig. 6	**<0.01%**	21.86%	20.28%	**9.40%**

Table 5. Unpaired one-tailed T test value for the Tuned Conservative Scheduling Policy relative to each of other four policies.

Exp	BOS	EAS	MS	NTSS
Fig. 3	**0.80%**	**<0.01%**	**7.50%**	12.07%
Fig. 4	**0.47%**	**<0.01%**	**4.15%**	**6.57%**
Fig. 5	**0.26%**	**<0.01%**	**0.11%**	11.71%
Fig. 6	**<0.01%**	24.05%	21.83%	10.86%

To summarize our results: for all loads and capabilities considered on our test bed, the Tuned Conservative Scheduling policy achieved better results than did the other policies considered. It was both the best policy in more situations under all load conditions, and also the policy that resulted in the shortest transfer time and the smallest variation in transfer time.

6. Conclusion

We have proposed a tuned conservative scheduling policy able to achieve efficient multiple-source parallel data transfers in heterogeneous and dynamic network environments. This policy uses information about the expected mean and variance of future network capabilities to determine the amount of data to transfer from multiple sources. Intuitively, the use of variance information is appealing because it provides a measure of resource “reliability.” Our results suggest that this intuition is valid.

Our work comprises two distinct components. First, we show how to obtain predictions of expected mean and variance network information by extending predictors used in the NWS system. Second, we show how to compute a Tuning Factor that adjusts the degree to which the variability is considered in the scheduling policy, based on expected future mean and variance. The Tuning Factor acts as a “knob” that determines how conservative the data allocation policy should be. We evaluate the effectiveness of our prediction techniques and scheduling policy by applying them to GridFTP. Our results demonstrate that our technique obtains better transfer times and more predictable transfer behavior than do methods that focus on predicted means alone, or that use variances in a less effective manner.

Acknowledgments

Thanks to our colleagues within the GrADS project for providing access to testbed resources. This work was supported in part by the Grid Application Development Software (GrADS) project of the NSF Next Generation Software program, under Grant No. 9975020, and in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract W-31-109-Eng-38.

References

[1] The t-Test: http://trochim.human.cornell.edu/kb/stat_t.htm.

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

[3] Beck, M. and Moore, T., The I-2 DSI project: An Architecture for Internet Content Channels, *Computer Networking and ISDN Systems*, 30 (1998) 2141-2148.

[4] Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Curmmey, J., Reed, D., Torczon, L. and Wolski, R., The GrADs Project: Software Support for High-level Grid Application Development, *The International Journal of High Performance Computing Applications*, 15 (2001) 327-344.

[5] Czajkowski, K., Foster, I., Kesselman, C., Sander, V. and Tuecke, S., SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource management in Distributed Systems. *8th Workshop On Job Scheduling Strategies for Parallel Processing*, Edinburgh, Scotland, 2002.

[6] Dinda, P.A., Online Prediction of the Running Time of Tasks, *Cluster Computing*, 5 (2002).

[7] Foster, I. and Kesselman, C., The Globus Project: A Status Report. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, 1998.

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

[9] Izal, M., Urvoy-Keller, G., Biersack, E.W., Fellber, P.A., A, A.H. and Garces-Erice, L., Dissecting BitTorrent: Five Months in a Torrent's Lifetime. *Proceedings of the 5th Passive & Active Measurement Workshop*, 2004.

[10] Leland, W.E., Taqqu, M.S., Willinger, W. and Wilson, D.V., On the Self-Similar Nature of Ethernet Traffic (Extended Version), *IEEE/ACM Transactions on Networking*, 2 (1994).

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

[12] Schopf, J.M. and Berman, F., Using Stochastic Information to Predict Application Behavior on Contended Resources, *International Journal of Foundations of Computer Science, Special Issue on Parallel Distributed Computing* (2001).

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

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

[15] Tierney, B., Johnston, W.E., Herzog, H., Hoo, G., Jin, G., Lee, J., Chen, L.T. and Rotem, D., Distributed Parallel Data Storage Systems: A Scalable Approach to High Speed Image Servers. *Proceedings of the ACM Multimedia*, 1994.

[16] Vazhkudai, S. and Schopf, J.M., Using Regression Techniques to Predict Large Data Transfers, *Journal of High Performance Computing Applications-Special Issue on Grid Computing: Infrastructure and Applications* (2003).

[17] Wolski, R., Spring, N. and Hayes, J., The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, *Journal of Future Generation Computing Systems* (1998) 757-768.

[18] Yang, L., Foster, I. and Schopf, J.M., Homeostatic and Tendency-based CPU Load Predictions. *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS2003)*, 2003.

[19] Yang, L., Schopf, J.M. and Foster, I., Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments. *Proceedings of SuperComputing 2003*, 2003.