Predicting Slowdown for Networked Workstations

Silvia M. Figueira[1] and Francine Berman[2]

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114

{silvia,berman}@cs.ucsd.edu
http://www-cse.ucsd.edu/users/{silvia,berman}

Abstract

Most applications share the resources of networked workstations with other applications. Since system load can vary dramatically, allocation strategies that assume that resources have a constant availability and/or capability are unlikely to promote performance-efficient allocations in practice. In fact, it is critical to provide a realistic model of the effects of contention on application performance in order to best allocate application tasks to machines. In this paper, we present a model that provides an estimate of the *slowdown* imposed by competing load on applications targeted to high-performance clusters and networks of workstations. The model provides a basis for predicting realistic communication and computation costs and is shown to achieve good accuracy for a set of scientific benchmarks commonly found in high-performance applications.

1 Introduction

In the last decade, networks of workstations have emerged as powerful platforms for executing high-performance parallel applications. For such applications, the development of a performance-efficient allocation of tasks to machines is dependent upon a realistic prediction of application behavior under changing system load. In particular, additional applications executing on the system can dramatically affect the availability and capability of resources, and must be factored into predictions of application execution costs. If an effective predictive model can be developed for time-shared multi-user distributed environments, applications can be allocated in a way that promotes execution performance in the system.

In current work, machine workload has been used to parameterize the allocation of tasks to workstations in a network. However, many allocation strategies do not consider load characteristics in the measurement of workload (e.g., [1, 2, 4, 6]). Load characteristics have been included in performance prediction models for networks of workstations (e.g., [10, 15]), but such models

1. Supported in part by CAPES and UFRJ (Brazil), and by NSF contract number ASC-9301788.
2. Supported in part by NSF contract number ASC-9301788.

assume that each workstation is shared by at most one compute-intensive task and local tasks which alternates idle with compute-intensive cycles.

We believe that a model to predict contention effects on application performance must reflect both system characteristics and workload behavior. The dependence on workload behavior reflects the fact that different applications utilize different types of resources, and the requirements for these resources vary during the execution of each application. The dependence on system characteristics reflects the fact that applications affect one another in different ways on different systems.

In [8], we proposed a contention model based on system characteristics and workload behavior for two-machine networks. In this paper, we develop a contention model for application performance on high-performance clusters and networked workstations. Our model is based on the evaluation of a *slowdown factor* which is used to predict computation and communication costs in a time-shared multi-user networked environment. The parameters required for our models are detailed enough to allow good accuracy, but reasonable to obtain or calculate at run-time.

This paper is organized as follows. Section 2 introduces our approach. In Section 3, we analyze contention effects in high-performance clusters. Section 4 discusses contention effects in workstation networks. Section 5 evaluates the model proposed. Section 6 concludes with a summary and future work.

2 Approach

In the following sections, we define a contention measure, the slowdown factor, to adjust the computation times and communication costs of an application to accommodate for system load. In a networked system, application tasks may be assigned to distinct execution sites to promote concurrent execution and/or to take advantage of distributed data sources, aggregate memory, or I/O bandwidth. If slowdown can be defined accurately, it can be utilized to produce more realistic predictions of application performance. Such predictions form the foundation of a performance-efficient strategy for scheduling. Consequently, we model slowdown for the time required to compute an application task at a single execution site in the system, and for the time required for communication between distinct execution sites.

Define

- $X_{m,\,ded}$ = time to execute task X on machine m in dedicated mode, and

- $(X_m \rightarrow Y_n)_{ded}$ = cost to transfer data between task *X* and task *Y*, when task *X* executes on machine *m* and task *Y* executes on machine *n*, in dedicated mode.

Since computation and communication costs are affected by contention for resources on both machines and on the communication link, the time to execute task *X* on machine *m* is given by

$$X_m = X_{m,\,ded} \times sd_m, \tag{1}$$

and the cost to transfer data between task *X* and task *Y*, when task *X* executes on machine *m* and task *Y* executes on machine *n*, is

$$X_m \rightarrow Y_n = (X_m \rightarrow Y_n)_{ded} \times sd_{m \rightarrow n}, \tag{2}$$

where sd_m is a slowdown factor dependent on the load in machine *m*, and $sd_{m \rightarrow n}$ is a slowdown factor dependent both on the traffic between machine *m* and machine *n* and on contention in both machines. Note that sd_i will be calculated differently for each component *i*.

Equations (1) and (2) demonstrate how slowdown factors are used with a predictive model. The following sections show how to derive slowdown factors for two different platforms: high-performance clusters and workstation networks.

We make the following assumptions: We assume applications to be coarse-grained, scientific programs that are basically CPU-bound. We assume that the applications executing on each workstation fit in memory, or at least that their working sets fit in memory, so that no delay is imposed by extra swapping caused by lack of memory space. We also assume multiple applications to have the same priority as one another and are scheduled locally in round-robin fashion on the time-shared systems. In practice, most operating systems executing on workstations employ a priority-based scheduling strategy that reduces to a round-robin policy when the executing applications are CPU-bound [7]. Since we assume the applications to be coarse-grained, considering a round-robin local scheduler for the time-shared systems is a reasonable assumption.

We assume that communication between machines involves the transference of large bursts, requires data conversion, and uses TCP/IP via sockets.

Our calculation of slowdown factors currently requires information about all the applications executing on the system, i.e., we assume that information about the load is available or is easily obtainable. This is a reasonable assumption for local environments (e.g., distributed facilities at the San Diego Supercomputer Center or UCSD Parallel Computation Laboratory), where such information is typically available. Determining slowdown factors when only partial information about contending applications is known is part of our future research plans.

Finally, we assume that the time to execute a task in dedicated mode to be known or to have been calculated previously. The contribution of this research is to use dedicated time estimates and slowdown factors to provide reasonably accurate estimates of application behavior in a time-shared multi-user system.

To validate our models we have used scientific computations on systems in which contention is generated by synthetic loads. In this paper, we demonstrate our model on serial versions of three benchmarks commonly used in scientific applications: an SOR benchmark [5], which solves Laplace's equation and was developed using PVM [13], a Matrix Multiplication benchmark [11], which was developed using KeLP [9] and MPI [12], and a Multigrid benchmark [5], which was also developed using KeLP and MPI. We have verified the model with various load conditions. In the experiments performed, contention was generated by a synthetic load generator in which computation and communication cycles alternate. The load was parameterized so that the duration of the computation cycle, the direction of the communication, and the number of messages and message size in a communication burst, as well as other parameters could be varied.

3 Contention Effects in High-Performance Clusters

Consider a set of homogeneous workstations connected by a dedicated high-speed network. Examples of this type of platform include the DEC Alpha-Farm and the IBM SP-2. We consider each node in the cluster as an individual machine and calculate the slowdown on computation and communication costs as sd_a and $sd_{a \to b}$, where a and b are two nodes in the cluster.

We demonstrate our results using three nodes of a DEC Alpha-Farm[3], which we call m_1, m_2, and m_3. Given this scenario, we want to calculate the delay imposed on both communication between nodes m_1 and m_2 and computation on node m_1.

In a cluster, applications executing on a machine may be communicating with various machines. Our experiments show that the delay imposed by communication activity on both computation and communication costs may vary with the *bandwidth available* on the links used by the competing applications. This happens because the bandwidth available reflects the amount of CPU used for communication and, consequently, the delay imposed by the respective communi-

[3] The DEC Alpha-Farm used is a cluster of eight DEC Alpha 3000/400 workstations located at the San Diego Supercomputer Center. The Alphas are connected via a dedicated GIGAswitch.

cation.

Figure 1 shows the difference between the delay imposed on computation on node m_1 by one application communicating from node m_1 to node m_2 when different bandwidths are available. It shows curves for execution of the SOR benchmark with different problem sizes (given by $N \times N$) subject to no contention, and subject to contention with 1.55 and 3.21 Mwords/second available. The communicating process is the same for both experiments. It is clear that when more bandwidth is available, the communicating process uses more CPU, and the delay it imposes on the SOR benchmark is larger. The bandwidth available was calculated with a ping-pong benchmark that transfers 1K messages with 1K words each to another machine and receives one word back.

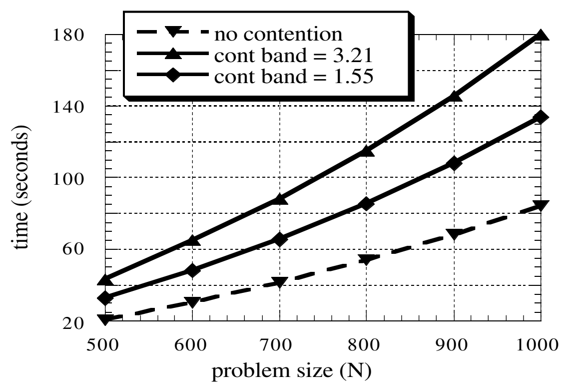

Figure 1: Time to execute the SOR benchmark on one node of the DEC Alpha-Farm in dedicated mode and competing for the CPU with one communication-bound process when different bandwidths are available.

In fact, the slowdown imposed by communication varies also with the communicating machines and the direction of the transferences. Note that the effects caused by the direction and target of the communication are reflected in bandwidth variation. Hence we focus on just the bandwidth measure for both computation and communication predictions.

3.1 Computation Slowdown in High-Performance Clusters

Computation on each node is affected by other applications executing on the same node, which may be computing or communicating with another node.

The delay imposed by applications computing on a node reflects the fact that CPU time is evenly split among the competing processes. (Recall that we assume that the node scheduling policy is round-robin.) The delay imposed by other applications communicating depends on the number of applications communicating, on the bandwidth available, and on the size of the messages being transferred. The impact of the message size is reflected by bandwidth variation. Hence we can approximate the slowdown imposed on a task executing on node a as:

$$sd_a = 1 + \sum_{i=1}^{p_a} (pp_{a,i} \times i) + \sum_{i=1}^{p_a} (pm_{a,i} \times delay_{comm}^{i,k}) , \tag{3}$$

where

- p_a = number of additional applications executing on node a,
- $pp_{a,i}$ = probability that i applications will compute on node a at the same time,
- $pm_{a,i}$ = probability that i applications on node a will try to communicate at the same time, and
- $delay_{comm}^{i,k}$ = delay imposed on computation by i communicating applications when the average bandwidth available is k Mwords/second.

In the slowdown calculation, the first summation accounts for other applications competing for CPU cycles, which are evenly split among all CPU-bound applications on the node. The second summation accounts for additional communicating applications. The values $pp_{a,i}$ and $pm_{a,i}$ are calculated based on the percentages of computation and communication associated with each application executing on node a.

The value $delay_{comm}^{i,k}$ is the average delay imposed on computation by i communication-intensive applications executing on the node. It is calculated as the average delay imposed on a CPU-bound application by i contention generators that transfer messages to another node when the bandwidth available is k Mwords/second. Since the delay imposed by receiving and sending applications is roughly the same, we use $delay_{comm}^{i,k}$ in both situations.

The value $delay_{comm}^{i,k}$ is system-dependent and does not change dynamically. Our experiments determined $delay_{comm}^{i,k}$ as a set of functions of k, each of which is indexed by a different value of i. The functions are determined just once for each platform and, therefore, their calculation does not impose any additional overhead on the scheduling process at run-time. Each function is determined with a two-piece linear regression on the numbers obtained by a benchmark. The benchmark measures the delay imposed on a CPU-bound application by i communication-bound applications that communicate with other nodes under decreasing bandwidth. At run-time, the value for k – the average bandwidth available on the links used by competing applications –

can be predicted with statistical methods (e.g. the Network Weather Service [3, 14]), and used to determine $delay_{comm}^{i,k}$.

Figure 2 shows the curves obtained for $delay_{comm}^{i,k}$, when $i = 1$ and $i = 2$. In low-bandwidth systems, $delay_{comm}^{i,k}$ may be approximated by just one curve or by just one value. In the first case, one curve is enough, because the delay does not vary with i. In the second case, described in Section 4.1, one value may be enough because the range of the delay imposed by communication is small.

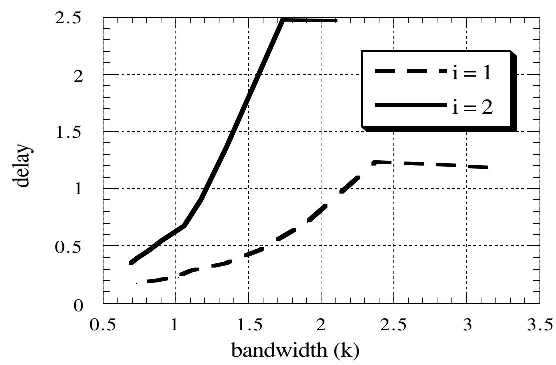

Figure 2: Each curve in the graph provides the value for the respective delay, which varies with the bandwidth (k).

The slowdown factor is calculated at run-time and, therefore, it is important to guarantee that its calculation is efficient so that it does not impose too much overhead on the scheduling process. Although the values $pp_{a,i}$ and $pm_{a,i}$ change with the load on the system and are obtained at run-time, they can be calculated using dynamic programming in time $O(p_a^2)$. Since p_a is generally small and the overall calculation of the slowdown takes $O(p_a^2)$ time, the overhead imposed by its calculation is negligible.

Figure 3 illustrates equation (3); it shows the modeled and actual times for executing the SOR benchmark on m_1 in dedicated and non-dedicated modes, parameterized by problem size ($N \times N$). In this experiment, one more application is executing on the same node. It alternates computation and communication cycles and communicates with another node 45% of the time. The bandwidth available in the system under contention is 1.79 Mwords/second, and the slowdown imposed by the application is 1.56. In this example, our predictions were within an average error of 3% of the actual measurements.

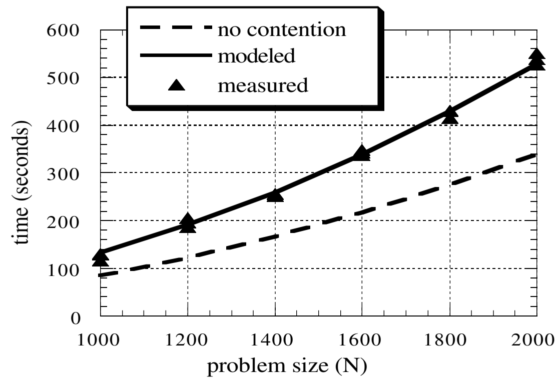

Figure 3: Time to execute the SOR benchmark on one node of the DEC Alpha-Farm in dedicated mode and competing with one application.

To validate our model, we ran a serial version of the benchmarks with diverse configurations of one, two and three instances of the contention generator. These configurations were defined by the contention generator parameters. Table 1 shows experiments with a serial version of the Multigrid benchmark and two competing applications, and Table 2 shows experiments with a serial version of the Matrix Multiply benchmark. Each row in the tables corresponds to one execution of the benchmark with five different problem sizes. As the tables show, we vary the burst size used by the competing application and the average bandwidth available on the links used by the competing applications.

Table 1: Experiments with Multigrid

Burst Sizes (number of messages / message size in words) per Competing Application	Average Bandwidth (Mwords / second)	Average Error
10 / 1000, 10 / 1000	2.75	2%
1000 / 1000, 1000 / 1000	2.75	1%
5000 / 1000, 5000 / 1000	2.75	1%
100 / 1000, 100 / 1000	2.12	11%
1000 / 1000, 1000 / 1000	2.12	9%
5000 / 1000, 5000 / 1000	2.12	4%

Table 2: Experiments with Matrix Multiply

Burst size(s) (number of messages / message size in words) per Competing Application	Average Bandwidth (Mwords / second)	Average error
4000 / 4000	1.51	5%
2000 / 2000, 2000 / 2000	3.18	1%
2000 / 2000, 3000 / 3000 4000 / 4000	3.89	9%
2000 / 1000, 2000 / 1000 2000 / 1000	2.45	6%

3.2 Communication Slowdown in High-Performance Clusters

In high-performance clusters, when the nodes are connected by switches and contention for the interconnection network is negligible, communication costs between two nodes are affected primarily by other applications on the two nodes. These applications may be computing, or communicating within the source and destination nodes or with other nodes.

Our experiments show that various factors affect communication costs. The number of communicating and computing applications executing on the source and destination nodes, the size of the messages being transferred by the communicating applications, the bandwidth available on the links used, and the destination and direction of the communication, all affect the amount of delay.

Figure 4 illustrates the fact that the bandwidth available in the links used by competing applications affects the delay imposed by these applications on communication costs. Suppose we want to measure communication costs from node m_1 to node m_2. If node m_2 hosts an application that is communicating with node m_3, the bandwidth available on this link (between nodes m_2 and m_3) will affect the amount of CPU required from nodes m_2 and m_3 to perform this communication and, consequently, communication costs between nodes m_1 and m_2.

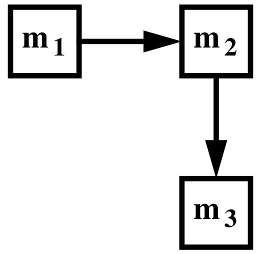

Figure 4: Communication activity among 3 nodes in a dedicated cluster.

Figure 5 shows the difference between the delay imposed on communication from node m_1 to node m_2 by one competing application that transfers data from node m_2 to node m_3 under different bandwidths. The graph shows the time to transfer bursts of 1000 messages in three different situations: when there is no contention on the nodes; when the competing application is also executing and the bandwidth available between nodes m_2 and m_3 is 2.23 Mwords/second, and when the competing application is also executing but the bandwidth available between nodes m_2 and m_3 is 1.20 Mwords/second. The competing application is the same for both experiments.

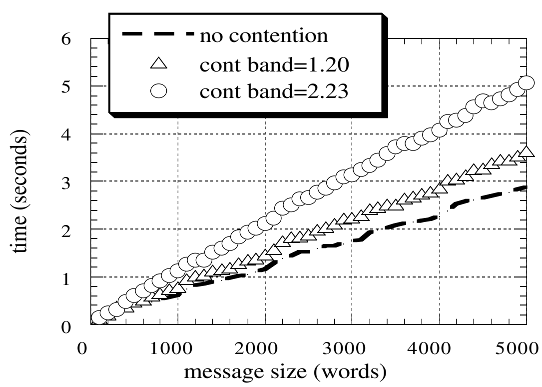

Figure 5: Time to transfer bursts with 1000 messages between two nodes of the DEC Alpha-Farm in dedicated mode and competing with a communication-bound process that transfers messages through a link where different bandwidths are available.

A prediction of the bandwidth available on the links used by competing applications is required at run-time so that the delay imposed on communication can be calculated. However, the bandwidth available between nodes *a* and *b* can be used in the calculation of the slowdown on communication as follows:

$$sd_{a \to b} = \frac{ib_{a \to b}}{cb_{a \to b}}, \tag{4}$$

where

- $ib_{a \to b}$ = initial bandwidth, or bandwidth available when $(Y_a \to X_b)_{\text{ded}}$ was calculated and
- $cb_{a \to b}$ = current bandwidth, or bandwidth available at run-time.

Equation (4) reflects the delays imposed by both computation and communication activities on communication costs. Although these activities do not appear in the formula, they are embedded in the current bandwidth measure, $cb_{a \to b}$, which accounts for competition for both CPU and communication link.

The value for $ib_{a \to b}$ can be calculated with a benchmark whenever $(Y_a \to X_b)_{\text{ded}}$ is calculated. The benchmark sends 1024 messages with 1024 words each from one node to another and waits for the answer.

The value for $cb_{a \to b}$ can be obtained at run-time with statistical methods employed by performance prediction tools, such as the Network Weather Service [3, 14]. In our experiments, since the contention in the cluster is generated by a synthetic load, we determine the value for $cb_{a \to b}$ by measuring the bandwidth at run-time with the same benchmark used to calculate $ib_{a \to b}$.

Figure 6 illustrates communication formula (4). It shows the time for transferring bursts of 1000 equal-sized messages between two nodes in dedicated time ($ib_{a \to b}$ = 6.21 Mwords/second), and modeled and actual times for transferring the same messages between the two nodes when $cb_{a \to b}$ = 3.67 Mwords/second. The error in this experiment was within 9% on the average.

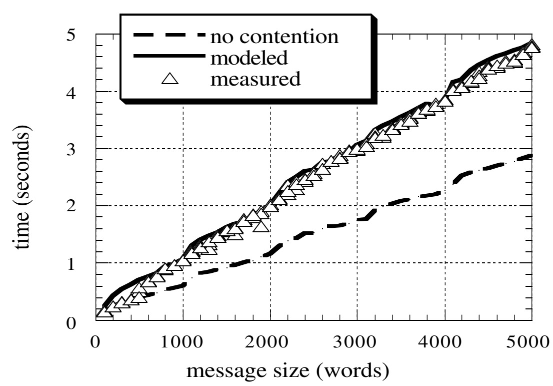

Figure 6: Time to transfer bursts with 1000 messages each between two nodes, in dedicated mode and under decreased bandwidth.

Figure 7 also illustrates communication formula (4). It shows the time for transferring bursts of 2000 equal-sized messages between two nodes in dedicated time ($ib_{a \to b}$ = 6.21 Mwords/second), and modeled and actual times for transferring the same messages between the two nodes when $cb_{a \to b}$ = 1.95 Mwords/second. The error in this experiment was within 13% on the average.

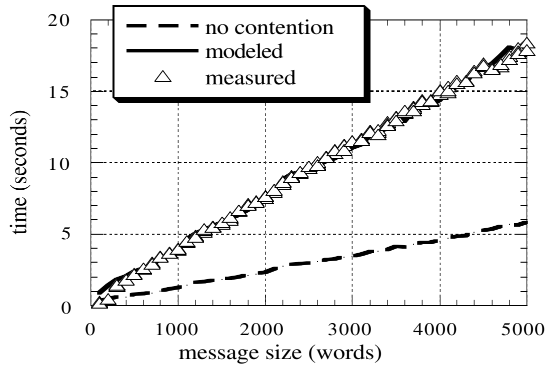

Figure 7: Time to transfer bursts with 2000 messages each between two nodes, in dedicated mode and under decreased bandwidth.

To validate the model shown in formula (4), we ran a ping-pong benchmark (which transfers bursts of same-sized messages) with diverse configurations of one, two and three instances of the contention generator. These configurations were defined by the contention generator parameters. Table 3 presents a representative set of the experiments performed to verify the model. Each row of the table corresponds to three executions of the ping-pong benchmark for each message size, which grows in steps of 100 words. As the table shows, we vary bandwidth available and burst size of the ping-pong benchmark.

According to the results shown in Table 3, the error increases when the burst size is small and the available bandwidth is low. This happens because, in this high-bandwidth platform, short bursts are not sensitive to bandwidth variations. This is clear in Table 4, which shows the slowdown $sd_{a \to b} = (ib_{a \to b}) / (cb_{a \to b})$ produced with a benchmark that measures the bandwidth using a specific burst size. A benchmark that calculates the bandwidth using a tailored burst size will overcome this problem.

Table 3: Experiments with Communication

Burst sizes (number of messages / message size in words)	Average Bandwidth (Mwords / second)	Average Error
100 / 100 – 1000	1.19	368%
100 / 100 – 1000	2.32	151%
100 / 100 – 1000	3.44	78%
100 / 1000 – 3000	1.19	16%
100 / 1000 – 3000	2.32	51%
100 / 1000 – 3000	3.44	23%
1000 / 100 – 1000	1.19	81%
1000 / 100 – 1000	2.32	42%
1000 / 100 – 1000	3.44	33%
1000 / 1000 – 3000	1.19	14%
1000 / 1000 – 3000	2.32	4%
1000 / 1000 – 3000	3.44	2%
5000 / 100 – 1000	1.19	57%
5000 / 100 – 1000	2.32	20%
5000 / 100 – 1000	3.44	16%
5000 / 1000 – 3000	1.19	10%
5000 / 1000 – 3000	2.32	3%
5000 / 1000 – 3000	3.44	2%

Table 4: Experiments with Bandwidth Benchmarks

Burst size (number of messages / message size in words) used by the Benchmark	$sd_{a \to b}$
100 / 100	1.08
100 / 500	0.96
1000 / 500	3.99
1024 / 1024	3.97
5000 / 500	3.96

4 Contention Effects in Workstation Networks

The slowdown factors calculated for clusters were developed for high-performance clusters, in which the network connecting the nodes of the cluster was dedicated to the nodes and non-negligible bandwidth variations were caused exclusively by applications on the cluster. In this section, we extend the model to reflect a more general scenario in which machines may belong to different networks and are connected by non-dedicated network links. The basic differences between this environment and the previous one are:

- Less bandwidth is typically available in workstation networks.
- In workstation networks, machines may not be all equal.
- Links connecting the machines in a workstation network may not have the same bandwidth.
- Bandwidth between machines in a workstation network may be affected by applications executing on other machines that share network links with the machines in the network.

It is important to note that, in this platform, we use the term *communication link* to refer to a connection between two nodes that may, or may not, be directly connected by a physical network link.

We consider each node in the workstation network as an individual machine. In this section, we calculate the slowdown for computation and communication costs, sd_a and $sd_{a \to b}$ respectively, where a and b are two nodes in the network.

We demonstrate our results using the four nodes of a workstation network consisting of two DEC Alphas[4] (located at the Computer Systems Laboratory at UCSD) connected by Ethernet and two IBM RS-6000s (located at the Parallel Computation Laboratory at UCSD) also connected by Ethernet. These workstations are part of two non-dedicated networks.

4.1 Computation Slowdown in Workstation Networks

Computation on each node is affected by other applications executing on the same node, which may be computing or communicating with another node. The delay imposed by applications computing on the node reflects the fact that CPU time is evenly split among the competing processes. The delay imposed on computation by communicating applications depends on the variation of the bandwidth on the links used. However, since the bandwidth is typically lower in this case, the range of its variation is small, as is the range of the variation on the delay. For this reason, an average delay can be used to approximate the delay imposed by communication on computation. Consequently, the slowdown imposed on computation on node a can be modeled by

[4] The DEC Alphas used are DEC Alpha 3000/800 workstations.

$$sd_a = 1 + \sum_{i=1}^{p_a} (pp_{a,i} \times i) + \sum_{i=1}^{p_a} (pm_{a,i} \times delay_{comm}) , \tag{5}$$

where

- p_a = number of extra applications executing on the node,
- $pp_{a,i}$ = probability that i applications will compute on node a at the same time,
- $pm_{a,i}$ = probability that i applications on node a will try to communicate at the same time, and
- $delay_{comm}$ = delay imposed on computation by communication activity.

In the slowdown calculation, the first summation accounts for other applications competing for CPU cycles, which are evenly split among CPU-bound applications on the node. The second summation accounts for computation required to support the communication of other applications. Both $pp_{a,i}$ and $pm_{a,i}$ are calculated based on the percentages of computation and communication associated with each application executing on node a.

The value $delay_{comm}$ is the average delay imposed on computation by communicating applications executing on the node. It is the average of two delays. The first is the maximum delay imposed by communication on computation. This can be calculated by measuring the delay imposed on a CPU-bound application by a communication-intensive application that communicates with the "closest" machine in the workstation network transferring messages with 1000 words. The closest machine is the one with which the node can communicate with highest bandwidth (generally the closest machine is one located on the same physical network, e.g., on the same Ethernet). The second is the minimum delay imposed by communication on computation, which generally tends to 0 because the delay decreases with the bandwidth available.

The slowdown factor is calculated at run-time and, therefore, it is important to guarantee that its calculation is efficient so that it does not impose too much overhead on the scheduling process. The value $delay_{comm}$ is system-dependent and does not change dynamically. It is calculated just once for each platform and, therefore, its calculation does not impose any additional overhead on the scheduling process at run-time. Although the values $pp_{a,i}$ and $pm_{a,i}$ change with the load on the system and are obtained at run-time, they can be calculated by dynamic programming in time $O(p_a^2)$. Since p_a is small and the overall calculation of the slowdown takes $O(p_a^2)$ time, the overhead imposed by its calculation is negligible.

Figure 8 shows the modeled and measured times for executing the SOR benchmark on a DEC

Alpha in dedicated and non-dedicated modes, parameterized by problem size (which is $N \times N$). In this experiment, two more applications are executing on the machine. They alternate computation and communication cycles and communicate with another machine 24% of the time. In this example, our predictions were within an average error of 6% of the actual measurements.

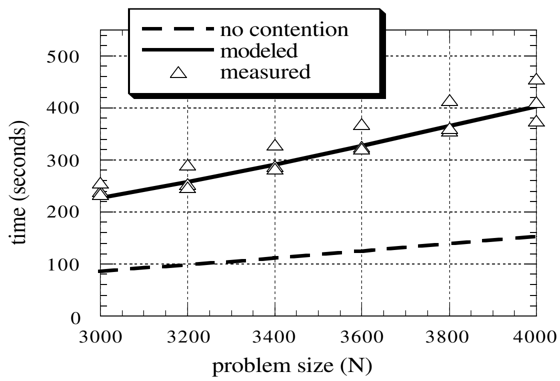

Figure 8: Time to execute the SOR algorithm on a DEC Alpha, in dedicated and non-dedicated mode, competing with two extra applications. The contending applications communicate 24% of the time and compute the remaining time.

To validate the model described in equation (5), we ran a serial version of the SOR benchmark and other applications with diverse configurations of one, two and three instances of the contention generator. These configurations were defined by the contention generator parameters. Table 5 presents experiments performed with a serial version of the SOR benchmark when there are competing applications on the node. Each row of the table corresponds to one execution of the benchmark (on a DEC Alpha) with six different problem sizes. As the table shows, we vary the computation/communication ratio of the competing applications and the burst size used by the competing applications. The bandwidth available varies with the competing applications executing.

Table 5: Experiments with Computation

Amount(s) of Communication per Competing Application	Burst Size(s) (number of messages / message size in words) per Competing Application	Average Error
24%	1000 / 2000	5%
49%	2000 / 1000	9%
58%	1000 / 3000	9%
44%	5000 / 1000	7%
15%	5000 / 100	8%
14%	100 / 5000	3%
49%, 49%	2000 / 1000, 2000 / 1000	6%
15%, 14%	5000 / 100, 100 / 5000	8%
38%, 15%	1000 / 2000, 5000 / 100	13%
48%, 48%	1000 / 2000, 1000 / 2000	3%
48%, 21%	1000 / 2000, 5000 / 100	18%
58%, 32%	1000 / 3000, 1000 / 1000	15%
49%, 49%, 49%	2000 / 1000, 2000 / 1000, 2000 / 1000	6%
58%, 32%, 49%	1000 / 3000, 1000 / 1000, 2000 / 1000	22%

4.2 Communication Slowdown in Workstation Networks

Our experiments show that various factors affect communication costs. The number of computing and communicating applications executing on the two nodes, the size of the messages being transferred by communicating applications, the bandwidth available on the links used, and the destination and direction of the communication, all affect the delay imposed on communication. As in the cluster case, bandwidth variation reflects all these factors and can be used in a simple model. Following the analysis for the high-performance clusters, slowdown on communication can be modeled by

$$sd_{a \to b} = \frac{ib_{a \to b}}{cb_{a \to b}} \tag{6}$$

as shown in Section 3.2.

Figure 9 illustrates communication formula (6). It shows the time for transferring bursts of 1000 equal-sized messages between the two IBM RS-6000s (located in the same subnet) when

$ib_{a \to b}$ = 0.91 Mwords/second, and modeled and actual times for transferring the same bursts when there are additional applications alternating computation and communication cycles on the nodes. The value for $cb_{a \to b}$ is 0.33 Mwords/second, $sd_{a \to b}$ = 2.76, and the average error for modeled versus actual communication cost was within 7%. The peak obtained by transferring messages with 3100 words reflects an inefficiency of the IBM RS-6000 system in dealing with this specific message size.

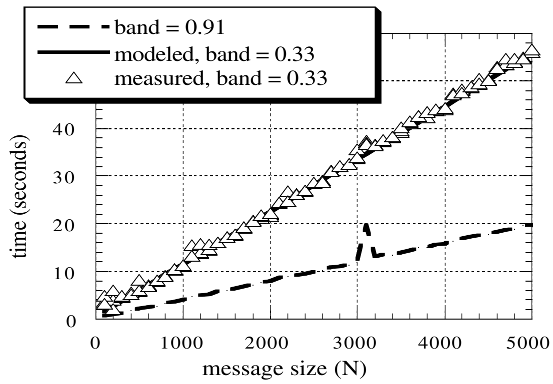

Figure 9: Time to transfer bursts with 1000 messages between machines in the same subnet.

Figure 10 also illustrates communication formula (6). It shows the time for transferring bursts of 2000 equal-sized messages between two nodes in different subnets when $cb_{a \to b}$ = 0.48 Mwords/second, and modeled and actual times for transferring the same bursts when there are additional applications alternating computation and communication cycles on the nodes. The value for $cb_{a \to b}$ is 0.28 Mwords/second, $sd_{a \to b}$ = 1.71, and the average error for communication cost modeled with slowdown versus actual communication cost was within 7%.

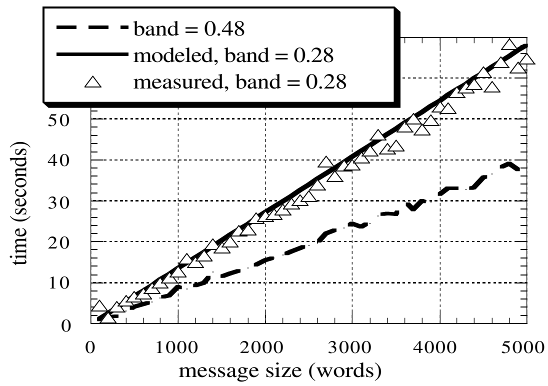

Figure 10: Time to transfer bursts with 1000 messages between machines in the different subnets.

To validate our model of communication slowdown, we ran a ping-pong benchmark (which transfers bursts of same-sized messages) with diverse configurations of one, two and three instances of the contention generator. These configurations were defined by the contention generator parameters. Table 6 presents a representative set of the experiments performed with the ping-pong benchmark. Each row of the table corresponds to three executions of the ping-pong benchmark for each message size, which grows in steps of 100 words. As the table shows, we vary the bandwidth available and burst size of the ping-pong benchmark.

Corroborating the results showed in Section 3.2, the error increases when the burst size is small. The error is smaller in this case than it is in the high-performance cluster case because the bandwidth is lower and the variation in its range is shorter. Also, the transference of short bursts, in this environment, is more sensitive to bandwidth variation than the transference of large bursts. This is clear in Table 7, which shows the slowdown $sd_{a \to b} = (ib_{a \to b}) / (cb_{a \to b})$ produced with a benchmark that measures the bandwidth using a specific burst size. As for high-performance clusters, a benchmark that calculates the bandwidth using a tailored burst size overcomes this problem.

Table 6: Experiments with Communication

Burst Sizes (number of messages / message size in words)	Average Bandwidth (Mwords / second)	Average Error
10 / 100 – 1000	0.48	23%
10 / 100 – 1000	0.68	43%
10 / 1000 – 3000	0.48	5%
10 / 1000 – 3000	0.68	43%
1000 / 100 – 1000	0.48	19%
1000 / 100 – 1000	0.68	8%
1000 / 1000 – 3000	0.48	2%
1000 / 1000 – 3000	0.68	7%
5000 / 100 – 1000	0.48	15%
5000 / 100 – 1000	0.68	8%
5000 / 1000 – 3000	0.48	1%
5000 / 1000 – 3000	0.68	6%

Table 7: Experiments with Bandwidth Benchmarks

Burst size (number of messages / message size in words) used by the Benchmark	$sd_{a \to b}$
10 / 100	5.09
10 / 500	2.26
1000 / 500	2.06
5000 / 500	2.05

5 Evaluation of the Model

The models developed for slowdown on computation were based on the relative amounts of computation and communication. We assume that the time to compute is longer than the computation/communication cycles of the competing applications and will be affected by both computation and communication activities. If the targeted computation is fast in comparison with the duration of these cycles, i.e., the time to compute is close to the duration of one computation/communication cycle, the accuracy of the model decreases. The error may also increase if the applications present a non-random behavior, e.g., if the computation and communication cycles of the competing applications are either totally synchronized or totally non-synchronized with one another.

Figure 11 and Figure 12 illustrate the difference in accuracy obtained in the prediction of the time to execute the SOR benchmark serially on a DEC Alpha in two situations. In both situations, the SOR competes for the CPU with one application that alternates computation with communication. In Figure 11, the time to execute one computation/communication cycle of the competing application was 24.2 seconds. In this case, the time to execute the algorithm was longer than one computation/communication cycle of the competing application, and the average error was 5%. In Figure 12, the time to execute one computation/communication cycle of the competing application was 288.0 seconds. In this case, the time to execute the same algorithm was shorter than one computation/communication cycle of the competing application, and the variation of the actual times to execute the algorithm causes the average error to be within 17%.

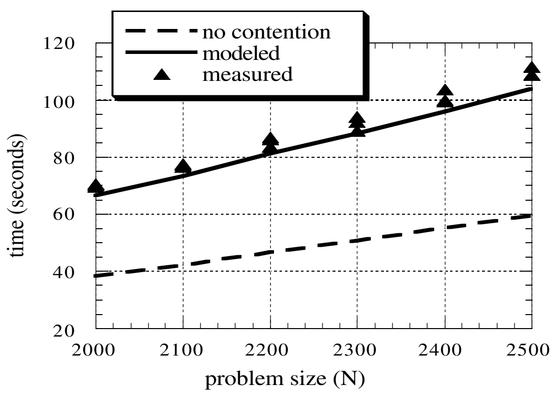

Figure 11: Time to execute the SOR benchmark in dedicated mode and competing with one application that has a 24.2-second computation/communication cycle.

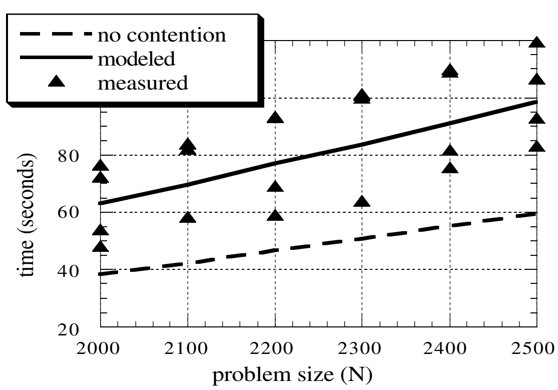

Figure 12: Time to execute the SOR benchmark in dedicated mode and competing with one application that has a 288.0-second computation/communication cycle.

6 Summary and Future Work

Most applications share the resources of networked environments with other applications. Since system load can vary dramatically, allocation strategies which assume that resources have a constant availability and/or capability are unlikely to promote performance-efficient allocations in practice. In fact, it is critical to provide a realistic model of the effects of contention on application performance in order to best allocate application tasks to machines.

In this paper, we have presented a model which predicts contention effects in representative networked platforms. The model provides a basis for predicting realistic communication and computation costs. Note that the variation in times to execute applications on production systems is typically high, and therefore makes it difficult for a contention model to be accurate. For this reason, our objective has been to obtain accuracy on an average basis. The experiments executed thus far have shown that we have achieved our goal.

We are continuing to investigate causes for contention. We are currently extending our model to include memory constraints, as well as priority issues. We also plan on characterizing the setting in which contending applications execute for only part of the execution of a given application. Since system load may vary during the execution of an application, the slowdown factors should be recalculated when the job mix changes, and task migration should be considered.

Finally, the slowdown factors developed for these platforms can be used for general networked systems. We believe that an accurate contention model is a fundamental part of a performance-efficient allocation strategy for networked systems and, ultimately, essential to the use of these systems as platforms for parallel and/or heterogeneous applications.

References

[1] M. Atallah, C. Black, D. Marinescu, H. Siegel, and T. Casavant, "Models and Algorithms for Coscheduling Compute-Intensive Tasks on a Network of Workstations", Journal of Parallel and Distributed Computing, vol. 16, pp. 319–327, 1992.

[2] A. Beguelin, J. Dongarra, G. Geist, R. Manchek, and V. Sunderam, "Graphical Development Tools for Network-Based Concurrent Supercomputing", Proceedings of Supercomputing 91, pp. 435–444.

[3] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-Level Scheduling on Distributed Heterogeneous Networks", in *Proceedings of Supercomputing'96*, November 1996.

[4] A. Bricker, M. Litzkow, and M. Livny, "Condor Technical Summary", Technical Report #1069, University of Wisconsin, Computer Science Department, May 1992.

[5] W. L. Briggs, "A Multigrid Tutorial", Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.

[6] H. Dietz, W. Cohen, and B. Grant, “Would you run it here...or there? (AHS: Automatic Heterogeneous Supercomputing)”, Proceedings of the International Conference on Parallel Processing, vol. II, pp.217–221, August 1993.

[7] A. C. Dusseau, R. H. Arpaci, and D. E. Culler, “Effective Distributed Scheduling of Parallel Workloads, in *Proceedings of ACM SIGMETRICS’96*, pp. 25–36, May 1996.

[8] S. M. Figueira and F. Berman, “Modeling the Effects of Contention on the Performance of Heterogeneous Applications,” in *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, August 1996.

[9] S. J. Fink, S. B. Baden, and S. R. Kohn, “Flexible Communication Mechanisms for Dynamic Structured Applications”, in *Proceedings of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems*, Santa Barbara, CA, August 1996.

[10] S. Leutenegger and X. Sun, “Distributed Computing Feasibility in a Non-Dedicated Homogeneous Distributed System”, NASA - ICASE Technical Report 93-65, September 1993.

[11] Matrix Multiply, http://HTTP.CS.Berkeley.EDU/~demmel/cs267.

[12] Message-Passing Interface Forum, “MPI: A Message-Passing Interface Standard”, University of Tennessee, Knoxville, TN, June 1995.

[13] V. Sunderam, “PVM: A Framework for Parallel Distributed Computing”, Concurrency: Practice and Experience, vol. 2, n. 4, pp. 315–339, December 1990.

[14] R. Wolski, “Dynamically Forecasting Network Performance Using the Network Weather Service,” UCSD CS Technical Report #CS96-494.

[15] X. Zhang and Y. Yan, “A Framework of Performance Prediction of Parallel Computing on Non-dedicated Heterogeneous Networks of Workstations”, Proceedings of 1995 International Conference of Parallel Processing, vol. I, pp. 163–167, 1995.