IN-61
191149
25 p

NASA Contractor Report 191532

ICASE Report No. 93-65

ICASE

20 Years of Excellence

DISTRIBUTED COMPUTING FEASIBILITY IN A NON-DEDICATED HOMOGENEOUS DISTRIBUTED SYSTEM

Scott T. Leutenegger
Xian-He Sun

NASA Contract No. NAS1-19480
September 1993

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23681-0001

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

Distributed Computing Feasibility in a Non-Dedicated Homogeneous Distributed System

Scott T. Leutenegger Xian-He Sun

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001

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. 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 large the demand of a parallel applications must be in order to make efficient use of a non-dedicated distributed system.

*This research was supported 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. We will refer to this environment as *distributed 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 distributed processing approach. Several commercial and noncommercial tools have been developed to support distributed computing. One widely used tool is the *Parallel Virtual Machine* (PVM) project [9, 5, 1, 2]. According to the authors, PVM is now being used at more than 100 sites. 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. 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 [3]. 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.

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 computing in a non-dedicated environment where workstation owner processes have preemptive priority over parallel tasks a viable approach?"

An analytical model is developed to predict the performance under the non-dedicated assumption. 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.

In addition to our analysis, a hypothetical local computation [11] 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, and our conclusions are in Section 5.

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.

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.

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.

2.1 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:

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

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 [7]. 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 [8]. The purpose of the simulation is solely to validate the coding of our analysis. We intend to use our simulation in future work to explore other service demand distributions.

All results have confidence intervals of 1 percent or less at a 90 percent confidence level. Confidence intervals are calculated using batch means [4] with 20 batches per simulation run and a batch size of 1000 samples. 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 did not plot the results since they are indistinguishable from the analysis.

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. 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

We first address the benefit of the distributed computing approach for a fixed-size job. In this case, 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 workstations 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 the 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 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 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.

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 [10]. With memory-bounded scaleup and perfect parallelism, ideally, we may be able

to complete W times the amount of work in the same time as the original problem on a single workstation by using a system with W nodes [12]. 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.

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. 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 [11]. 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. Further experiments are currently being conducted. 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.

If 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 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 Conclusions and Discussion

In this paper we have developed an abstract model of a distributed computing system to determine the feasibility of using distributed computing in a non-dedicated 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 can be utilized to reduce execution time and to solve large problems.

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. 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 particular, we find that the task ratio should be at least 8 for a parallel job to achieve 80 percent of the possible speedup, even adjusting for system utilization, for a system in which each homogeneous workstation has a utilization of 5 percent. In addition, the task ratio needed to achieve 80 percent of the possible speedup increases with system utilization. At a utilization of 10 percent the task ratio must be 13 or higher, and at a utilization of 20 percent the task ratio must be 20 or greater.

The model proposed in this paper assumes local workstation processes have deterministic service requirements. This assumption implies that results presented in this paper is conservative. Hence, even larger task ratios are likely to be necessary to achieve good performance. Thus, based on our study, distributed computing in a non-dedicated environment where workstation owner processes have preemptive priority over parallel tasks is a viable approach only if the task ratio is sufficiently large. The exact size of the ratio needed is both application and environment dependent.

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), 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.

We assume the workload of the non-dedicated environment is light and the effect of long running workstation owner jobs is not considered. How to provide reasonable execution times for parallel jobs in a non-dedicated system with long running workstation owner jobs must be solved if distributed computing is to be feasible in a non-dedicated environment. Currently our model only provides some initial insights into the general problem of distributed computing in a non-dedicated system. In the future we intend to extend the model to handle more complex workloads. In addition, we are currently pursuing further experimental validation of our model.

References

[1] Beguelin, A., Dongarra, J.J., Geist, G.A., Mancheck, R., and Sunderam, V.S., "A Users' Guide to PVM Parallel Virtual Machine," Technical Report ORNL/TM-11826, Oak Ridge National Laboratory, July 1991.

[2] Beguelin, A., Dongarra, J.J., Geist, G.A., Mancheck, R., Moore, K., and Sunderam, V.S., "Tools for Heterogeneous Network Computing", Proc. 6th SIAM conf. on Parallel Processing For Scientific Computing, Vol 2, March 1993.

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

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

[5] Geist, G.A., and Sunderam, V.S., "Experiences With Network Based Concurrent Computing on the PVM System", Technical Report ORNL/TM-11760, Oak Ridge National Laboratory, January 1991.

[6] J.L. Gustafson and G.R. Montry and R.E. Benner, "Development of Parallel Methods for a 1024-processor Hypercube", SIAM J. on SSTC, Vol. 9, No. 4, 1988.

[7] Sauer, C.H., Chandy, K.M., *Computer System Performance Modeling*, Prentice-Hall, 1981, page 16.

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

[9] Sunderam, V.S., "PVM: A Framework for Parallel Distributed Computing", Concurrency: Practice and Experience, Vol. 2, No. 4, December 1990.

[10] Xian-He Sun and L. Ni, "Another View on Parallel Speedup", Proc. of Supercomputing '90, Nov. 1990.

[11] 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.

[12] Xian-He Sun and L. Ni, "Scalable Problems and Memory-Bounded Speedup", J. of Parallel and Distributed Computing, Vol. 19, Sept. 1993.

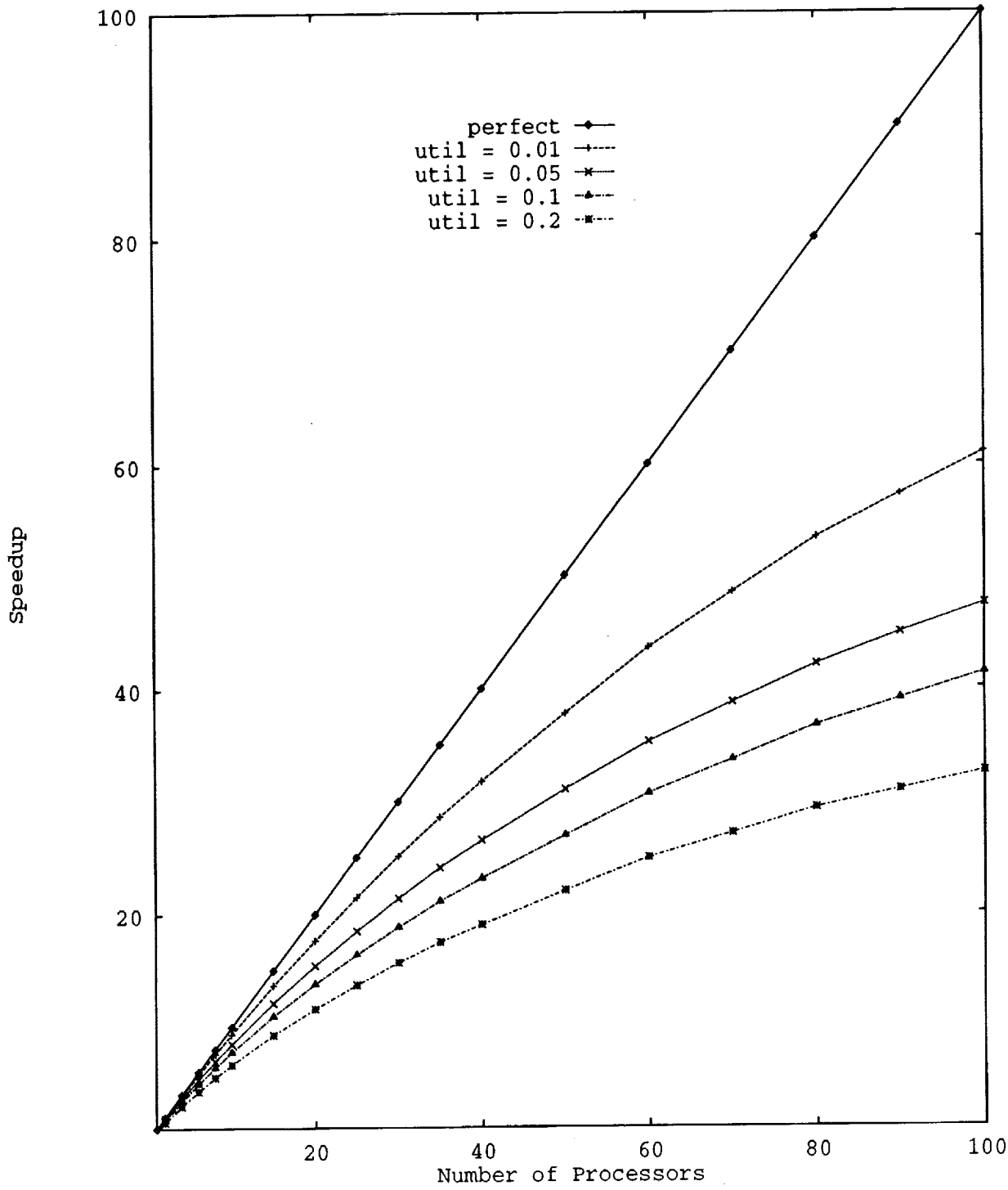

Figure 1: Speedup, J = 1000 units

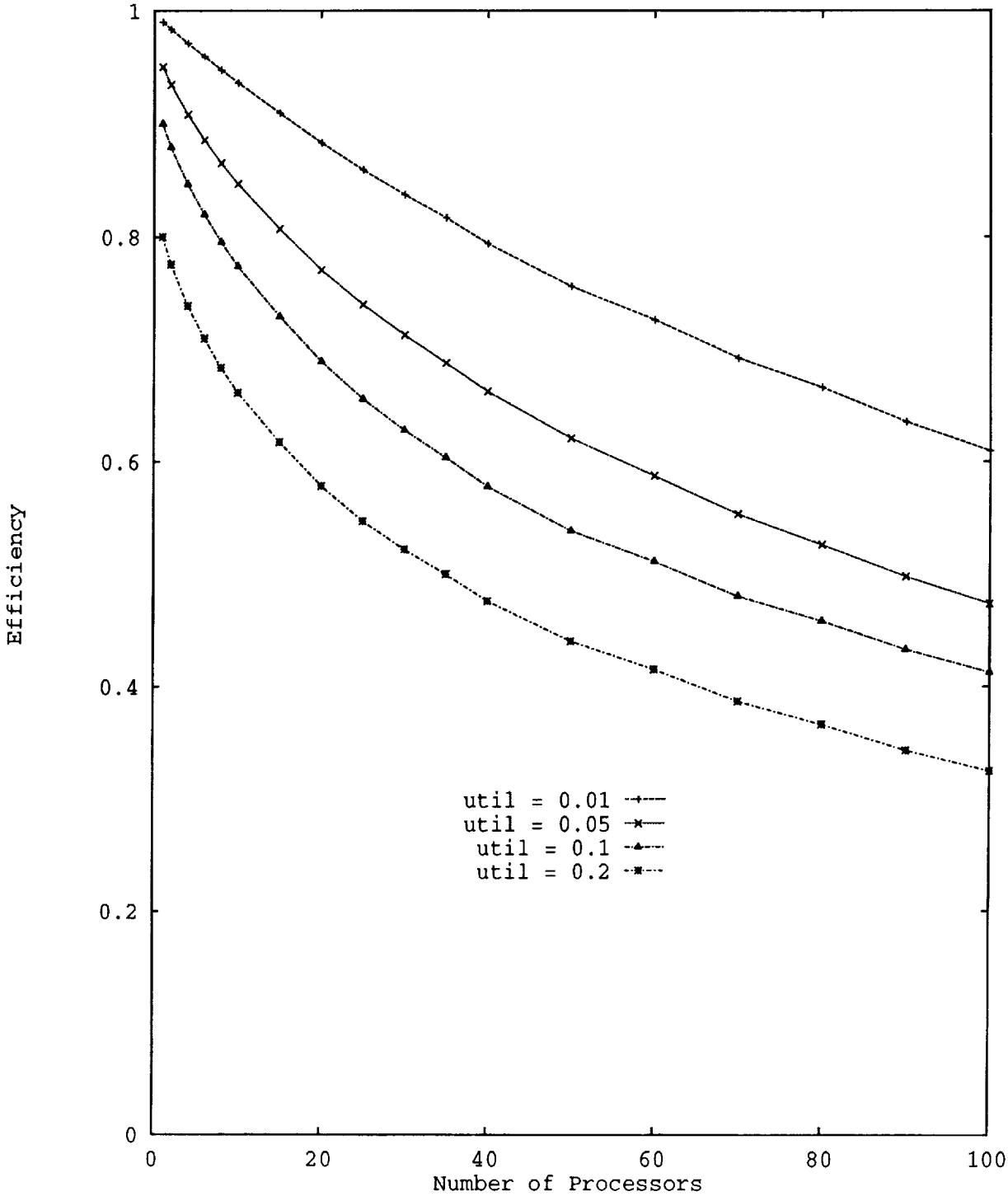

Figure 2: Efficiency, J = 1000 units

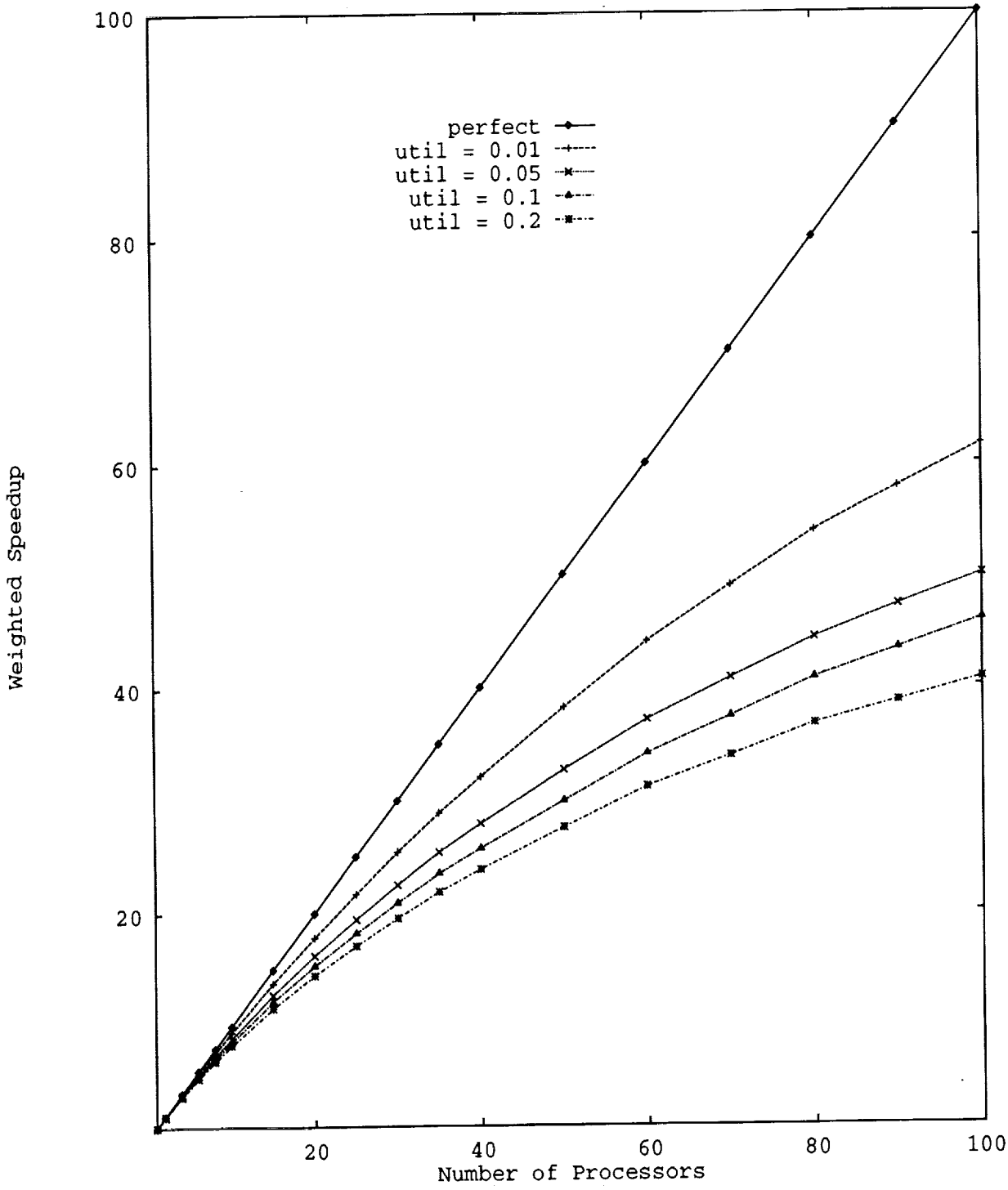

Figure 3: Weighted Speedup, J = 1000 units

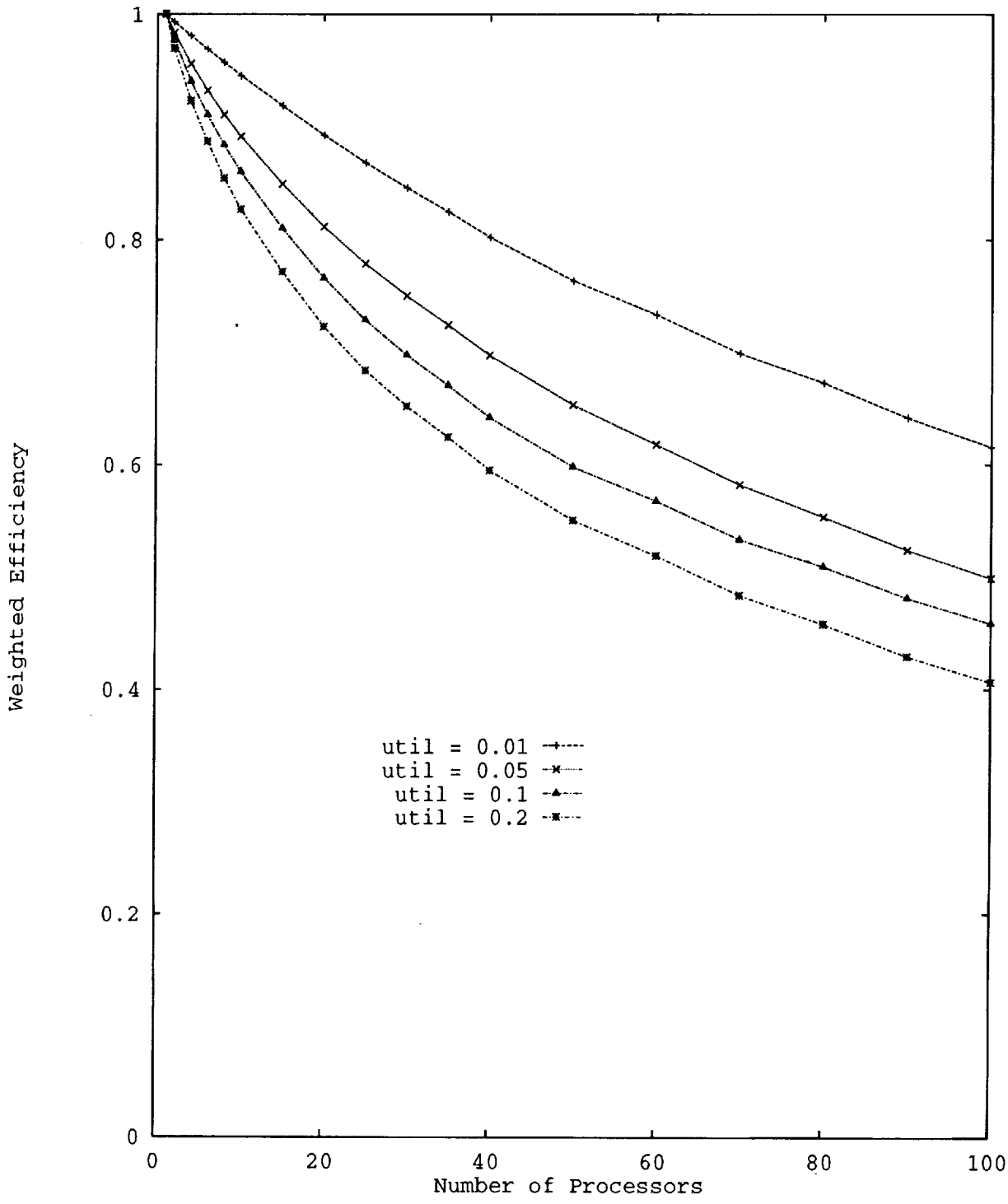

Figure 4: Weighted Efficiency, J = 1000 units

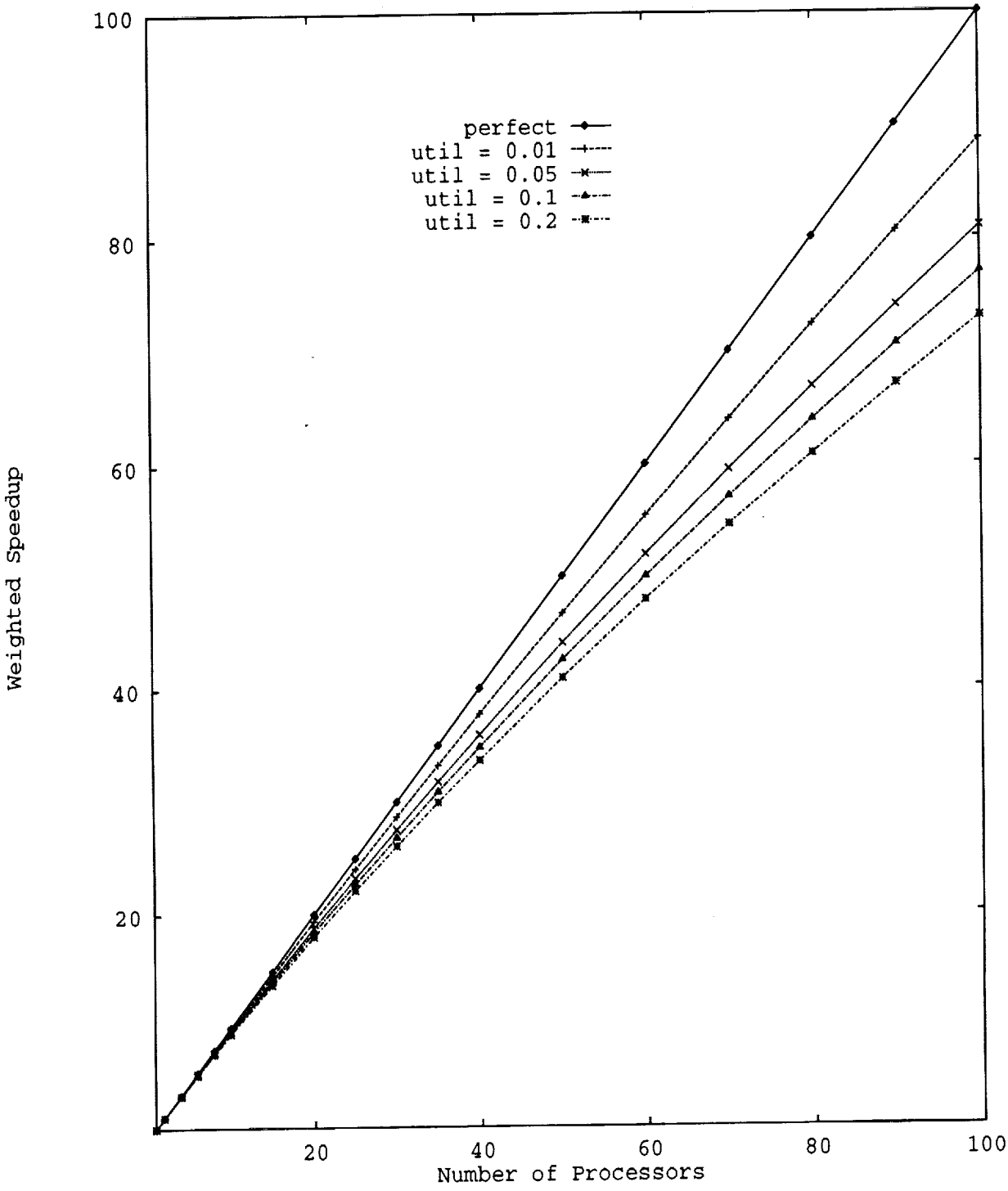

Figure 5: Weighted Speedup, J = 10,000 units

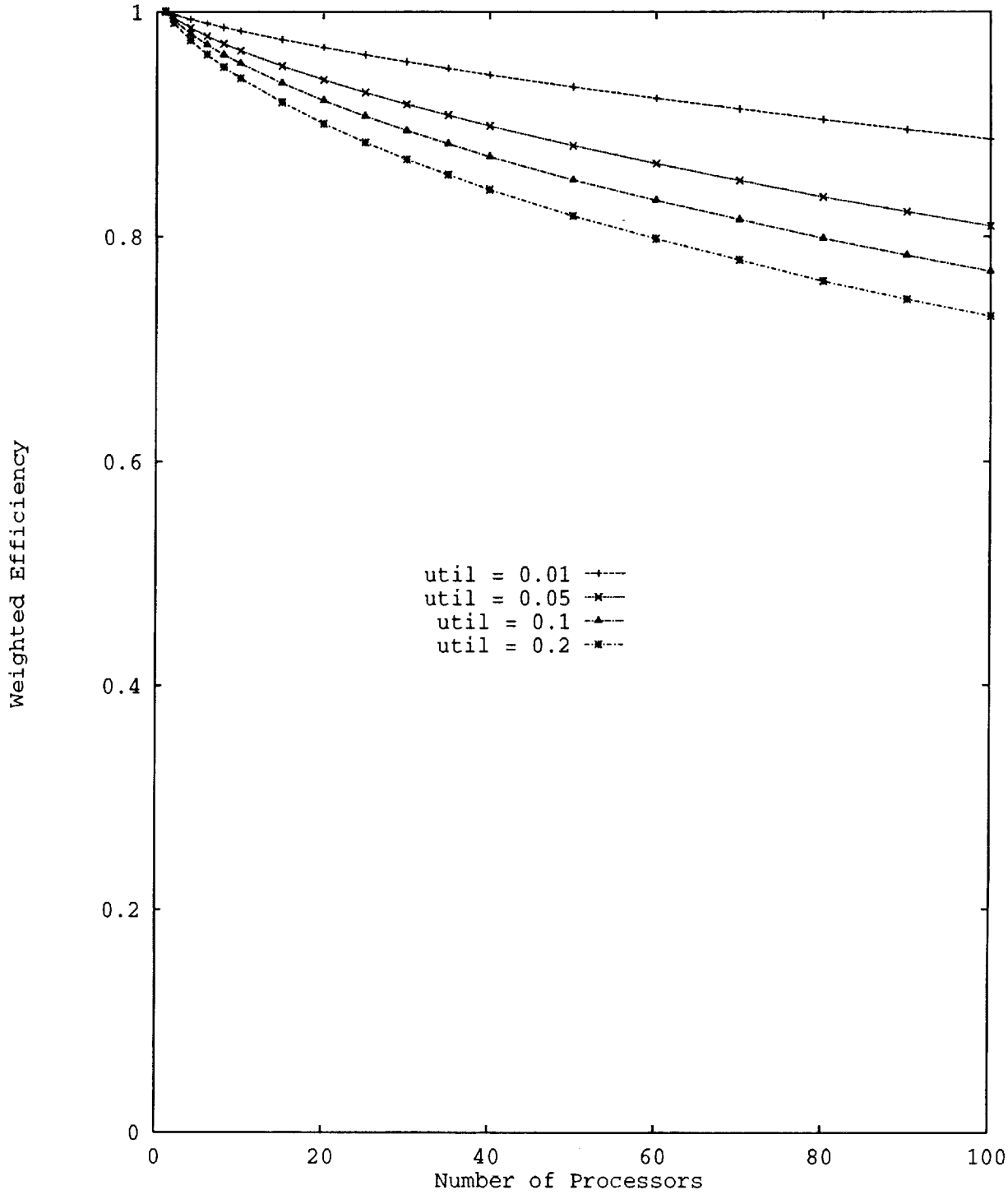

Figure 6: Weighted Efficiency, J = 10,000

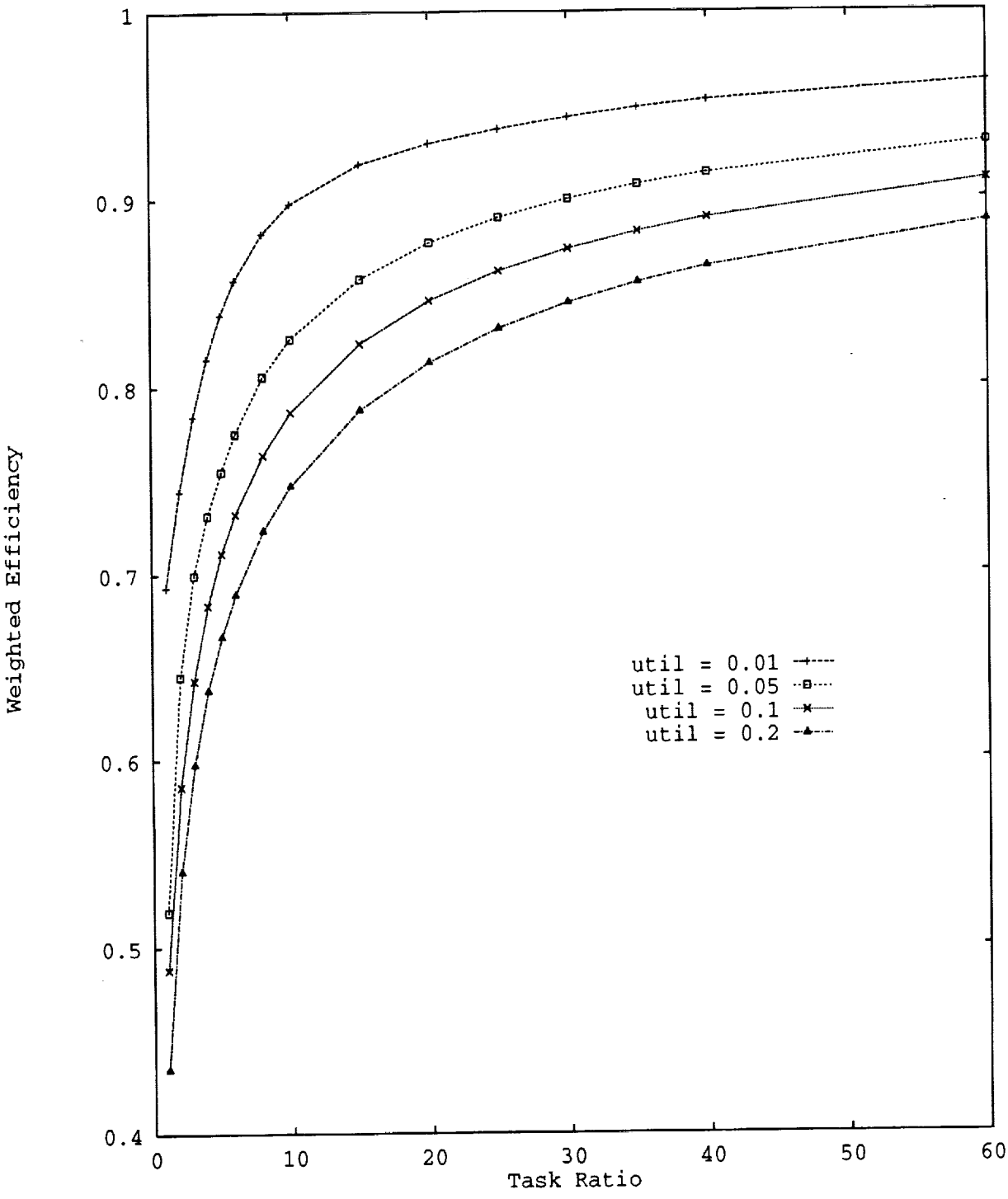

Figure 7: Effect of Task Ratio, 60 Workstations

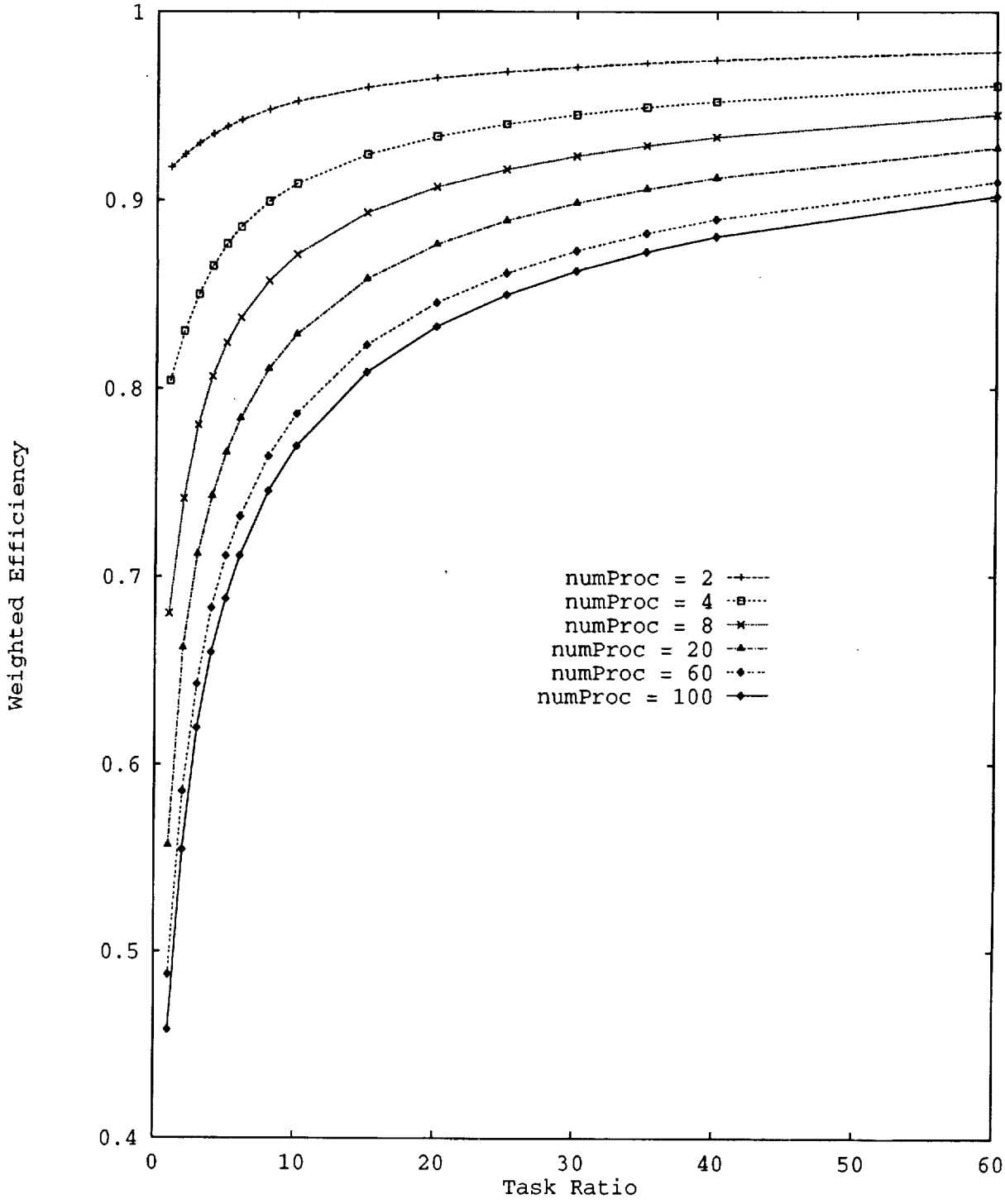

Figure 8: Effect of Task Ratio, Number Workstations Varied, Owner Utilization = 0.1

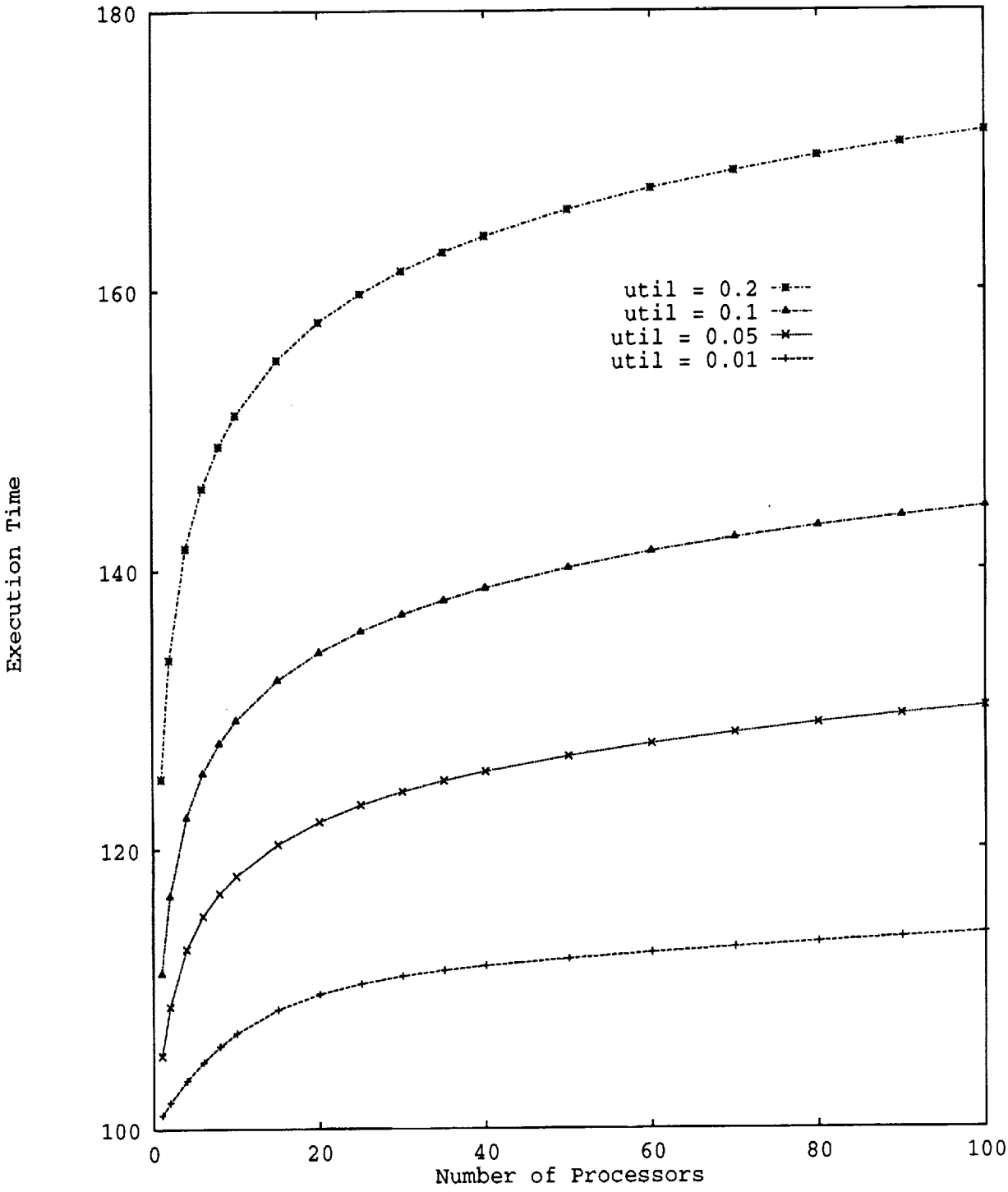

Figure 9: Effect of Scaling Problem

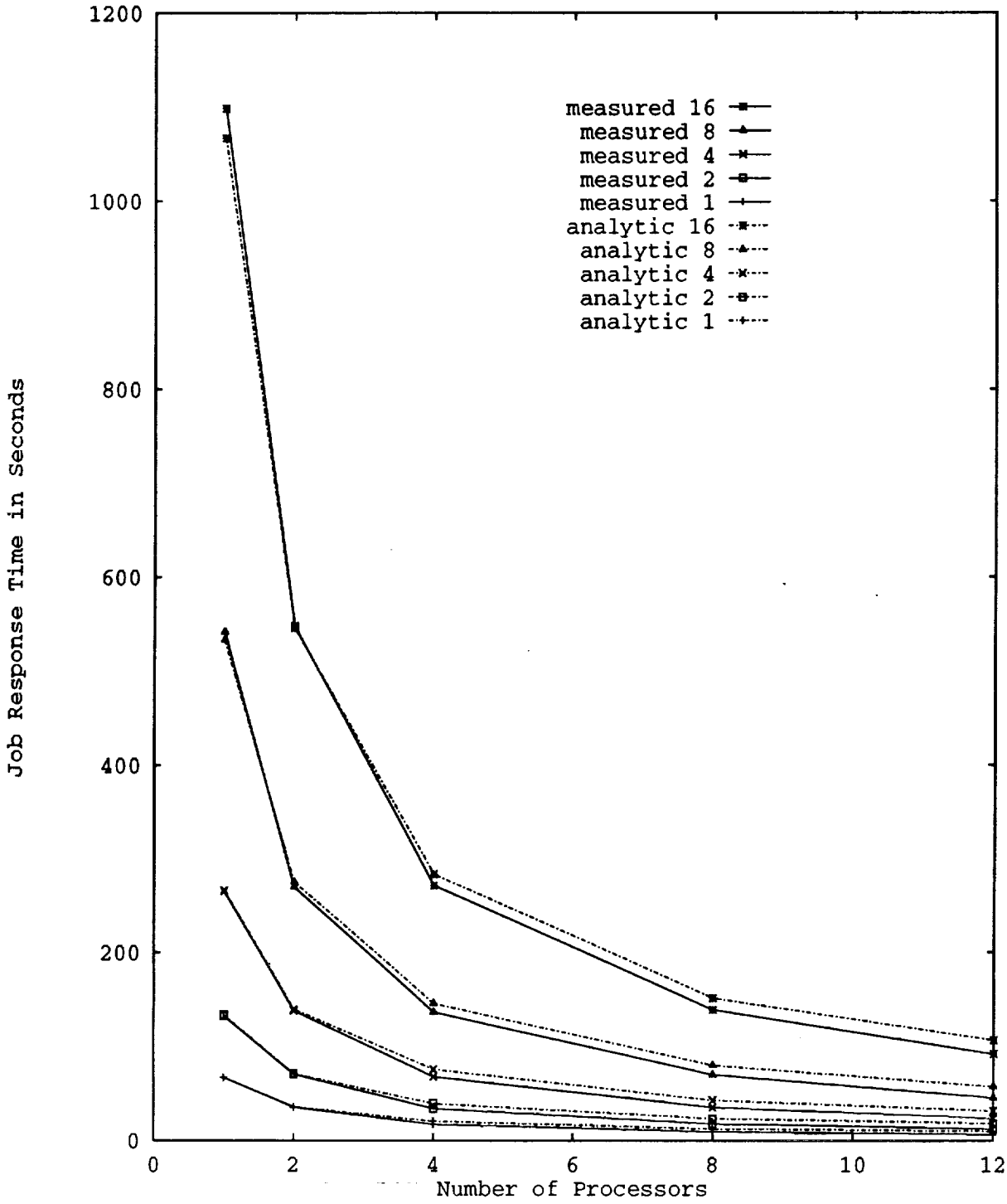

Figure 10: Experimental Validation: Response Time

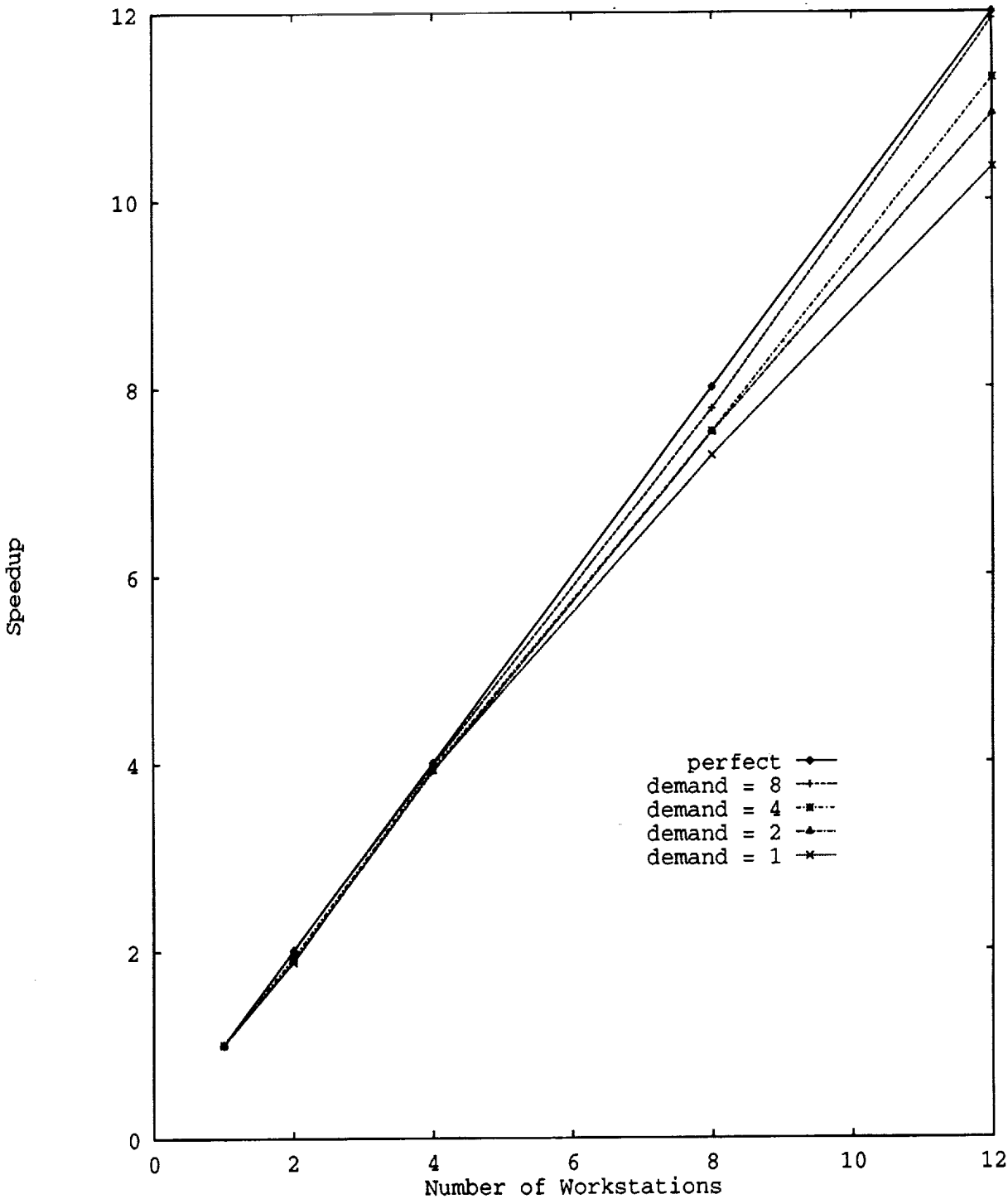

Figure 11: Experimental Validation: Speedups

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1993	Contractor Report

4. TITLE AND SUBTITLE
DISTRIBUTED COMPUTING FEASIBILITY IN A NON-DEDICATED HOMOGENEOUS DISTRIBUTED SYSTEM

5. FUNDING NUMBERS
C NAS1-19480

WU 505-90-52-01

6. AUTHOR(S)
Scott T. Leutenegger
Xian-He Sun

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 93-65

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-191532
ICASE Report No. 93-65

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Michael F. Card
Final Report

To appear in Supercomputing '93

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
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. 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 large the demand of a parallel applications must be in order to make efficient use of a non-dedicated distributed system.

14. SUBJECT TERMS	15. NUMBER OF PAGES
distributed systems, parallel computing	24
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std Z39-18
298-102

☆U.S. GOVERNMENT PRINTING OFFICE: 1993 - 728-064/86066

National Aeronautics and
Space Administration
Code JTT
Washington, D.C.
20546-0001

Official Business

Penalty for Private Use, $300

BULK RATE
POSTAGE & FEES PAID
NASA
Permit No. G-27

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return