Limitations of Cycle Stealing for Parallel Processing on a Network of Homogeneous Workstations*

Scott T. Leutenegger
Math and Computer Science Department
University of Denver
2360 S. Gaylord Street
Denver, CO 80208-0189, USA
leut@cs.du.edu

Xian-He Sun
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020
sun@bit.csc.lsu.edu

Abstract

The low cost and availability of clusters of workstations have lead researchers to re-explore distributed computing using independent workstations. This approach may provide better cost/performance than tightly coupled multiprocessors. In practice, this approach often utilizes wasted cycles to run parallel jobs. In this paper we address the feasibility of such a non-dedicated parallel processing environment assuming workstation processes have preemptive priority over parallel tasks. We develop an analytical model to predict parallel job response times. Our model provides insight into how significantly workstation owner interference degrades parallel program performance. It forms the foundation for efficient, feasible task partioning and scheduling in a non-dedicated network environment. A new term **task ratio**, which relates the parallel task demand to the mean service demand of non parallel workstation processes, is introduced. We propose that **task ratio** is a useful metric for determining how a parallel applications should be partitioned and scheduled in order to make efficient use of a non-dedicated distributed system.

*The research was supported in part by the National Aeronautics and Space Administration under NASA contract NAS1-19480 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

1 Introduction

Most early parallel processing research focused on using distributed systems to speedup computations. The basic approach was to utilize many computers connected via a local area network (LAN) to execute a parallel job. This environment has been referred as *distributed computing, network computing* and *distributed network computing.* With the advent of multiprocessor architectures the majority of the focus shifted from distributed computing to multiprocessing, the major distinction being the tightly coupled architecture allowing more finely grained parallelism.

Recently, a significant portion of the parallel community has returned to the network processing approach. Network Of Workstations (NOW) has been recognized as the primary computing infrastructure for science and engineering [1]. Several commercial and noncommercial tools have been developed to support network computing. Widely used tools include the *Parallel Virtual Machine* (PVM) software, the *P4* system, *Express*, and the *message passing interface* (MPI) [15, 3, 7, 8]. A major driving force behind the reevaluation of distributed computing is the high cost of parallel computers. Using a group of workstations connected via a LAN may provide better cost/performance, or may be the only way to achieve high performance within budget constraints for some organizations. Another factor in favor of distributed computing is the availability of many lightly loaded workstations. Exploiting these idle cycles for sequential jobs has been previously studied [16, 5, 10]. These otherwise wasted idle cycles can be used by a distributed computation to provided speedups and/or to solve large problems that otherwise could not be tackled.

It is clear that many problems are amenable to the distributed computing approach [6]. However, for some applications, the inherent synchronization requirements, communication/computation ratio, and the granularity of parallelism may limit the obtained performance. Even for the "good" applications, a tacit assumption of the expected high performance is that a system of *dedicated* workstations are used, which may not be true in practice. In this paper we study the performance of distributed computing in a *non-dedicated* system assuming workstation owner processes have preemptive priority over parallel tasks

We assume the parallel application considered belongs to the class of programs that can run efficiently in a dedicated distributed computing environment. We do not consider the effects of synchronization, communication, or granularity of parallelism. Given the program executes efficiently in a dedicated system, we wish to determine whether we can achieve good performance in a non-dedicated system.

By considering embarrassingly parallel applications for no parallel overheads we provide an upper bound on expected performance. One of our main conclusions is that in order to achieve good speedups on a non-dedicated parallel environment without task migration, the amount of work in the parallel job must be significantly large to absorb workstation owner interferences. Our optimistic assumptions imply that our conclusions are even stronger than stated.

One factor that must be considered in a non-dedicated system is how intrusive the parallel programs are to the owners of the workstations and vice versa. The priority of the parallel tasks relative to the priority of processes initiated by the owner of the workstation can have a significant impact on the performance of both the parallel job and the owner's serial jobs. We assume that a workstation owner is not tolerant of other people using their workstation, and hence surmise the most appropriate model of such a system is to assume workstation owner processes have preemptive priority over processes belonging to a parallel job. Hence, use of the workstation will interfere with parallel program performance. The major goal of this paper is to provide insight into how significantly workstation owner interference degrades parallel program performance. We seek to answer the question, "When is distributed network computing in a non-dedicated environment where workstation owner processes have preemptive priority over parallel tasks a viable approach?" Note, an alternative approach is to migrate parallel tasks when workstation owners start using the machine. An trace-driven simulation study of this approach is found in [2]. In fact, we show that unless parallel jobs are of sufficient size a non-dedicated environment without task migration will not provide acceptable speedups, and hence task migration is necessary.

An analytical model is developed to provide qualitative and quantitative performance predictions under the non-dedicated assumption. We also use simulation to consider workstation workloads with high variance. We expect that a large variance in workstation process demand should further reduce the utility of parallel computing when workstation owner processes have priority over tasks belonging to parallel jobs.

The new term *task ratio* is introduced along with new metrics that incorporate the utilization of workstations by owner processes. We find that the task ratio plays an important role in the overall performance, possibly as important as the communication/computation ratio in a dedicated system. The analytical model provides the relationships between the identified parameters and shows how these parameters influence the overall response time. It establishs a guideline of task partioning and scheduling in a non-dedicated network environment.

In addition to our analysis, a hypothetical local computation [13] problem is implemented with PVM on systems with 1 to 12 homogeneous workstations. These initial experimental results confirm the qualitative results from the analytical model.

This paper is organized as follows. In Section 2 we present the analytical model and introduce new parameters and metrics for non-dedicated distributed computing. The results from our analysis are presented in Section 3. Experimental results with PVM on 12 homogeneous workstations are presented in Section 4, sensitivity to the variation in workstation owner demands are presented in section 5, and our conclusions are in Section 6.

Table 1: Notational Definitions

$\mathcal{J}$	Total demand of the parallel job.
$\mathcal{W}$	Number of workstations in the system.
$\mathcal{T}$	Demand of one parallel task = $\mathcal{J}$ / $\mathcal{W}$.
$\mathcal{O}$	Time a owner process uses the workstation.
$\mathcal{U}$	Utilization of a workstation by owner.
P	Probability of the owner requesting the processor during a given time step.
E_t	Mean expected task completion time.
E_j	Mean expected job completion time.

2 Model Description, Analysis and Simulation

In this section we describe our system model, our analysis technique, and simulation model. We make simplifying assumptions that favor the distributed computing approach. In particular, we assume a parallel job is composed of $\mathcal{W}$ tasks (one per workstation), and the computation is perfectly balanced among these tasks. In addition, the parallel job is composed of one single parallel phase with no communication or synchronization requirements other than the final synchronization which occurs when all of the tasks have completed. Hence, we are assuming perfect parallelism of the problem. This model is simplistic, but provides the best case scenario for a distributed computing environment. In addition, by not incorporating communication or synchronization requirements into the model we are able to attribute all degradation of parallel program performance to workstation process interference. Since our assumptions are always optimistic, the model predictions provide an upper bound on expected performance.

We assume there are $\mathcal{W}$ homogeneous workstations in the system and that there is one owner per workstation. Workstation owners are in a continuous cycle of thinking (idle time) and then use time. We assume there is one parallel job being executed on the system at a time. In Table 1 we define our notation used through out the paper. The demand of a job is the total computing cycles (time) needed for the job.

We first describe our deterministic model which assumes workstation owner processes demands are deterministic. This model provides an upper bound on expected performance since all assumptions are optimistic. We next describe our simulation model which allows for hyper-exponentially distributed workstation owner process demands. The simulation model is used to demonstrate the sensitivity of expected performance to workstation owner process variance.

2.1 Deterministic Model Description

Our model is a discrete time model. We assume a geometric distribution with mean $\frac{1}{P}$ for the owner think time, i. e. at each time unit the owner requests the processor with probability P. When an owner process starts execution an executing parallel task is suspended and the owner process is immediately started. The owner process executes for $\mathcal{O}$ units. Once the owner processes completes execution, the parallel task restarts execution and is guaranteed to complete at least one unit of work before the owner may issue another process requesting the processor.

The model guarantees the parallel task will complete in at most $\mathcal{T}+(\mathcal{T}\times\mathcal{O})$ units. Task execution time at a single workstation is thus the sum of task demand plus the time to complete any owner processes that occur during the tasks tenure in the system, i. e.

$$task\ time = \mathcal{T} + (n \times \mathcal{O}), \tag{1}$$

where n equals the number of owner process requests. The owner process can make a request after each unit of time the parallel task uses the processor, hence the number of owner requests is binomially distributed:

$$Bin(\mathcal{T}, n, P) = \binom{\mathcal{T}}{n} P^n \ (1 \ - \ P)^{\mathcal{T}-n}. \tag{2}$$

Thus, expected task execution time is equal to

$$E_t = \mathcal{T} + \sum_{i=0}^{\mathcal{T}} \mathcal{O} \cdot i \cdot Bin(\mathcal{T}, i, P). \tag{3}$$

The job execution time is the time until the last of the parallel tasks completes execution. Thus, job completion time is at least $\mathcal{T}$ units and at most $\mathcal{T}+(\mathcal{T}\times\mathcal{O})$ units. We first derive the probability that job execution time equals i and then from these probabilities get the expectation.

Let S[n] equal the probability that an individual task is interrupted by at most n owner processes.

$$S[n] = \sum_{i=0}^{n} Bin(\mathcal{T}, i, P). \tag{4}$$

Let C[W,n] equal the probability that all parallel tasks are interrupted by at most n owner processes. By independence,

$$C[W, n] = (S[n])^W. \tag{5}$$

Let Max[W,n] equal the probability that the maximum number of owner process interferences over all the parallel tasks is equal to n.

$$Max[W, n] = C[W, n] - C[W, n-1]. \tag{6}$$

Using these functions, expected job execution time is calculated as:

$$E_j = \mathcal{T} + \sum_{i=0}^{\mathcal{T}} \mathcal{O} \cdot i \cdot Max[W, i]. \tag{7}$$

Owner utilization ($\mathcal{U}$) can be calculated as:

$$\mathcal{U} = \frac{\mathcal{O}}{\mathcal{O} + 1/P} \tag{8}$$

For the purposes of analysis we were forced to make some simplifying assumptions. Our model makes assumptions that favor the distributed computing approach, hence the model provides a lower bound on expected response time. In particular, the model is optimistic with regards to the three following points:

- We assume parallel task times are deterministic. Although this is one of the goals of parallel algorithm design, in practice there is often some imbalance of load.
- Variance of owner process service demands. We have assumed a deterministic owner process service demand when in fact typical processes experience a much larger variance [11]. Assuming a distribution with more variance could cause some parallel tasks to be delayed much longer than $\mathcal{T} + (\mathcal{T} \times \mathcal{O})$.
- Guaranteeing the parallel task at least one unit of execution between requests. In a real system owner processes may be reissued in less time, thus parallel tasks could be delayed longer than $(\mathcal{T} \times \mathcal{O})$.

These assumptions together clearly show that our results are optimistic, and hence actual performance could be worse than predicted by our observations.

2.2 Simulation Description

We have simulated the system using the CSIM simulation language [12]. We first use the simulation to validate the coding of our analysis. We duplicated the experiment found in Figure 1 of this paper and the simulation results were identical to the analysis thus verifying the correctness of analysis code. We obtained confidence intervals of 1 percent or less at a 90 percent confidence level. Confidence intervals are calculated using batch means [9] with 20 batches per simulation run and a batch size of 1000 samples. We did not plot the results since they are indistinguishable from the figure.

We then extend our simulation to allow for workstation owner request times to be exponentially distributed, and workstation owner service times to be hyper-exponentially distributed. Parallel task demands remain deterministic to provide an upper bound on performance given the variance in the workstation owner processes. We obtained confidence intervals of 5 percent or less at a 90 percent confidence level. Confidence intervals for these simulations used 20 batches per simulation run and a batch size of 3000 samples.

3 Analysis Results

In this section we present the results from our analysis. All results in this section assume an owner process has preemptive priority over a parallel task and that owner processes demands are deterministic. We first present results for a fixed size problem, and then discuss the impact of scaling problem size with the number of workstations.

3.1 Fixed-Size Speedup

For a fixed-size job the desired goal of parallelizing the program is to achieve faster execution times, hence we use expected speedup as our primary metric. Since the standard definition of speedup does not take into consideration the cycles consumed by the (higher priority) owner processes, we also define the metric *weighted-speedup*. We also consider the metrics efficiency and *weighted-efficiency* to illustrate more concretely the achieved percent of optimal performance. Specifically, once again let $\mathcal{J}$ equal the total job demand, $\mathcal{W}$ equal the number of workstations, E_j equal the expected job completion time, and $\mathcal{U}$ equal the owner process utilization of the workstations. Then:

$$
\begin{aligned}
\text{Task Ratio} &= \frac{\mathcal{T}}{\mathcal{O}} \\
\text{Speedup} &= \frac{\mathcal{J}}{E_j} \\
\text{Weighted-Speedup} &= \frac{\mathcal{J}}{(1-\mathcal{U})\,E_j} \\
\text{Efficiency} &= \frac{\mathcal{J}/\mathcal{W}}{E_j} \\
\text{Weighted-Efficiency} &= \frac{\mathcal{J}/\mathcal{W}}{(1-\mathcal{U})\,E_j}
\end{aligned}
$$

The expected speedup and efficiency metrics are of interest if a user wishes to determine the benefit of parallelizing the job relative to running the program on a single dedicated machine. The weighted metrics incorporate utilization to clearly demonstrate how effectively the parallel program is able to use the idle system cycles. We focus primarily on the weighted metrics since they provide a better metric for determining how well the distributed computing approach can utilize idle cycles.

In Figure 1 we plot speedup versus the number of workstations for workstation utilizations of 1%, 5%, 10%, and 20% assuming a parallel job demand ($\mathcal{J}$) equal to 1000 units, and an owner processes demand ($\mathcal{O}$) equal to 10 units. For a given utilization we assume all workstations have the same owner process utilization. The top curve is the theoretical optimal speedup, i.e. unitary linear. The speedup curves are concave increasing, i.e. the benefit of adding more nodes decreases as nodes are added, despite ignoring overhead for parallelizing the program (synchronization, communication, non-balanced load, etc). At 100 nodes the speedup for a system with only 1% utilization is only 61% of the optimal speedup, for a 20% utilization the speedup is only 32.5% of the optimal speedup.

To present the efficiency of the system, i.e. how close to optimal speedups are achieved, we plot efficiency versus number of nodes in Figure 2.

In both of the preceding plots we compare the performance of the parallel program executed on a system of workstations with a given owner utilization to that of the same program executed on a single node with no owner utilization. To focus on how effective distributed computing utilizes wasted cycles we consider the weighted-speedup and weighted-efficiency metrics. In Figures 3 and 4 we plot weighted-speedup and weighted-efficiency versus the number of nodes for the same parameters as in Figures 1 and 2. Note that the weighted-efficiency is still only 61.5% (41%) for a utilization of 1% (20%). Hence, even once owner utilization is taken into consideration achieved performance is significantly worse than optimal.

One cause for the degradation of performance is that the probability of one of the workstations experiencing a transient period of high utilization increases as the number of nodes increases. Since the parallel job must wait for each task to complete execution, just one workstation experiencing a

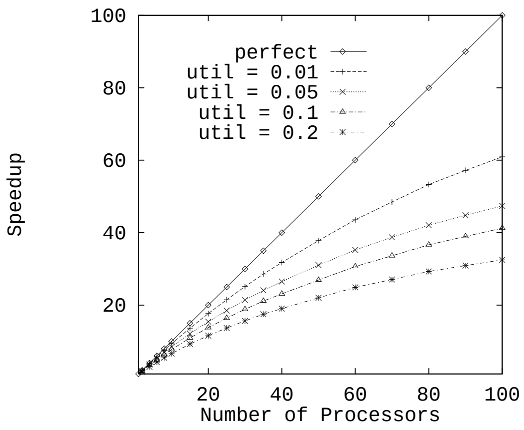

Figure 1: Speedup, J = 1000 units

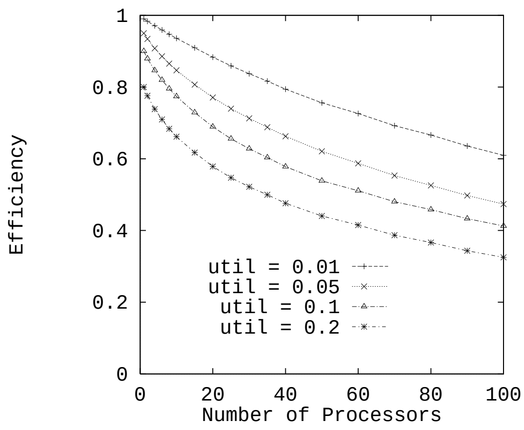

Figure 2: Efficiency, J = 1000 units

transient high utilization will slow down the entire computation, hence performance degrades as the number of workstations increases.

A second more subtle cause of performance degradation results from a decrease in the ratio of parallel task time to owner process task time (*task ratio*). To demonstrate this effect consider what happens if we increase the parallel job demand from 1K units to 10K units. In Figure 5 and 6 we plot the weighted-speedup and weighted-efficiencies for the same experiment as in Figures 3 and 4, except job demand equals 10K. The weighted-speedups and weighted-efficiencies for a job demand of 10K units are much higher than their counterparts in Figure 3 and 4. For $\mathcal{J}$ equal to 10K, $\mathcal{T}$ equals 100 units for a 100 workstation system, whereas $\mathcal{J}$ equal to 1K results in a $\mathcal{T}$ equal to 10 units for a 100 workstation system. Tasks of demand 10 units experience a proportionally larger delay by owner processes than tasks requiring 100 units.

To more clearly illustrate the point, we plot weighted-efficiency versus the task ratio for a system with 60 workstations in Figure 7. (The plot for weighted-speedups is identical except the y-axis is scaled from 0 to 60 instead of 0 to 1.) From the figure we conclude that in order to achieve acceptable efficiencys, and thus good speedups, we must ensure that the parallel task demand is sufficiently large relative to the average demand of owner processes, i. e. we must ensure a large task ratio.

In the previous experiment we fixed the number of workstations equal to 60. In Figure 8 we plot the weighted-efficiency versus task ratio for various system sizes for an owner utilization of 10%. Sensitivity to the task ratio increases with system size.

One of the main conclusions from these experiments is that in order to achieve good speedups for fixed size problems, it is essential that the task ratio be sufficiently large. Similar to the computation to communication ratio being an important consideration for parallel computations, the task ratio is an important factor in non-dedicated distributed computing.

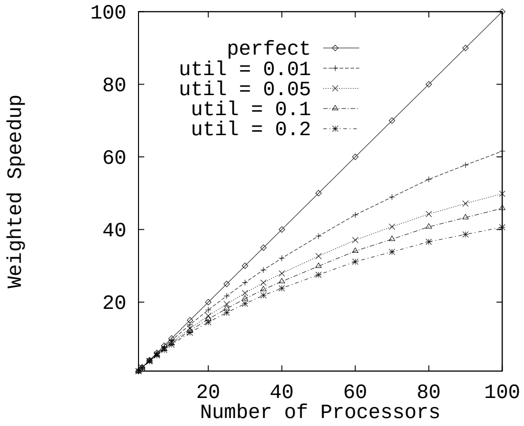

Figure 3: Weighted Speedup

J = 1000 units

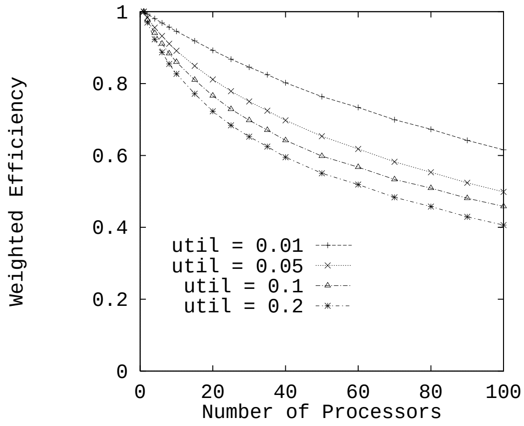

Figure 4: Weighted Efficiency

J = 1000 units

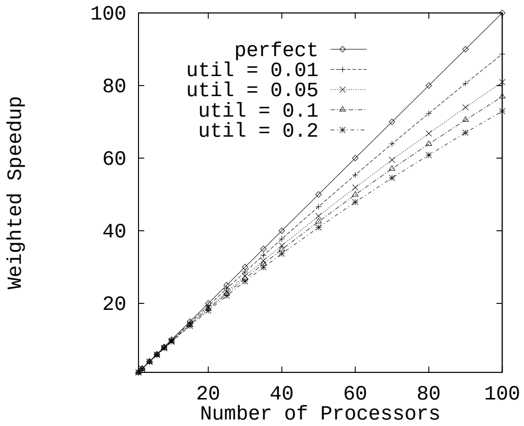

Figure 5: Weighted Speedup

J = 10,000 units

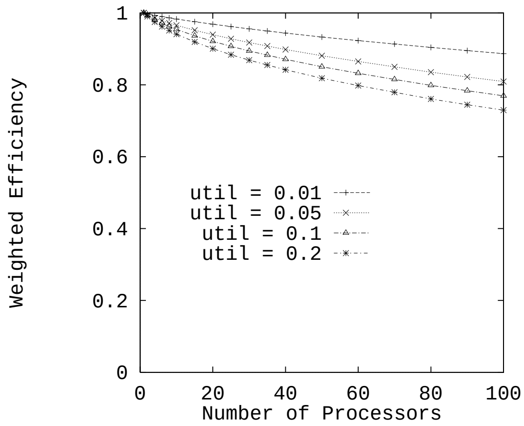

Figure 6: Weighted Efficiency

J = 10,000 units

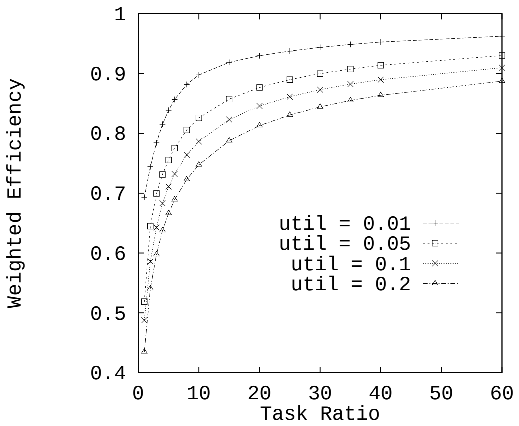

Figure 7: Effect of Task Ratio
60 Workstations

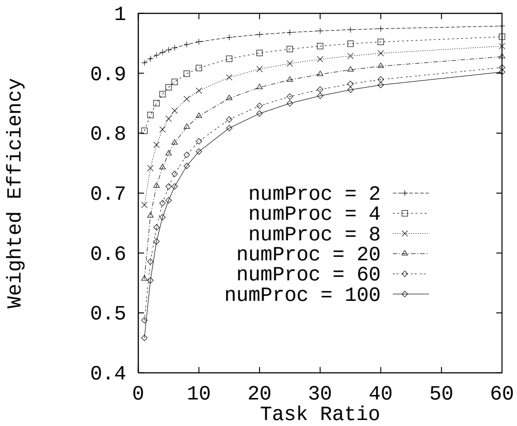

Figure 8: Effect of Task Ratio
Number Workstations Varied,
Owner Utilization = 0.1

3.2 Scaled Problem Size

We now consider the effect of scaling the problem size with the number of nodes. We assume job demand scales linearly with the number of workstations. This type of scaling has been called *memory-bounded* scaleup [14]. With memory-bounded scaleup and perfect parallelism, ideally, we may be able to complete $\mathcal{W}$ times the amount of work in the same time as the original problem on a single workstation by using a system with $\mathcal{W}$ nodes [14]. In Figure 9 we plot job execution time versus the number of workstations assuming job demand is equal to 100 units times the number of workstations. Since the problem size scales, the parallel task demand is a constant 100 units, and hence, the task ratio is fixed at 10. Initially there is a sharp increase in response time as system size increases, but the increase diminishes as system size becomes large. For system utilizations of 1, 5, 10, and 20%, the response time for a problem using 100 workstations increases by 14, 30, 44, and 71% relative to the response time for a problem using one workstation with the same owner utilization. In other words, the distributed computing approach offers the potential to increase the problem size by a factor of 100 and only increase response time by 44% assuming all workstations have a utilization of 10%.

Memory-bounded scaleup exhibits better performance than fixed-size computing since the task ratio is fixed, while the task ratio in fixed-size computing decreases with an increase in the number of workstations. We also considered larger job demands and found the increase in response time to be even less. Hence, we conclude that the distributed computing approach offers significant potential for scaling of problems even if workstation owner processes are granted preemptive priority over parallel tasks.

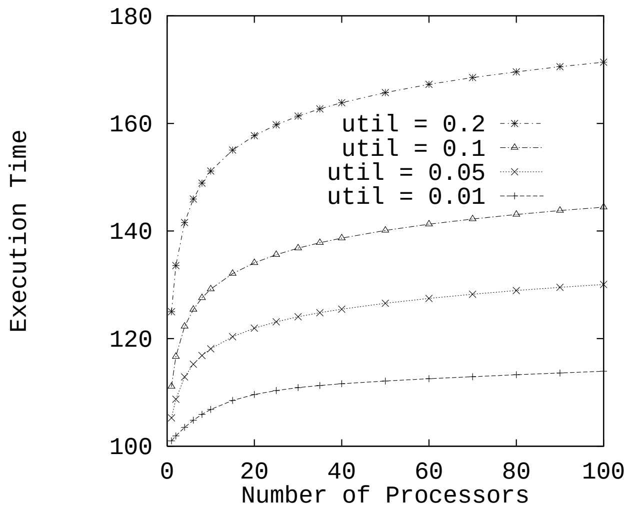

Figure 9: Effect of Scaling Problem

4 Experimental Validation

In this section we present preliminary results from experimental studies to validate the analysis. In these initial studies we focus only on fixed size problems. We have chosen to implement our parallel program using the PVM package [7]. We chose the PVM package based on the package being well known and highly available. We made no attempt to compare the PVM package with any other distributed computation packages.

To isolate the effects of workstation owner interference we assume the parallel program is a local computation problem [13]. That is, the problem has perfect parallelism and no interprocess communication. The parallel program forks W parallel tasks, one for each workstation in the system, and each task executes independently. Each parallel task is "niced" (runs at low priority) granting workstation owner processes preemptive priority over the parallel tasks.

Our primary metrics are maximum task execution time and speedup. The most common metric for a study such as this is job response time, i. e. the time from the parallel job is started until it completes. This metric is influenced by the overhead of the parallel computing package for initiating the processes and collecting the results. We want to focus only on the interference of workstation owner processes and thus rejected defining response time in this standard way. Instead, we focus on the maximum task execution time. This time was obtained by having each task record the system time when it started computation and noting the system time immediately when completing computation. Each of the parallel tasks then return their task execution time to the master process which selects and reports the maximum. By considering the maximum task execution time we isolate the impact of workstation owner process interference.

We report the results from one experiment. The system studied is composed of at most 12 Sun ELC Sparcstations. We varied the number of workstations from 1 to 12, first ensuring that none of the workstations are executing long running jobs. In general the only interference is from more trivial usage such as editing files, reading mail, news, etc. For each number of workstations considered we ran the parallel program 10 times for each parameter value and calculated the mean of these 10 runs as our metric. Given the number of workstations, the input parameter to our parallel program is the problem size. We consider five different problem sizes; 1,2,4,8, and 16 minutes are the service demands of these problems on a single dedicated machine. No attempt has yet been made to provide confidence intervals or more detailed statistical analysis.

In Figure 10 we plot the maximum task execution time versus the number of workstations for the five different job demands assuming a fixed problem size. The solid lines are the measured values from our experiment. The dashed lines are predictions from our analytical model where the input parameter for workstation owner utilization is set to 3%. We obtained the 3% value by computing the mean of the machine utilizations (by using the unix uptime command) over two working days when no PVM programs were executing. The models qualitative and quantitative predictions are in close agreement with the measured results.

In Figure 11 we plot the speedup versus the number of workstations. The values plotted were obtained from measurement of the system. In this case we define speedup as the ratio of the maximum task execution time using one workstation over the maximum task execution time using $\mathcal{W}$ workstations. The utilization of the machines is very low and thus there is not significant degradation of parallel program performance. In a more heavily loaded system we would expect much more degradation. Focusing on the 8 and 12 workstation cases we see that the speedup decreases as the job demand decreases, i.e. the speedup for a job demand of 1 is lower than the speedup for a job demand of 16. This is because the task ratio is smaller for a job demand of 1 than it is for a job demand of 16. This experiment thus qualitatively validates the analysis. Note that the analysis shows a more significant drop in speedup as system size increases. Unfortunately we only have 12 homogeneous workstations with which to validate our results and hence can not experimentally validate this result.

5 Sensitivity to Workstation Owner Process Variance

As noted before, most workloads running on computers have a significant amount of variation in processor demands. For many system the coeffecient of variation has been found to be around 10 [11]. In this section we study the effect of workstation owner process variance on parallel job performance.

We first compare weighted speedup predictions of our deterministic analysis with those obtained

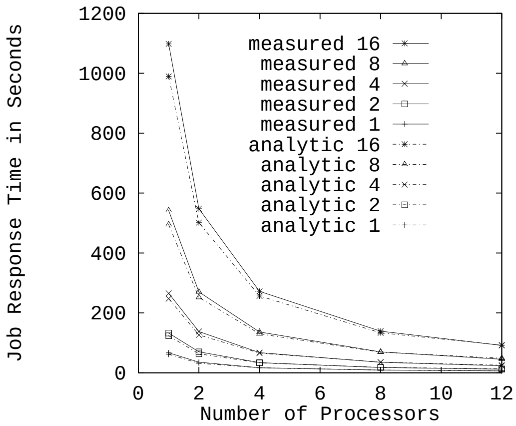

Figure 10: Experimental Validation
Response Time

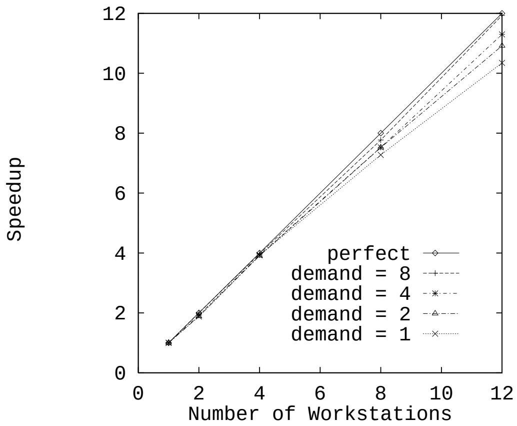

Figure 11: Experimental Validation
Speedups

from our simulation assuming a coefficient of variation (abbreviated CX in the figures) of 5 for workstation owner process demands. In figure 12 we plot the weighted speedup versus the number of processors for utilizations of 1%, 5%, 10%, and 20% assuming a parallel job demand ($\mathcal{J}$) equal to 1000 units, and an owner processed demand ($\mathcal{O}$) equal to 10 units. The dashed curves are from the deterministic analysis, CX = 0, and the solid curves are from the simulation with CX = 5. The difference is quite substantial. For utilizations of 5% and greater the weighed speedup actually decreases beyond a certain number of processors. This is not due to communication, as found in many speedup curves of real systems, since communication costs are assumed to be zero. Instead, this degradation is due to an increased probability of having a workstation owner processes have a large demand thus slowing down the parallel job. In the deterministic model the probability of interferences also increases with the number of workstations, but the penalty is not as high since the amount of the workstation owner will use the workstation is bounded. In figure 13 we plot results from the same experiment when parallel job demand ($\mathcal{J}$) is equal to 10,000 units. We see once again that the increased variance results in substantially less speedup potential than when workstation process demands are deterministic.

To determine the sensitivity of the weighted speedup to the variance of workstation owner processes we hold all parameters constant and vary the coefficient of variation. In figures 14 and 15 we plot the weighted speedup versus the coefficient of variation assuming parallel job demands of 1000 and 10,000 units respectively. An increase in the variation of owner process demand quickly degrades weighted speedup.

The proceeding experiment indicates that the task ratio most likely needs to be increased to achieve good speedups in a distributed system with high workstation owner process variance. In

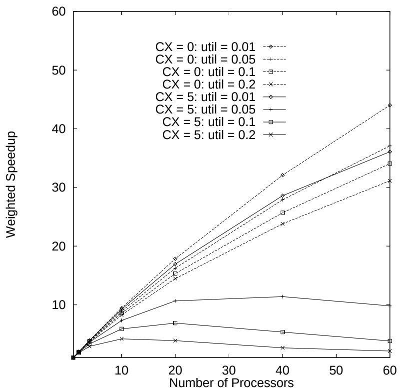

Figure 12: Weighted Speedup
J = 1000 units

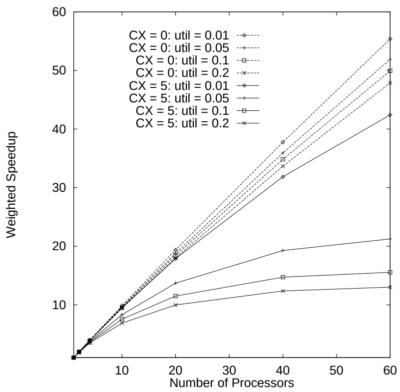

Figure 13: Weighted Speedup
J = 10,000 units

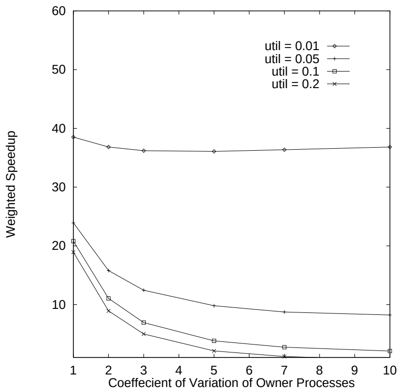

Figure 14: Sensitivity to Variance
J = 1000 units

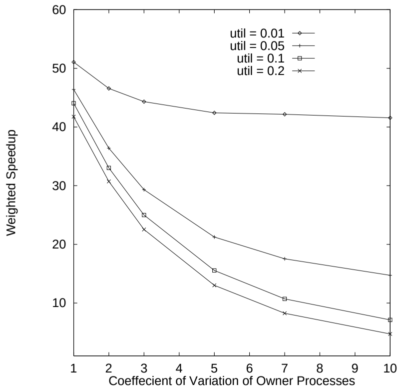

Figure 15: Sensitivity to Variance
J = 10,000 units

figure 16 we plot the weighted efficiency versus the task ratio for different variances. The number of processors is set to 60, and the utilization is set to 10%. The curves from top to bottom are for a coefficient of variation of 0, 1, 3, 5, and 10. Note, the weighted speedups can be obtained by multiplying by 60. As the coefficient of variation increases, the task ratio needed to achieve good efficiency increases dramatically. For example, to achieve a weighted efficiency of 80%, interpolation of our graph shows that the task ratio must be greater than (13,30,200,500) for coefficient of variations of (0,1,3,5). For a coefficient of variation of 10 the weighted efficiency is only 71% even at a task ration of 1000. Thus, it is crucial to know workstation process variance in order to determine the minimum size of a job necessary to achieve acceptable performance.

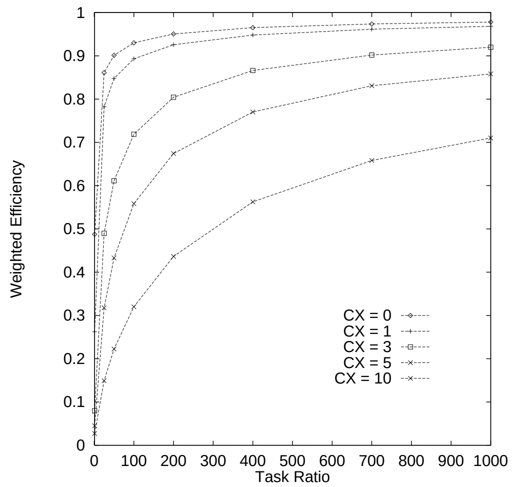

Figure 16: Effect of Task Ratio With Variance, 60 Workstations

6 Conclusions and Discussion

In this paper we have developed an abstract model and simulation of a network computing system to determine the feasibility of using distributed computing in a non-dedicated network system assuming workstation owner processes have preemptive priority over parallel tasks. The model is an abstraction of a parallel program ignoring communication and synchronization overheads. We assume the targeted parallel programs execute efficiently on a dedicated distributed system, hence we can ignore these overheads and focus on the impact of a non-dedicated environment. The purpose of considering a non-dedicated system is to determine if idle (wasted cycles) workstations on network can be utilized to reduce execution time and to solve large problems. We limit our study to systems where workstation owner processes have preemptive priority over tasks belonging to parallel jobs.

For fixed-size problems we have found that good speedups can be achieved, but only if the amount of work allocated to each machine is sufficiently large compared to the mean service demand of workstation processes and the variance of workstation owner processes is low. Hence, for non-dedicated systems where the workstation owner processes have preemptive priority over parallel tasks, the parallel task demand to owner task demand ratio (*task ratio*) is a determining factor in performance of the parallel program. In addition, the task ratio needed to achieve good performance of the parallel program increases with system utilization. Further more, high variance in workstation owner process demands require the task ratio to be significantly greater to achieve reasonable speedups. A system with high workstation owner process variance will require either processes migration or extremely large parallel jobs to absorb the interferences from the workstation owners. A software package, namely MpPVM, has been under developing to support process migration in a non-dedicated network environment [4]. Notice that the task ratio is a function of number of workstations used in the network computing. By chosing an appropriate number of workstations for a given application, the analytical model proposed in this study has established a foundation for efficient task partioning and scheduling in network computing environment.

For scaled problems under a non-dedicated environment, we have found that distributed computing offers significant potential for the efficient execution of scaled problems. In particular, assuming each workstation in the system has a utilization of 5 percent (20 percent) and that workstation owner process demands are deterministic, mean job response time is only increased by 30 percent (71 percent) when comparing the response time of a scaled problem using 100 workstations relative to that of problem using one workstation with a 5 percent (20 percent) utilization. The performance difference between fixed-size and scaled problems is due to the fact that the task ratio of scaled problems is fixed, while the task ratio of fixed-size problems decreases as the number of workstation increases. Note that the results are based on our idealized assumption and hence are optimistic. The actual response time of these problems would be dependent on communication bandwidth requirements which are ignored in our model.

Our models provide insight into the general problem of network computing in a non-dedicated system, actual performance prediction would require more detailed workload characterization of both workstation owner use, and parallel job requirements.

References

[1] T. Anderson, D. Culler, D. Patterson, "A Case for Networks of Workstations: NOW," IEEE Micro, Feb, 1995.

[2] R.H. Arpaci, A.C. Dusseau, A.M. Vahdat, L.T. Liu, T.E. Adderson, D.A. Patterson, "The Interaction of Parallel and Sequential Workloads on a Network of Workstations," Proc. 1995 ACM SIGMETRICS, May 1995.

[3] A. Beguelin and J.J. Dongarra and G.A. Geist and R. Mancheck and K. Moore and V.S. Sunderam, "Tools for Heterogeneous Network Computing," Proc. 6th SIAM conf. on Parallel Processing For Scientific Computing, 1993.

[4] K. Chanchio, X.-H. Sun, "MpPVM: The Migration-point based PVM, (Part I High Level Design)," Technical Report, Department of Computer Science, Louisiana State University, 1995.

[5] F. Douglis, J. Ousterhout, "Transparent Process Migration: Design Alternative and the Sprite Implementation," Sotware Practice and Experience, vol. 21, no. 8, pp. 757-785, August 1991.

[6] G. Fox et. al., **Solving Problems on Concurrent Processors**, Prentice-Hall Inc., 1988.

[7] G.A. Geist and et. al., **PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing**, The MIT Press, 1994.

[8] W. Gropp and E. Lusk and A. Skjellum, **Using MPI: Portable Parallel Programming with the Message-Passing Interface**, The MIT Press, 1994.

[9] Kobayashi, **Modeling and Analysis**, Addison-Wesley, 1978.

[10] M.M. Mutka, M. Livny, "The Available Capaciy of a Privately Owned Worstation Environment," Performance Evaluation, vol. 12, no. 4, pp. 269-284, July 1991.

[11] C.H. Sauer and K.M. Chandy, **Computer System Performance Modeling**, Prentice-Hall, 1981.

[12] H.D. Schwetman, "CSIM: A C-Based Process-Oriented Simulation Language," Proc. of 1986 Winter Simulation Conference, Dec., 1986.

[13] Xian-He Sun and L. Ni, "A Structured Representation for Parallel Algorithm Design on Multicomputers," Proc. of the Sixth Conf. on Distributed Memory Computing, April 1991, pp. 596-599.

[14] Xian-He Sun and L. Ni, "Scalable Problems and Memory-Bounded Speedup," J. of Parallel and Distributed Computing, Sept. 1993.

[15] V.S. Sunderam, "PVM: A Framework for Parallel Distributed Computing," Concurrency: Practice and Experience, vol 2, no 4, Dec 1990.

[16] M. Theimer, K. Landtz, D. Cheriton, "Preemptable Remote Execution Facilitis for the V System," Proc 10th ACM Syposium on Operating System Priciples, Dec 1985.