Load Balancing for Minimizing Execution Time of a Target Job on a Network of Heterogeneous Workstations

S.-Y. Lee and C.-H. Cho
Department of Electrical and Computer Engineering
Auburn University
Auburn, AL 36849.
sylee@eng.auburn.edu

Abstract

A network of workstations (NOWs) may be employed for high performance computing where execution time of a target job is to be minimized. Job arrival rate and size are "random" on a NOWs. In such an environment, partitioning (load balancing) a target job based on only the first order moments (means) of system parameters is not optimal. In this paper, it is proposed to consider the second order moments (standard deviations) also in load balancing in order to minimize execution time of a target job on a set of workstations where the round-robin job scheduling policy is adopted. It has been verified through computer simulation that the proposed static and dynamic load balancing schemes can significantly reduce execution time of a target job in a NOWs environment, compared to cases where only the means of the parameters are used.

Key Words: Dynamic load balancing, Execution time, Network of workstation, Round-robin job scheduling, Standard deviation, Static load balancing, Stochastic model

1 Introduction

A network of workstations (NOWs), or computers, is being employed as a high performance distributed computing tool in an increasing number of cases [1] [2]. Accordingly, using a NOWs efficiently for speeding up various applications has become an important issue. In particular, load balancing has a significant effect on performance one can achieve on a NOWs environment. The issue of load balancing in general is not new. Many researchers have investigated various aspects of load balancing for quite long a time [3][4][5][6][7][8][9][10].

There were many parameters and characteristics considered in load balancing. They include processor speed, job arrival rate and size, communication among jobs (or subtasks), homogeneity (or heterogeneity) of system, load balancing overhead, specific characteristics of a job, etc. In most of the previous work [11][12], only the *means* of such parameters were used in load balancing. However, a parameter may have the same mean for all workstations, but quite different a variance on a different workstation in a heterogeneous environment. Also, in many cases [13][14][15], the emphasis was on balancing job distribution rather than minimizing execution time of a *target* job.

There is a feature of NOWs, which distinguishes it from other high performance computing platforms, especially dedicated tightly-coupled multiprocessor systems, i.e., *randomness of job arrival* on an individual workstation. This is mainly due to the fact that each workstation is usually shared by multiple independent users who submit their jobs at any time. Also, the size of a job is random. This randomness in job arrival and size makes the number of jobs sharing (the processor on) a workstation time-dependent. That is, the number of jobs on a workstation is to be modelled as a random variable. When the processor is shared among jobs in a round-robin fashion, the amount of work completed in each job during a time interval depends on (e.g., inversely proportional to) the number of jobs (sharing the processor) in that interval. On a network of such workstations, distributing (load balancing) a target job considering only the mean of the number of jobs on each workstation does not achieve the minimum possible execution time. As will be shown later, not only the mean but also the standard deviation of the number of jobs affects execution time of a job on a workstation. Therefore, in order to minimize execution time of a target job, it is necessary to take into account the standard deviation of the number of jobs on each workstation in addition to its mean in load balancing.

In this paper, as a first step toward developing an efficient load balancing scheme, it is shown analytically and demonstrated via simulation that the second order moments as well as the first order moments of parameters are to be used to minimize execution time of a target job on a network of heterogeneous workstations. Workstations are considered to be heterogeneous when the mean and standard deviation of the number of jobs vary with workstation. Each workstation is time-shared by multiple jobs. In this early study, it is assumed that a target job can be arbitrarily partitioned for load balancing and communication is not required among subtasks.

The main contributions of this work are (i) derivation of analytic formulas of performance

measures used in load balancing, (*ii*) design of static and dynamic load balancing schemes for minimizing execution time of a target job, and (*iii*) showing that a load balancing scheme utilizing the standard deviations as well as the means of parameters can outperform those considering the means only.

In Section 2, a stochastic model of workstations is described. In Section 3, a set of measures is derived analytically on a single workstation, which are to be used for load balancing on multiple workstations. In Section 4, the proposed static and dynamic load balancing strategies are described. In Section 5, results from extensive computer simulation are discussed to validate the proposed strategies. In Section 6, a conclusion is provided with remarks on the future directions.

2 A Stochastic Model of Workstations

Glossary

The following notations are adopted in this paper.

W	the number of workstations
W_i	workstation i
X	the size of a target job to be distributed
X_i	the portion of X assigned to W_i
a_i	the number of jobs (random variable) arrived in an interval on W_i
A_i	the mean of a_i
σ_{a_i}	the standard deviation of a_i
n_i	the number of jobs (random variable) in an interval on W_i, excluding those arriving in the current interval
N_i	the mean of n_i
σ_{n_i}	the standard deviation of n_i
s_i	the size of job (random variable) arriving at W_i
S_i	the mean of s_i
σ_{s_i}	the standard deviation of s_i
μ_i	the service rate (computing power) of W_i
t_i	execution time (random variable) measured in intervals on W_i
T_i	the mean of t_i
σ_{t_i}	the standard deviation of t_i
O_c	overhead involved in checking load distribution
O_r	overhead involved in redistributing load
$E[Z]$	expectation of Z

When a variable is to be distinguished for each time interval, a superscript with parentheses will be used, e.g., $a_i^{(j)}$ denotes a_i for the jth interval. The subscript of a random variable, which

is used to distinguish workstations, is omitted when there is no need for distinction, e.g., a single workstation or when it does not vary with workstation.

Service Policy

A time *interval* is a unit of time for scheduling jobs (sharing the processor) on a workstation. All time measures are expressed in intervals. It is assumed that each workstation adopts a round-robin scheduling policy. A workstation (processor) spends, on each job in an interval, an amount of time which is inversely proportional to the number (n_i) of jobs in that interval. That is, $\frac{\mu_i}{n_i}$ is allocated for each job in an interval on workstation i (denoted by W_i) where μ_i is *the service rate* of W_i. Those jobs arrived in an interval start to be serviced (processed) in the following interval without any distinction depending on their arrival times (as long as they arrive in the same interval). It is to be noted that n_i includes all jobs arrived but not completed by I_{i-1}.

Job Arrival

Jobs are submitted at random time instances and, therefore, the number of jobs arriving in an interval may be modelled by a random variable denoted by a_i. *The mean and standard deviation* of a_i are denoted by A_i and σ_{a_i}, respectively.

Job Size

The size of a job varies with job and may have a certain distribution with a mean (S_i) and a standard deviation (σ_{s_i}). It is assumed that the job size is independent of the job arrival rate.

Overheads

Loads on workstations are checked to determine if load balancing is to be done. It is assumed that, given a number of workstations, there is a fixed amount of overhead, O_c, for checking information such as the remaining portion of X_i on W_i, and also a constant overhead, O_r, for redistributing the remaining X over workstations when it is decided to perform load balancing.

The job arrival rate and job size will be referred to as *system parameters*. The distributions of the system parameters may be known in some cases. Or their means and standard deviations can be estimated from a given network of workstations. Also, n_i can be directly monitored in practice. It needs to be noted that the proposed load balancing schemes do not assume any particular distribution of each of the system parameters.

3 Performance Measures on a Workstation

In this section, certain (performance) measures on a single workstation, to be used in the proposed load balancing schemes for multiple workstations, are derived.

The job of which execution time is to be minimized is referred to as *target job*. When the load characteristics (more specifically, the means and standard deviations of the system parameters) do not vary with time on a workstation, it is said that the workstation is in "steady state". When they vary with time, the workstation is said to be in "dynamic state".

3.1 Number of Jobs

The number of jobs, $n^{(j)}$, may be related to the job arrival rate, $a^{(j)}$, as follows.

$$n^{(j)} = 1 + a^{(j-1)} + (n^{(j-1)} - 1)p^{(j-1)} \tag{1}$$

where $p^{(j-1)}$ is the probability that a job in the $(j-1)th$ interval is carried over to the jth interval and the first term (of 1) corresponds to the target job.

Noting that $E[n^{(j)}] = E[n^{(j-1)}] = N$, and letting P denote the steady state value of $p^{(j)}$ which depends on μ and the distributions of $a^{(j)}$ and $s^{(j)}$, from Equation 1,

$$N = 1 + \frac{A}{1 - P} \tag{2}$$

Also, the standard deviation of $n^{(j)}$ can be derived as follows.

$$\sigma_n = \sqrt{E[(n^{(j)} - N)^2]} = \frac{\sigma_a}{1 - P} \tag{3}$$

3.2 Execution Time

In the jth interval, the target job (any job) is processed by the amount of $\frac{\mu}{n^{(j)}}$ for all j. If it takes t intervals to complete the target job,

$$\sum_{i=1}^{t} \frac{\mu}{n^{(i)}} = X. \tag{4}$$

Let's express $n^{(j)}$ as $N + \Delta n^{(j)}$. Then, $E[\Delta n^{(j)}] = 0$ since $E[n^{(j)}] = N$, and $E[(\Delta n^{(j)})^2] = \sigma_n^2$. Then, $\frac{1}{n^{(j)}}$ can be approximated by ignoring the higher order terms beyond the second order term as follows.

$$\frac{1}{n^{(j)}} = \frac{1}{N + \Delta n^{(j)}} \approx \frac{1}{N}\left(1 - \frac{\Delta n^{(j)}}{N} + \left(\frac{\Delta n^{(j)}}{N}\right)^2\right) \tag{5}$$

By taking $E[\ ]$ (expectation) on both sides of Equation 4 with Equation 5 incorporated into, the mean of execution time of the target job, T, can be derived.

$$T = \frac{NX}{\mu\left(1 + \frac{\sigma_n^2}{N^2}\right)} \tag{6}$$

Note that T depends on not only N but also σ_n both of which in turn depend on the standard deviations as well as the means of the system parameters, A, σ_a, S, and σ_s (and of course μ). Note that execution time of a target job on a workstation with a round-robin job scheduling decreases as variation (σ_n) in the number of jobs increases.

In order to derive the standard deviation of execution time, let $\Delta X^{(j)}$ denote a portion of X that is processed (completed) in the jth interval. Then $\Delta X^{(j)} = \frac{\mu}{N + \Delta n^{(j)}}$. Following the similar approximation used to obtain T, the standard deviation, $\sigma_{\Delta X}$, of ΔX can be shown to be $\frac{\mu\sigma_n}{N^2}$. Now, assuming "uncorrelatedness" of $\Delta X^{(j)}$ between intervals, the standard deviation, σ_X, of the amount of target job processed over T intervals (which is the mean execution time of the target job) can be easily shown to be $\sqrt{T}\sigma_{\Delta X}$. Finally, the standard deviation of execution time of a target job may be derived (approximated) by dividing σ_X by the mean processing speed which is $\frac{X}{T}$ and using Equation 6. That is,

$$\sigma_t = \frac{\sigma_X T}{X} = \sqrt{T}\frac{\frac{\sigma_n}{N}}{\sqrt{1 + \frac{\sigma_n^2}{N^2}}} \tag{7}$$

4 Load Balancing over Heterogeneous Workstations

4.1 Static Load Balancing

In the proposed static load balancing scheme, a target job is partitioned such that the fraction of X assigned to W_i for $i = 1, \cdots, W$ is inversely proportional to the expected execution time of X on W_i where W is the number of workstations available for X. Let T_i denote execution time of the target job on W_i (i.e., when W_i only is employed for the entire X). Then,

$$T_i = \frac{N_i X}{\mu_i\left(1 + \frac{\sigma_{n_i}^2}{N_i^2}\right)} \quad \text{for} \quad i = 1, \cdots, W. \tag{8}$$

Let X_i denote the size of the portion of X to be assigned to W_i. Then, X_i is determined as follows.

$$X_i = \frac{\frac{X}{T_i}}{\sum_{i=1}^{W}\frac{1}{T_i}} \tag{9}$$

Note that, even when N_i is the same for all i, X would not be distributed evenly unless σ_{n_i} is constant for all i. This load balancing strategy assigns more work to a workstation with a

larger variation in the number of jobs on it when the average number of jobs is the same for all workstations. Suppose that $N_1 = N_2 = 2$, $\sigma_{n_1} = 0$, and $\sigma_{n_2} > 0$ (say, n_2 alternates between 1 and 3). Then, a target job would be processed at the average rate of $\frac{\mu}{2}$ on W_1 while at the average rate of $\frac{\mu}{1} + \frac{\mu}{3} = \frac{2\mu}{3}$ on W_2. Therefore, a larger portion of the target is to be assigned to W_2 which has a larger variation in the number of jobs.

4.2 Dynamic Load Balancing

Two essential issues in dynamic load balancing are how load should be redistributed (redistribution of a target job) and how frequently load distribution is to be checked (determination of checking point).

Redistribution

Let $X_{i.rem}$ denote the size of the remaining portion of a target job on W_i before load balancing at a checking point. If load balancing (redistribution) is not done at the checking point, the expected completion time of the target job would be $max_i\{\ T_{i.rem}\ \}$ where $T_{i.rem}$ is computed using Equation 6. That is,

$$T_{i.rem} \ = \ \frac{N_i X_{i.rem}}{\mu_i \left(1 \ + \ \frac{\sigma_{n_i}^2}{N_i^2}\right)} \qquad \text{for} \ \ i = 1, \cdots, W \tag{10}$$

Now, if load balancing is to be done at the checking point, the total remaining job, of which size is $X_{rem} = \sum_I \ X_{i.rem}$, is redistributed over W_i according to Equation 9. That is, $X'_{i.rem} \ = \ \frac{\frac{X_{rem}}{T_{i.rem}}}{\sum_{i=1}^{W} \frac{1}{T_{i.rem}}}$ where $X'_{i.rem}$ is the size of target job (portion) to be assigned to W_i after load balancing. The expected execution time of $X'_{i.rem}$ on W_i is denoted by $T'_{i.rem}$. Then, note that $T'_{i.rem} \ = \ T'_{j.rem}$ for all i, j since load has been balanced.

The expected reduction, ΔT, in execution time of the target job can be expressed as

$$\Delta T \ = \ max_i\{\ T_{i.rem}\ \} \ - \ T'_{1.rem} \ - \ O_r \tag{11}$$

where O_r is the overhead for redistribution.

Load balancing (redistribution) is carried out only when the expected benefit (reduction in execution time, ΔT) exceeds a certain threshold, i.e., $\Delta T \ \geq \ Threshold$.

Determination of Checking Point

In order to minimize execution time of a target job, it is necessary to minimize the duration in which any W_i is idle (does not work on the target job) before the target job is completed on all

W_i. Hence, the load distribution is to be checked before any of W_i becomes idle while the others still work on the target job.

The standard deviation, $\sigma_{t'_{i.rem}}$, of execution time to complete $X'_{i.rem}$ can be derived using Equation 7. A reasonable estimate of the "highly likely" earliest completion time on W_i may be approximated to be $T'_{i.rem} - h\sigma_{t'_{i.rem}}$ where h is a tuning factor close to 1.

Then, the next checking point, T_{check}, measured with respect to the current checking point is set as follows.

$$T_{check} = min_i\{ T'_{i.rem} - h\sigma_{t'_{i.rem}} \} \tag{12}$$

That is, in the proposed dynamic load balancing scheme, it is attempted to check load distribution before any of W_i completes its execution of target job ($X'_{i.rem}$). Note that once a W_i completes $X'_{i.rem}$ it will not be utilized for the target job at least until next checking point.

5 Simulation Results and Discussion

An extensive computer simulation has been carried out in order to verify the reduction in execution time of a target job, which can be achieved by considering the second order moments (standard deviations) as well as the means of system parameters for load balancing on a network of heterogeneous workstations.

5.1 Simulation

In this simulation, three different distributions, i.e., exponential, uniform, and truncated Gaussian distributions, have been considered for each of the job inter-arrival time and the job size. Since similar trends have been observed for all three distributions, only the results for the (truncated) Gaussian distribution are provided in this paper. The proposed load balancing schemes have been tested for a wide range of each parameter. The program was run multiple times in each test case, each with a different seed, and then results (execution time of a target job) were averaged.

The proposed static and dynamic load balancing schemes ("S_P" and "D_P" which reads "static proposed" and "dynamic proposed", respectively) are compared to other approaches, i.e., "S_E" (Static_Even) which distributes a target job *evenly* in the static load balancing and "S_M" (Static_Mean) and "D_M" (Dynamic_Mean) which use only the means (N_i) of the number of jobs in the static and dynamic load balancing, respectively.

In the cases of dynamic load balancing, the overhead, O_c, for checking load distribution is added to the execution time for each checking point. If load is redistributed, O_r (redistribution overhead) is also added (for each redistribution).

5.2 Results and Discussion

Execution time is measured in intervals. In addition to execution time of a target job, the measure of "relative improvement" is used in comparison, which is defined as $RI\_S = \frac{T_{S\_M} - T_{S\_P}}{T_{S\_M}}$ for the static load balancing where T_{S_M} and T_{S_P} are execution times of a target job achieved by S_M and S_P, respectively. Similarly, the relative improvement is defined for the dynamic load balancing as $RI\_D = \frac{T_{D\_M} - T_{D\_P}}{T_{D\_M}}$ where T_{D_M} and T_{D_P} are execution times of a target job achieved by D_M and D_P, respectively.

Results for cases where workstations are in the steady state are discussed first.

In Figure 1, execution time of a target job on a single workstation is plotted as a function of σ_n for different N in a steady state. Obviously, T increases as N increases. More importantly, as discussed in Section 3.2, T clearly shows its dependency on σ_n and decreases as σ_n increases.

In Figure 2, (parallel) execution time of a target job in steady states on two workstations is compared among the five load balancing schemes mentioned above. First, it can be seen that, as expected, the proposed load balancing schemes (S_P and D_P) work better than the other schemes, confirming that the second order moments of the system parameters are to be considered in load balancing in order to minimize the execution time of a target job. Second, it needs to be noted that S_P achieves shorter execution times than D_P. This is due to the fact that in a steady state the load characteristics do not vary in the long term and therefore the one-time initial load balancing (by S_P) is good enough, and that D_P pays the overhead of checking and balancing during execution. Third, an increase in σ_a leads to a shorter execution time (Figure 2-(a)) while that in σ_s to a longer execution time (Figure 2-(b)). This is because an increase in σ_a causes a larger increase in σ_n than in N, leading to a shorter execution time (refer to Equation 6). However, increasing σ_s has an opposite effect.

In Figure 3, dependency of execution time on the load checking and redistribution overheads is analyzed in a steady state. As shown in the figure, when the overheads are relatively low, D_P can still achieve a shorter execution time than that by S_P. However, as the overheads become larger, they start to offset the gain by the dynamic load balancing and eventually make D_P perform worse than S_P.

The relative improvement by S_P over S_M is considered in Figure 4-(a), and that by D_P over D_M in Figure 4-(b). It can be seen in both cases that the relative improvement increases as the difference in σ_a or σ_s between two workstations becomes larger. This is due to the fact that the larger the difference is, the less accurate the load balancing by S_M becomes.

Effects of the number of workstations, W, are analyzed for steady states in Figure 5 where σ_{a_gi} and σ_{s_si} are σ_a and σ_s of the ith group of workstations. It can be observed that the relative improvement by S_P over S_M increases with the number of workstations. It increases more rapidly when the difference in either σ_a or σ_s is larger. Again, these observations stem from the fact that the load balancing by S_P becomes more (relatively) accurate than that by S_M as the difference in the second order moments between groups of workstations grows.

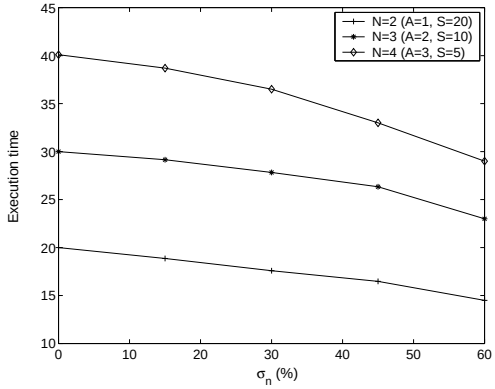
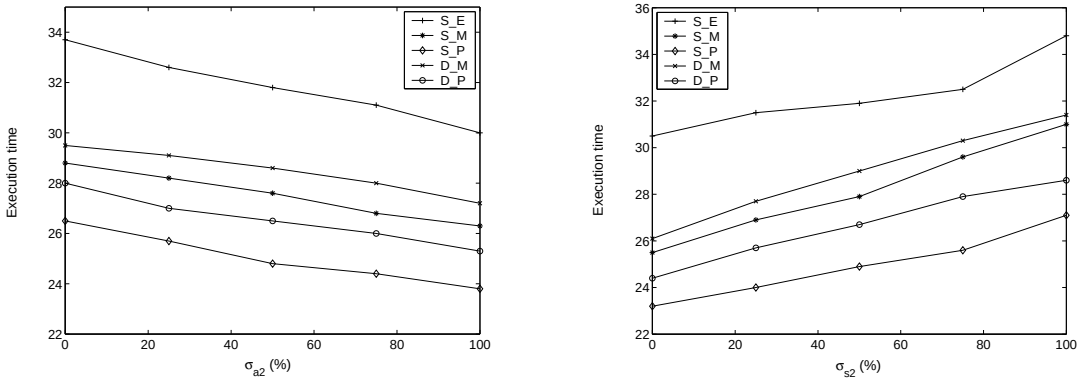


Figure 1: Execution time of X on one workstation where μ=100 and X=1000.



(a) (b)

Figure 2: Parallel execution time on two workstations where μ=100, A_1=1, A_2=2, S_1=10, S_2=20, and X=2000. (a) σ_{a_1}=53%, σ_{s_1}=30%, σ_{s_2}=48%, (b) σ_{a_1}=53%, σ_{a_2}=45%, σ_{s_1}=30%

Now, a case where a system parameter varies with time is considered, i.e. dynamic state. In Figure 6, σ_{a_1} varies with time such that its deviation from the value used by S_P is changed (larger for Case i with larger i). As expected, the dynamic schemes (D_P and D_M) perform better than the static schemes (S_P and S_M). Also, it is noted that D_P which takes σ_{a_1} into account achieves a shorter execution time compared to D_M. The improvement by D_P over D_M tends to increase with the deviation.

6 Conclusion and Future Study

In this paper, it has been proposed that the second order moments (standard deviations) as well as the first order moments (means) of system parameters be taken into account for load balancing on a time-shared heterogeneous parallel/distributed computing environment. These load balancing schemes which attempt to minimize execution time of a target job have been tested via computer simulation. It has been verified that considering the second order moments also in both static and

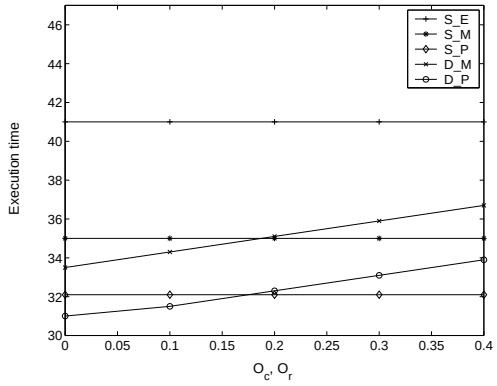
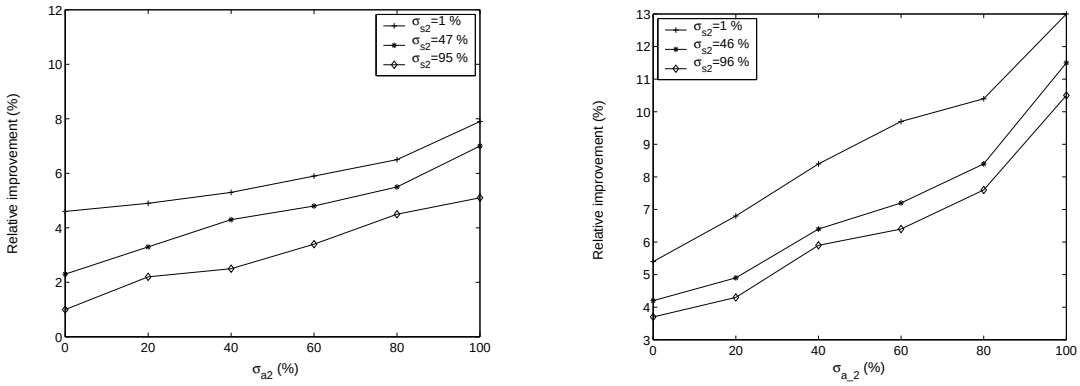


Figure 3: Parallel execution time on two workstations where μ=100, A_1=1, A_2=2, S_1=20, S_2=30, X=2000, σ_{a_1}=90%, σ_{a_2}=0%, σ_{s_1}=96%, σ_{s_2}=96%.



(a) (b)

Figure 4: (a) Relative improvement, RI_S, by S_P over S_M on two workstations. μ=100, A_1=0.5, A_2=0.5, S_1=40, S_2=40, X=3000, σ_{a_1}=0%, σ_{s_1}=100%, (b) Relative improvement, RI_D, by D_P over D_M on two workstations. μ=100, A_1=2, A_2=2, S_1=20, S_2=20, X=16000, σ_{a_1}=0%, σ_{s_1}=96%, O_c =0.1, O_r=0.1.

dynamic load balancing can lead to a significant reduction in execution time of a target job on a NOWs with a round-robin job scheduling policy adopted in each workstation. The improvement (reduction in execution time of a target job) becomes larger as the difference in the second order moments between workstations or groups of workstations increases. The similar observations have been made for all of the three distributions considered for each system parameter. The proposed schemes are simple and general, and therefore are believed to have a good potential for wide application.

The future study includes consideration of communication among subtasks and job granularity, performance analysis for real workloads on a NOWs, etc.

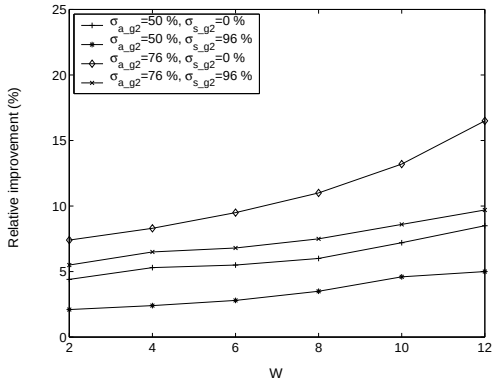


Figure 5: Relative improvement, RI_S, by S_P over S_M on multiple workstations where μ=100, A_{g1}=0.5, A_{g2}=0.5, S_{g1}=40, S_{g2}=40, X=1500, σ_{a_g1}=50%, σ_{s_g1}=96%.

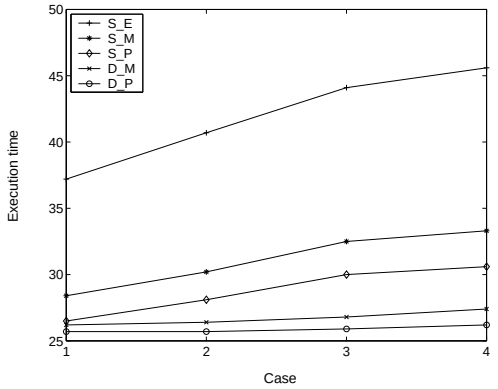


Figure 6: Parallel execution time on two workstations where μ=100, A_1=1, A_2=2, S_1=20, S_2=30, X=2000, σ_{a_1}=24%, σ_{a_2}=0%, σ_{s_1}=0%, σ_{s_2}=20%. σ_{a_1} varies with time where its deviation from the value used by S_P is larger for Case i with larger i.

References

[1] D. Culler, "Parallel Computer Architecture", Morgan Kaufman, 1999.

[2] Pf, "In Search of Clusters".

[3] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multiprocessors", *J. Parallel and Distributed Computing*, vol. 7, pp279-301, 1989.

[4] C. Polychronopoulos and D. Kuck, "Guided Self-Scheduling Scheme for Parallel Supercomputers", *IEEE Transactions on Computers*, vol. 36, no.12, pp1,425-1,439, December 1987.

[5] S. Ranka, Y. Won, and S. Sahni, "Programming a Hypercube Multicomputer", *IEEE Software*, pp69-77, September 1988.

[6] M. Maheswaran and H. Siegel, "A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems", *Proc. Heterogeneous Computing '98*, pp57-69, 1998.

[7] M. Cierniak, W. Li, and M.J. Zaki, "Loop Scheduling for Heterogeneity", *Proc. of the 4th IEEE International Symposium High-Performance Distributed Computing*, pp78-85, August 1995.

[8] A. Gerasoulis and T. Yang, "On the Granularity and Clustering of Directed Acyclic task graphs", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no.6, pp686-701, 1993.

[9] S. M. Figueira and F. Berman, "Modeling the Slowdown of Data-Parallel Applications in Homogeneous Clusters of Workstations", *Proc. Heterogeneous Computer Workshop*, pp90-101, 1998.

[10] J.C. Jacob and S.-Y. Lee, "Task Spreading and Shrinking on Multiprocessor Systems and Networks of Workstations", *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 10, pp1082-1101, October 1999.

[11] E.P. Makatos and T.J. Leblanc, "Using Processor Affinity in Loop Scheduling on Shared-Memory Multiprocessors", *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no.4, pp379-400, April 1993.

[12] S. Subramaniam and D.L. Eager, "Affinity Scheduling of Unbalanced Workloads", *Proc. Supercomputing '94*, pp214-226, 1994.

[13] M.-Y. Wu, "On Runtime Parallel Scheduling for Processor Load Balancing", *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no.2, pp173-186, February 1997.

[14] X. Zhang and Y. Yan, "Modeling and Characterizing Parallel Computing Performance On Heterogeneous Networks of Workstations", *Proc. of the 7th IEEE Symp. Parallel and Distributed Processing*, pp25-34, October 1995.

[15] B.-R. Tsai and K. G. Shin, "Communication-Oriented Assignment of Task Modules in Hypercube Multicomputers", *Proc. of the 12th International Conference on Distributed Computing Systems*, pp 38-45, 1992.