Scheduling in an Environment of Multiple Job Submission

Kwangil Ko and
Thomas G. Robertazzi
Dept. of Electrical and Computer
Engineering
University at Stony Brook
Stony Brook, NY 11794
Phone: 631-632-8412/8400
Fax: 631-632-8494
e-mail: tom@ece.sunysb.edu

Abstract — **The widespread interconnection of computers or workstations is due to high performance, availability, and extensibility at low cost. These distributed systems can be utilized for file sharing, multiuser access and parallel processing. Potential advantages of distributed systems have been studied by many researchers, often through simulation.**

In this paper we combine two tractable performance evaluation methodologies, Markovian queueing theory and divisible (i.e. partitionable) scheduling theory [1], to model a load sharing environment. The environment consists of N computers (nodes). A given load is solved on a subset of the nodes.

In this work we determine:

- **Solution time for a single queue model.**
- **The optimal number of processors to distribute a particular load to. This is done using group performance measures such as the equivalent processing speed and average utilization of a group of processors.**
- **This optimization is done for both homogeneous and heterogeneous computational environments.**
- **Finally an algorithm to balance load across subsets of computers is proposed.**

This work is noteworthy because of the integration of Markovian queueing statistics and the associated analytical expressions. Much past work has relied on simulation.

I. Introduction

One of the important topics in distributed systems for multiple users is load sharing. The purpose of load sharing is to balance loads over nodes in a distributed system, although loads arrive according to some irregular pattern. This improves the overall performance of distributed systems. A taxonomy of load sharing algorithms dividing them between source-initiative and server-initiative algorithm was proposed and the algorithms were evaluated using both analysis and simulation [2], [3]. Load sharing is also classified according to the level of information dependency. With more information one expects better performance. However, there is a trade-off between performance and communication overhead cost. Two important strategies classified according to the level of information are the static strategy and the adaptive strategy. A static strategy is independent of any system state whereas an adaptive strategy uses and reacts to the system state.

Another significant part of distributed systems is parallel processing which leads to improvements of speedup. In this work, static load sharing strategy is discussed which takes advantage of parallel processing.

A queueing model for multiple jobs is proposed in chapter 12 of [1]. That is, loads are assumed to arrive at a control processor or one processor. In this paper, each node has a queue and can accept loads. Serveral nodes in a network makes a set and process a job together. The number of nodes in a set is discussed to obtain a best overall performance in the sense of response time. As discussed in Chap. 3 of [5], the number of nodes in a set never exceeds the number of nodes which achieves speedup saturation. The number of nodes in a network is also assumed to be much larger than the number of nodes which achieves speedup saturation.

II. System Model

A completely connected network is considered for the load scheduling problem proposed in this paper. Assume that $(M+1)$ nodes are completely connected to each other. A node consists of a processor, a queue and M links. The inverse communication speed between the i^{th} node and the j^{th} node is defined as $z_{i,j}$. It is assumed that $z_{i,j} = z_{j,i}$ $(i,j = 0,1,\cdots,M)$ and $z_{i,i} = 0$ $(i = 0,1,\cdots,M)$, that is, the inverse communication speed between two nodes is symmetric and the communication delay in the internal node is ignored.

A node can be individually analyzed using an M/M/1 model. The Poisson average arrival rate of loads to the i^{th} node is defined as λ_i. Load is thus not necessarily uniform. The total arrival rate in a network is denoted as λ^T.

$$\lambda^T = \sum_{i=0}^{M} \lambda_i \tag{1}$$

Also load size is assumed to be negative exponentially distributed with mean size, $\bar{l}$. Let the processing time depend on the load size. Then, the average processing time of the i^{th} node, μ_i^{-1}, can be obtained by multiplying mean size, $\bar{l}$ by computation speed, $w_i T_{cp}$.

$$\mu_i^{-1} = \bar{l} w_i T_{cp} \tag{2}$$

The utilization factor is the ratio of the average arrival rate to the average processing rate in a fundamental sense. The utilization factor, ρ_i, is defined as follows:

$$\begin{aligned} \rho_i &= \frac{\lambda_i}{\mu_i} \\ &= \lambda_i \cdot \bar{l} w_i T_{cp} \end{aligned} \tag{3}$$

The average of ρ_i for all $i = 0, 1, \cdots, M$ is denoted as $\overline{\rho}$ and can be obtained as follows:

$$\overline{\rho} = \frac{1}{M+1} \sum_{i=0}^{M} \rho_i \tag{4}$$

The response time of the i^{th} node is defined as T_{p_i} and obtained by summing the time to wait in the i^{th} node queue and the time to be processed.

$$T_{p_i} = T_{q_i} + \mu_i^{-1} \tag{5}$$

Here, T_{q_i} is defined as the wait time in the i^{th} node queue. Further, the wait time in the i^{th} node queue can be obtained by multiplying the mean number in the i^{th} node queue by the mean processing time of the i^{th} node.

$$T_{q_i} = q_i \cdot \mu_i^{-1} \tag{6}$$

Here μ_i^{-1} is the mean processing time of the i^{th} node and q_i is the mean number in the i^{th} node queue. From Markovian queueing theory [4], the mean number in the i^{th} node queue can be expressed as follows:

$$q_i = \frac{\rho_i}{1-\rho_i} \tag{7}$$

Using equation (6) and (7), the response time of the i^{th} node in equation (5) can be expressed as follows:

$$\begin{aligned} T_{p_i} &= q_i \cdot \mu_i^{-1} + \mu_i^{-1} && (8) \\ &= \left(\frac{\rho_i}{1-\rho_i} + 1 \right) \cdot \mu_i^{-1} && (9) \\ &= \frac{\mu_i^{-1}}{1-\rho_i} && (10) \end{aligned}$$

Now the average response time in a network for a individual queueing model is obtained with respect to a weighted average sum.

$$\begin{aligned} \overline{T}_R^{ind} &= \frac{1}{\sum_{i=0}^{M} \lambda_i} \sum_{i=0}^{M} T_{p_i} \cdot \lambda_i && (11) \\ &= \frac{1}{\sum_{i=0}^{M} \lambda_i} \sum_{i=0}^{M} \frac{\mu_i^{-1}}{1-\rho_i} \cdot \lambda_i && (12) \\ &= \frac{1}{\sum_{i=0}^{M} \lambda_i} \sum_{i=0}^{M} \left[\frac{\rho_i}{1-\rho_i} \right] && (13) \end{aligned}$$

Further the bracket in the above equation is the mean number in the i^{th} node queue in equation (7). Thus, the average response time in a network can be obtained by dividing the sum of q_i's by the total arrival rate, λ^T in equation (1).

$$\overline{T}_R^{ind} = \frac{\sum_{i=0}^{M} q_i}{\lambda^T} \tag{14}$$

The above equation is the response time averaged over all queues and arrivals assuming Markovian statistics. That is, each node is assumed to process its own load. Each node is regarded as an M/M/1 queueing system if nodes don't share any load. Thus, some nodes can have heavy loads, and some nodes can have light loads. This may causes worse overall performance. Thus we seek to load share to improve response time.

III. Divisible Load Scheduling in a Set

The load sharing proposed here is not global in nature. Instead sets of $(K(j)+1)$ nodes in the completely connected network constitute a set in which nodes can share loads with each other. This means that all nodes in a set participate in processing any load arriving at nodes in a set. The arrival rate at the i^{th} node in the j^{th} set is denoted as $\lambda_{i,j}$. In addition, $\lambda'_{i,j}$ is the sum of arrival rates from all of the nodes to the i^{th} node in the j^{th} set.

$$\lambda'_{i,j} = \sum_{\substack{k=0 \\ k \neq i}}^{K(j)} \lambda_{k,j} \tag{15}$$

In this section, divisible load scheduling in a set is presented. Eventually the inverse equivalent node processing speed is obtained.

A number of parameters to be used are defined as follows:

- $p_{i,j}$: The i^{th} node in the j^{th} set.
- $\alpha_{i,j}$: The fraction of the entire processing load assigned to the i^{th} node in the j^{th} set.
- $w_{i,j}$: A constant inversely proportional to the computation speed of the i^{th} node in the j^{th} set.
- $z_{i,r}$: A constant inversely proportional to the channel speed of link between the i^{th} node and the r^{th} node.
- T_{cp}: Computing intensity constant. The entire load is processed in $w_{i,j}T_{cp}$ seconds by the i^{th} node in the j^{th} set.
- T_{cm}: Communication intensity constant. The entire load can be transmitted in $z_{i,r}T_{cm}$ seconds over the link between the i^{th} node and the r^{th} node.
- T_{cm}^{sol}: Solution reporting communication intensity constant. The entire solution report can be transmitted in $z_{i,r}T_{cm}^{sol}$ seconds over the link between the i^{th} node and the r^{th} node.

A homogeneous network is assumed:

$$\begin{aligned} w_{i,j} &= w && (16) \\ z_{i,r} &= z && (17) \end{aligned}$$

- σ : The ratio of the communication time to processing time.

$$\sigma = \frac{zT_{cm}}{wT_{cp}} \tag{18}$$

- σ^{sol} : The ratio of the reporting time to processing time.

$$\sigma^{sol} = \frac{zT_{cm}^{sol}}{wT_{cp}} \tag{19}$$

It is assumed that a load arrives at $p_{r,j}$, the r^{th} node in the j^{th} set for any $r = 0, 1, \cdots, K(j)$ and any $j = 1, \cdots, H$. Here, $K(j)+1$ is the number of nodes that belongs to the j^{th} set, and H the number of sets in a completely connected network. Then, $p_{r,j}$ at which a load arrives, becomes the originator (root) processor and begins to distribute fractions to $K(j)$ nodes in its set. Furthermore, the root processor and children processors have front-end processors so that communication and computation may be processed simultaneously.

The originator node, $p_{r,j}$ distributes fractions to $K(j)$ nodes from $p_{0,j}$ to $p_{K(j),j}$ in sequence skipping $p_{r,j}$. Let n

and $p'_{n,j}$ be the index and the n^{th} fraction reception node with respect to the distribution sequence, respectively: That is the index, i in $p_{i,j}$ for $i = 0, 1, \cdots, K(j)$ means a processor identification number but the index n in $p'_{n,j}$ for $n = 0, 1, \cdots, K(j)$ means the n^{th} receiving node. Thus, $p'_{n,j}$, the n^{th} fraction receiving node, is $p_{n-1,j}$ for $1 \leq n \leq r$. Also $p'_{n,j}$ is $p_{n,j}$ for $r < n \leq K(j)$. Further $p'_{0,j}$ or $p_{r,j}$ is the originator (root) node which is $p_{r,j}$.

$$p'_{n,j} = \begin{cases} p_{r,j} & \text{if } n = 0 \\ p_{n-1,j} & \text{if } 1 \leq n \leq r \\ p_{n,j} & \text{if } r < n \leq K(j) \end{cases} \tag{20}$$

Furthermore, the fraction of the entire processing load assigned to the n^{th} reception node in the j^{th} set is defined as $\alpha'_{n,j}$. Also $w'_{n,j}$ and $z'_{n,0}$ are relative to $p'_{n,j}$. Here the primed terms represent the actual distribution sequence. The subscription, 0 in $z'_{n,0}$ denotes the originator node. Thus links between the originator node and the n^{th} node for $n = 1, \cdots, K(j)$ will be used.

For homogeneous processor and link speed, solutions are reported in the same sequential processor order that load is received (see Chap. 2 of [5]). The equation for the originator node is expressed as follows:

$$\alpha'_{0,j} = \frac{\sigma \sum_{n=1}^{K(j)} X^{K(j)-n} + 1 + \sigma^{sol}}{(1+\sigma) \sum_{n=1}^{K(j)} X^{K(j)-n} + 1 + \sigma^{sol}} \tag{21}$$

For heterogeneous processor and link speeds, solutions are reported in the reverse sequential processor order that load is received in (see Chap. 2 of [5]).

The load fraction, $\alpha'_{0,j}$ for node r is obtained as follows:

$$\alpha'_{0,j} = \frac{\prod_{m=0}^{K(j)-1} Y_m}{\sum_{n=0}^{K(j)-1} \prod_{m=n}^{K(j)-1} Y_m + 1} \tag{22}$$

Once the load fraction assigned to the originator node is obtained, the finish time of $(K(j)+1)$ nodes can be expressed as follows when a load is arrives at $p_{r,j}$:

$$T_f(K(j))|_{\text{a load at } p_{r,j}} = \alpha'_{0,j} w'_{0,j} T_{cp} \tag{23}$$

Speedup is the ratio of the time to process a load by one node to the time taken to process the same load by $(K(j)+1)$ nodes. The speedup of $(K(j)+1)$ nodes is defined as $S(K(j))$.

$$\begin{aligned} S(K(j))|_{\text{a load at } p_{r,j}} &= \frac{w'_{0,j} T_{cp}}{T_f(K(j))|_{\text{a load at } p_{r,j}}} \\ &= \frac{1}{\alpha'_{0,j}} \\ &= \frac{1}{\alpha_{r,j}} \end{aligned} \tag{24}$$

Now a set of $(K(j)+1)$ nodes can be regarded as one equivalent node in the sense of processing speed. The equivalent node processing speed when a load arrives at the r^{th} node in the j^{th} set is defined as follows:

$$w^{eq}_{r,j} = \frac{w_{r,j}}{S(K(j))|_{\text{a load at } p_{r,j}}} \tag{25}$$

$$= \alpha_{r,j} w_{r,j} \tag{26}$$

Here $\alpha_{r,j}$ is the load fraction assigned to the originator $p_{r,j}$ node at which a load arrives. Further $w_{r,j}$ is the inverse node speed of the originator node $p_{r,j}$. If $p_{r,j}$ doesn't share its arrived load, $\alpha_{r,j}$ is one. Thus in this case $w^{eq}_{r,j}$, the equivalent node processing speed is the same as $w_{r,j}$, the processing speed of $p_{r,j}$.

The object of load balancing is to balance loads over sets in the network in order to improve overall performance. Let H be the number of sets in the completely connected network. There exist H equivalent nodes in the completely connected network. A single set can be analyzed as an M/M/1 model. The $(K(j)+1)$ nodes in a set can simultaneously process one load which arrives to any node in the set. The arrival rate in a set is defined as λ^{set}_j for $j = 1, 2, \cdots, H$ and can be obtained by summing the individual node arrival rates in a set.

$$\lambda^{set}_j = \sum_{i=0}^{K(j)} \lambda_{i,j} \tag{27}$$

$$= \lambda_{i,j} + \lambda'_{i,j} \tag{28}$$

Here, $\lambda_{i,j}$ denotes the arrival rate at the i^{th} node in the j^{th} set.

Further, $(K(j)+1)$ nodes in a set can be regarded as one equivalent node. The average processing time of the equivalent node which represents nodes in a set, $\left(\mu^{set}_j\right)^{-1}$ can be expressed in terms of the inverse equivalent node processing speed, w^{set}_j:

$$\left(\mu^{set}_j\right)^{-1} = \bar{l} w^{set}_j T_{cp} \tag{29}$$

Here $\bar{l}$ is the mean size of the negative exponentially distributed load. The equivalent node processing speed can vary depending on which node in a set accepts a load arrival and distributes fractions to the $K(j)$ nodes in a set. Thus, it is assumed that the inverse equivalent set processing speed is obtained using a weighted sum.

$$w^{set}_j = \frac{1}{\sum_{i=0}^{K(j)} \lambda_{i,j}} \sum_{i=0}^{K(j)} \lambda_{i,j} \cdot w^{eq}_{i,,j} \tag{30}$$

Here, $w^{eq}_{i,,j}$ denotes the inverse equivalent node processing speed when a load arrives at the i^{th} node in the j^{th} set. This was obtained in equation (25). Equation (30) is substituted into (29), for:

$$\left(\mu^{set}_j\right)^{-1} = \frac{\bar{l} T_{cp}}{\sum_{i=0}^{K(j)} \lambda_{i,j}} \sum_{i=0}^{K(j)} \lambda_{i,j} \cdot w^{eq}_{i,,j} \tag{31}$$

Now, the utilization factor of a set is defined as follows:

$$\rho^{set}_j = \lambda^{set}_j \cdot \left(\mu^{set}_j\right)^{-1} \tag{32}$$

Equation (27) and (31) are substituted into the above equation.

$$\begin{aligned} \rho^{set}_j &= \sum_{i=0}^{K(j)} \lambda_{i,j} \cdot \frac{\bar{l} T_{cp}}{\sum_{i=0}^{K(j)} \lambda_{i,j}} \sum_{i=0}^{K(j)} \lambda_{i,j} \cdot w^{eq}_{i,,j} \\ &= \bar{l} T_{cp} \cdot \sum_{i=0}^{K(j)} \lambda_{i,j} \cdot w^{eq}_{i,,j} \end{aligned} \tag{33}$$

The above equation indicates that the utilization of the j^{th} set is the sum of the utilization of the i^{th} node for $i = 0, 1, \cdots, K(j)$ in the j^{th} set.

Let $\overline{\rho}^{set}$ be the average set utilization.

$$\overline{\rho}^{set} = \frac{1}{H}\sum_{j=1}^{H}\rho_j^{set} \tag{34}$$

Here H is the number of sets in a network. Equation (33) is substituted into the above equation.

$$\begin{aligned}\overline{\rho}^{set} &= \frac{\overline{l}T_{cp}}{H}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot w_{i,,j}^{eq} &(35)\\ &= \frac{\overline{l}T_{cp}}{H}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot\frac{w_{i,,j}}{S\left(K\left(j\right)\right)|_{\text{a load at } p_{r,j}}} &(36)\\ &= \frac{\overline{l}T_{cp}}{H}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot\alpha_{i,j}\cdot w_{i,,j} &(37)\end{aligned}$$

It is desired that each set is as close as possible to $\overline{\rho}^{set}$ in order to balance load over sets in a network.

IV. Load Balancing over Sets

Sets act like an equivalent node with utilization factor, ρ_j^{set} using divisible load scheduling. The load balancing over sets discussed in this section is a static strategy. The strategy is independent of any system state. That is, loads are balanced over sets using the arrival rates of nodes, λ_i's for all $i = 0, 1, \cdots, M$. Thus it is important to make a set whose utilization factor is close to those of the other sets.

The optimal number of processors in a set, K^*, is discussed to obtain the best performance in the sense of response time.

Originally the speedup can vary depending on which node in a set accepts a load arrival, particularly in a heterogeneous network. To estimate for the heterogeneous network, $\widehat{w}$ and $\widehat{z}$, the average inverse computation and communication speeds, let:

$$\begin{aligned}\widehat{w} &= \frac{1}{M+1}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}w_{i,j}\\ &= \frac{1}{M+1}\sum_{i=0}^{M}w_i &(38)\end{aligned}$$

and

$$\begin{aligned}\widehat{z} &= \frac{1}{M!}\sum_{j=1}^{H}\sum_{r=0}^{K(j)}\sum_{\substack{i=0\\ i\neq r}}^{K(j)}z_{i,r}\\ &= \frac{1}{M!}\sum_{r=0}^{M}\sum_{\substack{i=0\\ i\neq r}}^{M}z_{i,r} &(39)\end{aligned}$$

Then speedup is estimated as $\widehat{S}\left(K\right)$ which is function of K and is identical no matter which node a load arrives at. For a homogeneous network, $\widehat{w}$ and $\widehat{z}$ reduce w and z, respectively. The network parameters are defined as follows:

$$\begin{aligned}\widehat{\sigma} &= \frac{\widehat{z}T_{cm}}{\widehat{w}T_{cp}} &(40)\\ \widehat{\sigma}^{sol} &= \frac{\widehat{z}T_{cm}^{sol}}{\widehat{w}T_{cp}} &(41)\end{aligned}$$

Thus the estimated speedup can be obtained in a manner similar to that of a homogeneous network.

$$\widehat{S}\left(K\right) = \frac{\left(1+\widehat{\sigma}\right)\sum_{n=1}^{K}X^{K-n}+1+\widehat{\sigma}^{sol}}{\widehat{\sigma}\sum_{n=1}^{K}X^{K-n}+1+\widehat{\sigma}^{sol}} \tag{42}$$

Consequently the estimated version of $w_{r,j}^{eq}$ in equation (25) is expressed follows:

$$\widehat{w}^{eq} = \frac{\widehat{w}}{\widehat{S}\left(K\right)} \tag{43}$$

From equation (31), the estimated set processing time can be expressed as follows:

$$\begin{aligned}\left(\widehat{\mu}^{set}\right)^{-1} &= \frac{\overline{l}T_{cp}}{\sum_{i=0}^{K(j)}\lambda_{i,j}}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot\widehat{w}^{eq} &(44)\\ &= \frac{\overline{l}\widehat{w}T_{cp}}{\widehat{S}\left(K\right)} &(45)\end{aligned}$$

Now the equation (36) is estimated as follows:

$$\begin{aligned}\widehat{\rho}^{set} &= \frac{\overline{l}T_{cp}}{H}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot\frac{w_{i,j}}{\widehat{S}\left(K\right)} &(46)\\ &= \frac{1}{\widehat{S}\left(K\right)}\cdot\left[\frac{1}{H}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot\overline{l}w_{i,j}T_{cp}\right] &(47)\end{aligned}$$

The two summation in equation (47) denote all nodes in the completely connected network. The term in the bracket in this equation can be expressed as follows:

$$\frac{1}{H}\sum_{j=1}^{H}\sum_{i=0}^{K(j)}\lambda_{i,j}\cdot\overline{l}w_{i,j}T_{cp} = \frac{1}{H}\sum_{i=0}^{M}\lambda_i\cdot\overline{l}w_iT_{cp} \tag{48}$$

Here $\lambda_i\cdot\overline{l}w_iT_{cp}$ can be expressed as ρ_i from equation (3). From equation (4) the right side of the above equation can be expressed as follows:

$$\begin{aligned}\frac{1}{H}\sum_{i=0}^{M}\lambda_i\cdot\overline{l}w_iT_{cp} &= \frac{M+1}{H}\overline{\rho} &(49)\\ &= \left(K+1\right)\overline{\rho} &(50)\end{aligned}$$

Here, $\frac{M+1}{H}$ can be considered to be the number of nodes in a set. This can be expressed as $(K+1)$.

Therefore from above the estimated set utilization can be expressed as follows:

$$\widehat{\rho}^{set} = \frac{\left(K+1\right)\overline{\rho}}{\widehat{S}\left(K\right)} \tag{51}$$

Next we wish to find the optimal value of K^* nodes in the sense of response time. A set behaves like a node with equivalent node speed. We use the estimated speedup, equivalent set speed and set utilization factor obtained in equation (42), (45) and (51). The estimated response time in a set of $(K+1)$ nodes is defined as $\widehat{T}_{set}\left(K\right)$. Similarly in equation (10), $\widehat{T}_{set}\left(K\right)$ is expressed as follows:

$$\widehat{T}_{set}\left(K\right) = \left(\frac{1}{1-\widehat{\rho}^{set}}\right)\left(\widehat{\mu}_j^{set}\right)^{-1} \tag{52}$$

Equation (45) and (51) are substituted into the above equation.

$$\begin{aligned}\widehat{T}_{set}(K) &= \left(\frac{1}{1-\frac{(K+1)\overline{\rho}}{\widehat{S}(K)}}\right)\frac{\overline{l}\widehat{w}T_{cp}}{\widehat{S}(K)} \\ &= \frac{\overline{l}\widehat{w}T_{cp}}{\widehat{S}(K)-(K+1)\overline{\rho}}\end{aligned}$$

Here:

$$\widehat{S}(K) > (K+1)\overline{\rho} \tag{53}$$

To obtain the optimal value of K^* to minimize $\widehat{T}_{set}(K)$, a difference equation is used:

$$\widehat{T}_R(K^*) - \widehat{T}_R(K^*-1) = 0 \tag{54}$$

or

$$\left[\widehat{S}(K^*) - (K^*+1)\overline{\rho}\right] - \left[\widehat{S}(K^*-1) - K^*\overline{\rho}\right] = 0 \tag{55}$$

Obtaining K^* to hold the above equation, K^* can be expressed as follows (Appendix C in [5]).

- For $\widehat{\sigma} \neq \widehat{\sigma}^{sol}$:

$$\begin{aligned}K^* &= \left\lfloor 1 + \log_X \frac{1}{2}\left[\left(\overline{\rho}\widehat{\sigma}^2\right)^{-1}\left(\widehat{\sigma}^{sol}-\widehat{\sigma}\right)\left(X^{-1}-1\right)\right.\right. \\ &\quad \left.\left. +\widehat{\sigma}^{-1}\widehat{\sigma}^{sol}(X^{-1}+1) + \sqrt{A_1}\right]\right\rfloor \end{aligned} \tag{56}$$

- For $\widehat{\sigma} = \widehat{\sigma}^{sol}$:

$$K^* = \left\lfloor \frac{-\widehat{\sigma}\left[2-\widehat{\sigma}+2\widehat{\sigma}^{sol}\right] + \sqrt{A_2}}{2\widehat{\sigma}^2} \right\rfloor \tag{57}$$

Here A_1 and A_2 are given in the appendix. Here $\lfloor\cdot\rfloor$ is rounding down to the nearest integer. Although the number of nodes in a set is larger than the value before rounding down in the above equation, the response time in a set doesn't decrease.

The purpose of this section is to find the best choice for the number of sets, H^* in the sense of response time. Let (K^*+1) be the number of nodes in a set, which produces the best performance in the sense of response time. Thus the number of sets, H^* in the network can be obtained as follows:

$$H^* = \left\lceil \frac{M+1}{\lfloor K^* \rfloor + 1} \right\rceil \tag{58}$$

Here $\lceil\cdot\rceil$ is rounding up to the nearest integer in the case that $(M+1)$ is not divisible by $(\lfloor K^*\rfloor+1)$. Once the optimal number of a set is obtained by rounding up to the nearest integer the actual nodes in a set, $K(j)$ for $j = 1, 2, \cdots, H^*$ may be less than K^*. The reason to make $K(j)$ to be less than K^* is that the response time worsens $K(j)$ is made greater than K^*.

A set behaves like a node with an equivalent set processing speed, w_j^{set}, as in equation (30) under divisible load scheduling. In this section, an algorithm to balance load among nodes is proposed. This algorithm ensures that the response time for each set is close to each other. Load balancing over sets is a static strategy. The static strategy is independent of any system state. Once the arrival rates of nodes, λ_i's for all $i = 0, 1, \cdots, M$ are experimentally determined, the utilization factor of each node, ρ_i can be obtained. The response time of each node, T_{p_i} in equation (10) can be calculated with ρ_i and w_i. Let $\mathbf{\Gamma}_0$ be the descending sorting order of T_{p_i} for $i = 0, 1, \cdots, M$.

$$\mathbf{\Gamma}_0 = \left[\begin{array}{cccccc} T_{p^{(0)}} & \cdots & T_{p^{(i)}} & \cdots & T_{p^{(M)}} \end{array}\right] \tag{59}$$

Here $T_{p^{(i)}} \leq T_{p^{(j)}}$ if $i \leq j$. Further, $p^{(i)}$ is the processor with the $(i+1)^{th}$ largest response time.

Now the functions used in the algorithm are defined as follows:

- Let $\mathbf{S}$ be the set matrix. The processors in a column of $\mathbf{S}$ make a set.

$$\begin{aligned}\mathbf{S} &= \left[\begin{array}{cccc} p_{0,1} & p_{0,2} & \cdots & p_{0,H^*} \\ p_{1,1} & p_{1,2} & \cdots & p_{1,H^*} \\ \vdots & \vdots & \ddots & \vdots \\ p_{K,1} & p_{\overline{K},2} & \cdots & p_{\overline{K},H^*} \end{array}\right] \\ &= \left[\begin{array}{cccc} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_{H^*} \end{array}\right] \qquad (60) \\ &= \left[\begin{array}{c} \mathbf{r}_0 \\ \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{\overline{K}} \end{array}\right] \qquad (61)\end{aligned}$$

  Here, $\mathbf{s}_j$ is the j^{th} column which consists of nodes in the j^{th} set and $\mathbf{r}_i$ is the $(i+1)^{th}$

- Let $\mathbf{D}$ be the sum of the response time of processors in column of $\mathbf{S}$

$$\mathbf{D} = \left[\begin{array}{cccccc} d_1 & \cdots & d_j & \cdots & d_{H^*} \end{array}\right] \tag{62}$$

Here:

$$d_j = \sum_{i=0}^{\overline{K}} T_{p_{i,j}} \tag{63}$$

- Let $ISORT(\mathbf{D})$ return the index after sorting $\mathbf{D}$ in ascending order.
- Let $GETID\left(T_{p^{(i)}}\right)$ return the node identification number, p_i with respect to $p^{(i)}$.

Algorithm

STEP 1. Set $\overline{K} = \left\lceil \frac{M+1}{H^*} \right\rceil$.
STEP 2. Sort T_{p_i} for $i = 0, 1, \cdots, M$ in descending order.

$$\mathbf{\Gamma}_0 = \left[\begin{array}{cccccc} T_{p^{(0)}} & \cdots & T_{p^{(i)}} & \cdots & T_{p^{(M)}} \end{array}\right]$$

STEP 3. Initialize a set matrix.

$$\mathbf{S} = []$$

STEP 4. Set $k = 0$.
STEP 5. While $\mathbf{\Gamma}_k$ is not empty.
STEP a) Set $k = k+1$.
STEP b) If k is odd,
Take out the first H^* elements from $\mathbf{\Gamma}_{k-1}$ and then put them into $\mathbf{r}_k$.

$$\mathbf{\Gamma}_{k-1} = [\mathbf{t}_k \mid \mathbf{\Gamma}_k]$$

STEP d) Else,

Take out the last H^* elements from $\mathbf{\Gamma}_{k-1}$ and put them into $\mathbf{r}_k$.

$$\mathbf{\Gamma}_{k-1} = [\mathbf{\Gamma}_k \mid \mathbf{t}_k]$$

STEP g) Calculate $\mathbf{D}$.

STEP h) Set $\mathbf{I}_k = ISORT\,(\mathbf{D})$.

STEP i) Obtain $\mathbf{t}_k^{sort}$ from sorting $\mathbf{t}_k$ in terms of $\mathbf{I}_k$.

STEP j) Set $\mathbf{r}_k = GETID\left(\mathbf{t}_k^{sort}\right)$.

STEP 6. End

The above algorithm ensures that the response time over sets is as similar as possible. Similarly in equation (10), the response time of the j^{th} set can be expressed as follows:

$$T_{\mathbf{s}_j} = \frac{\left(\mu_j^{set}\right)^{-1}}{1-\rho_j^{set}} \tag{64}$$

Here $\left(\mu_j^{set}\right)^{-1}$ and ρ_j^{set} are defined in equation (31) and (33), respectively.

Then, the average response time in a network for a set queueing model is obtained as follows:

$$\overline{T}_R^{set} = \frac{1}{\sum_{j=1}^{H^*}\lambda_j^{set}} \sum_{j=1}^{H^*} T_{\mathbf{s}_j} \cdot \lambda_j^{set} \tag{65}$$

$$= \frac{1}{\sum_{j=1}^{H^*}\lambda_j^{set}} \sum_{j=1}^{H^*} \frac{\left(\mu_j^{set}\right)^{-1}}{1-\rho_j^{set}} \cdot \lambda_j^{set} \tag{66}$$

$$= \frac{1}{\lambda^T} \sum_{j=1}^{H^*} \frac{\rho_j^{set}}{1-\rho_j^{set}} \tag{67}$$

Here λ^T is the total arrival rate. The best response times are achived under optimal set size.

V. Conclusion

This chapter discusses the problem of load sharing in a multiprocessor environment where multiple jobs are submitted. The network is divided into sets of processors that load share within each other. Using a divisible load model and analysis, the optimal size of a set in terms of response time is found. Also found is the optimal allocation of load to processors for a given set size for both heterogeneous and homogeneous networks. Finally, expressions for response time for both the cases with and without load sharing are found. Simulation results indicate load sharing significantly reduces response time.

References

[1] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi, "Scheduling Divisible Loads in Parallel and Distributed Systems," *IEEE Computer Society Press,* Los Alamitos CA, 1996

[2] B. A. Shirazi, A. R. Hurson and K. M. Kavi, "Scheduling and Load Balancing in Parallel and Distributed System," *IEEE Computer Society Press,* 1996

[3] Y. Wang, R. J. T. Morris, "Load Sharing in Distributed Systems," *IEEE Transactions on Computers*, vol. c-34, pp. 204–217, Mar. 1985

[4] T. G. Robertazzi, "Computer Networks and Systems: Queueing Theory and Performance Evaluation," 2nd Edition *Springer-Verlag*, 1994,

[5] K. Ko, "Scheduling Data Intensive Parallel Processing in Distributed and Networked Environments", *State University of New York at Stony Brook,* Aug. 2000