Scheduling Computational Loads in Single Level Tree Network

Run Cui* S. Suresh**, and H. J. Kim**

Abstract

This paper is the introduction of our work on distributed load scheduling in single-level tree network. In this paper, we derive a new calculation model in single-level tree network and show a closed-form formulation of the time for computation system.There are so many examples of the application of this technology such as distributed database, biology computation on genus, grid computing, numerical computing, video and audio signal processing, etc.

Keywords : Distributed, loads, scheduling, single-level tree, scheduling

I. Introduction

In recently 20 year, distributed network computation is growing quite fast with the development ofthe internet. As the information processing scaling in the network is much larger than before, finishing the whole computing work in a single computer is becoming much harder than before. So the research of the load distributed computation has been one ofmost important work of the modern computer science. In this situation, the research work of the distributed load scheduling is a valuable pointer to the future work of the big amount of information processing.

II. Models and Notations

1. Single Level Tree

The single level tree network architecture which is shown in Figure 1 is the model used in this paper. Suppose there are m processors (P_1, $P_2 \cdots P_m$) directly connected to the root processor (P_0) with the links (L_0, $L_1 \ldots L_m$). N is a very big value presented the total size of send payloads to all the child nodes send payloads the payload to be processed. The root processor (P_0) owns all the payloads before the start of the computation and it can arbitrarily to all the child nodes (P_1, $P_2 \cdots P_m$) at the same time. The fraction sent to processor P_i is presented by α_i (i =0, 1⋯m), so the size of the payload of processor P_i is $\alpha_i N$. The root processor can begin computing at the beginning time because it owns all the payloads. The child processor P_i can begin to compute only after it gets all the payloads $\alpha_i N$ and there is no time gap between communication and computation process.

접수일자 : 2009년 8월 04일
최종완료 : 2009년 8월 20일
*Multimedia and information security Lab Korea University
**Korea Univsersity Multimedia and Information Security
교신저자, E-mail : khj-@korea.ac.kr

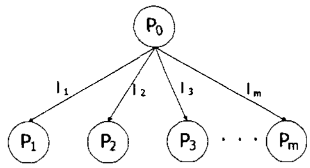


Figure 1 A single level tree

As α_i is fraction of the payloads we get the equation below:

$$\sum_{i=0}^{m} \alpha_i = 1 \quad (1)$$

$$\sum_{i=0}^{m} \alpha_i N = N \quad (2)$$

2. Collective Communication Model

The collective communication model is used in this paper. In this kind of model, the root processor P_0 can send the payloads to its child processors simultaneously while it is computing, which means P_0 can send the load fractions ($\alpha_1 N$, $\alpha_2 N$ ⋯ $\alpha_m N$) to child processors(P_1, $P_2 \cdots P_m$) at the same time. Suppose Ti is total time for processor P_i to finish its task. To child processors Ti is consisted by two parts: Communication time Ticomm and computing time Ticomp. But to P_0, as it does not need to receive the payload from other processors, T_0 is the total time used in computing. So Ti can be presented as follows:

$$T_i = \begin{cases} T_0^{comp}, i=0 \\ T_i^{comp} + T_i^{comm}, i \neq 0 \end{cases} \quad (3)$$

3. Computing models

The communication Ticomm time is the sum of constant start-up delay θ_{cm} and transmission time which is in direct proportion to the size of the deferent payload. We get the formulation below:

$$T_i^{comm} = \theta_{cm} + G_i(\alpha iN), i = 1,2,...,m \quad (4)$$

Where G_i is the time taken to read unit data from the buffer for processor Pi and $\alpha_i N$ is the size of the data received by P_i.

In practical applications, the computation complexity is nonlinear in problem size. In this paper, we consider two cases: polynomial computation model and a mixed model. Ai is the time for processor pitaken to process a unit data. The computing time Ticompis the sum a constant start-up delay θ_{cp} and the nonlinear computation time as follows:

$$T_i^{comp} = \theta_{cp} + A_i f(\alpha_i N), i = 0,1,...,m \quad (5)$$

Where f is a nonlinear function. In this paper, $f(\alpha_i N)$ can be substituted by $(\alpha_i N)^\gamma$ or $(\alpha_i N)+\log(\alpha_i N)$ in different case. In normal case, the start-up header θcm and θcp are very small value compared with other parts in the formulations and we always ignore them.

Now we get a computation model shown in figure 2. From the processing model we can get the computing model as follows:

$$T_i = \begin{cases} A_0 f(\alpha_0 N), i = 0 \\ G_i f(\alpha_i N) + A_i f(\alpha_i N), i = 1,...,m \end{cases} \quad (6)$$

$$T_{\min} = \min(T_i), i = 0,1,...,m \quad (7)$$

In this paper, we will ignore the start-up delay, θ_{cm} and θ_{cp}.

4. Notations and Definitions

Notations:

N: The total size of the processing payload.

m: Number of the child processors.

α_i: Fractions of the processing load assigned to processor Pi.

A_i: The computation time speed parameter for processor Pi.

G_i: The communication time parameter for processor P_i.

θ_{cm}: A constant additive communication overhead component that includes the sum of all delays associated with the communication process.

θ_{cp}: A constant additive computation overhead component that includes the sum of all delays associated with the communication process.

T_i^{comm}: Time for processor Pi to send the payload

T_i^{comp}: Time for processor Pi to computing the entire assigned payload.

Definitions:

T_i: The total time for processor P_i to finish its task, including communication time and computing time.

T_{min}: The minimum processing time among T_i.

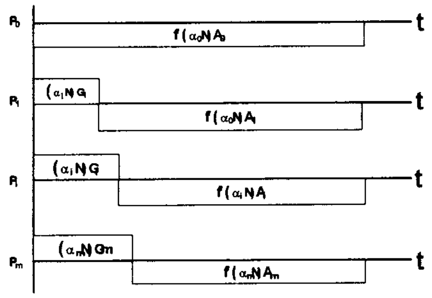


Figure 2 Processing time model

III. Closed-Form Expression for Processing Time

In this section, we will derive closed-form expression for different models. As shown in the computing model, P0can send load to the child processors simultaneously at time t = 0. In divisible load theory, it has been rigorously proved that for optimal processing time, all the processing involved the computation of the processing load must stop computing at the same time instant. Associated with the computing model in figure 2, we can get the formulation below:

$$f(\alpha_0 N)A_0 = f(\alpha_i N)A_i + (\alpha_i N)G_i, \quad i = 1,2...,m \quad (8)$$

Denoting $f_i = \frac{A_{i-1}}{A_i}$ and $\beta_i = \frac{G_i}{A_i}$ for all i = 1, 2··· m. Equation (8) can rewritten as:

$$f(\alpha_i N) + (\alpha_i N)\beta_i - f(\alpha_0 N)\prod_{k=1}^{i} f_k = 0, \quad i = 1,2...,m \quad (9)$$

And solution of α_i (I = 0, 1··· m) in equation set consist of equation (2) and (8) is optimal solution for load assignment.

1. Polynomial solution for αiN

In this case, $f(\alpha_i N)=(\alpha_i N)^r$, so the computing model in equation (9) can be rewritten as below:

$$(\alpha_i N)^{\gamma} + (\alpha_i N)\beta_i - (\alpha_0 N)^{\gamma}\prod_{k=1}^{i} f_k = 0,$$
$$i = 1,2...,m \tag{10}$$

Some results and derivation comes from the previews work, and the result for higher order is our new work result for high speed processor case.

For ɣ = 1:

Equation (10) can be expressed as:

$$(\alpha_i N) + (\alpha_i N)\beta_i - (\alpha_0 N)\prod_{k=1}^{i} f_k = 0$$
$$i = 1,2...,m \tag{11}$$

The solution of equation (9) can be written as follows:

$$\alpha_i N = \frac{\prod_{k=1}^{i} f_k(\alpha_i N)}{1+\beta_i} = \prod_{k=1}^{i} f_k(\alpha_i N) - \frac{\beta_i}{(1+\beta_i)(\prod_{k=1}^{i} f_k(\alpha_i N))^{-1}}$$
$$i = 1,2,...,m \tag{12}$$

For ɣ = 2:

Equation (10) can be expressed as:

$$(\alpha_i N)^2 + (\alpha_i N)\beta_i - (\alpha_0 N)^2\prod_{k=1}^{i} f_k = 0,$$
$$i = 1,2...,m \tag{13}$$

The solution of equation (13) is as follows:

$$\alpha_i N = \frac{-\beta_i + \sqrt{\beta_i^2 + 4\prod_{k=1}^{i} f_k(\alpha_i N)^2}}{2} \tag{14}$$

Here the negative root is ignored. Using the method introduced in reference [2], we can simple the solution by removing the radical signs:

$$\alpha_i N \approx (\prod_{k=1}^{i} f_k)^{1/2}(\alpha_0 N) - \frac{\beta_i}{2} = (\prod_{k=1}^{i} f_k)^{1/2}(\alpha_0 N) - \frac{\beta_i}{2[(\prod_{k=1}^{i} f_k)^{1/2}(\alpha_0 N)]^0} \tag{15}$$

For ɣ = 3:

Equation (10) is as follows:

$$(\alpha_i N)^3 + (\alpha_i N)\beta_i - (\alpha_0 N)^3\prod_{k=1}^{i} f_k = 0$$
$$i = 1,2...,m \tag{16}$$

The significative root of equation (16) is:

$$\alpha_i N = -\frac{(\frac{2}{3})^{1/3}}{(9\prod_{k=1}^{i} f_k(\alpha_0 N)^3 + \sqrt{3}\sqrt{4\beta_i^3 + 27(\prod_{k=1}^{i} f_k)^2(\alpha_0 N)^6})^{1/3}} + \frac{(9\prod_{k=1}^{i} f_k(\alpha_0 N)^3 + \sqrt{3}\sqrt{4\beta_i^3 + 27(\prod_{k=1}^{i} f_k)^2(\alpha_0 N)^6})^{1/3}}{2^{1/3}3^{2/3}} \tag{17}$$

Using the same as before, we can get a simplified solution as shown in equation (18):

$$\alpha_i N \approx (\prod_{k=1}^{i} f_k)^{1/3}(\alpha_0 N) - \frac{\beta_i}{3[(\prod_{k=1}^{i} f_k)^{1/3}(\alpha_0 N)]^{1/3}} \tag{18}$$

2. Polynomial solution for α0N

From the section 3.1.1, we get a solution of αiN for order 1 to 3, now we can devote the final solution of α0 with the equations (2) and result of αiN. Substitute the result to the equation (2), and for different case different solution of α0 will be got.In the previews work, order 1 and 2 have been got, now we mentioned them again to show the approach of getting the solution of order 3. That is the main distribution of our work.

For ɣ=1:

After the substitution we get the equation below:

$$\alpha_0 N + \sum_{i=1}^{m}((\prod_{k=1}^{i} f_k)(\alpha_0 N) - \frac{\beta_i}{(1+\beta_i)[(\prod_{k=1}^{i} f_k)(\alpha_0 N)]^{-1}}) = N \tag{19}$$

And equation (21) can be written as:

$$1 - A(m)_1(\alpha_0 N)^{-1} - B(m)_1 = 0 \tag{20}$$

Where A(m)1 and B(m)2 are the expressions as follows:

$$A(m)_1 = \frac{N}{1+\sum_{i=1}^{m}(\prod_{k=1}^{i} f_k)} \tag{21}$$

$$B(m)_1 = \frac{\sum_{i=1}^{m}\frac{\beta_i}{(1+\beta_i)(\prod_{k=1}^{i} f_k)^{-1}}}{1+\sum_{i=1}^{m}(\prod_{k=1}^{i} f_k)} \tag{22}$$

Considering the expressions of A(m) and B(m), we can see that A(m) is much bigger then B(m) because in practice case N is a big value and fi is small. With this condition we can simple the following equation as we do in section 3.1.1.

From equation (30) we get the solution for the case ɣ=1:

$$\alpha_0 N = \frac{A(m)_1}{1 - B(m)_1} \tag{23}$$

For ɣ=2:

Using the same method we can get that:

$$(\alpha_0 N) - A(m)_2 - B(m)_2 = 0 \quad (24)$$

And the root of this equation is:

$$(\alpha_0 N) = A(m)_2 + B(m)_2 \quad (25)$$

Here, the new expression for A(m) and B(m) is as follows:

$$A(m)_2 = \frac{N}{1+\sum_{i=1}^{m}(\prod_{k=1}^{i} f_k)^{1/2}} \quad (26)$$

$$B(m)_2 = \frac{\sum_{i=1}^{m}\frac{\beta_i}{2}}{1+\sum_{i=1}^{m}(\prod_{k=1}^{i} f_k)^{1/2}} \quad (27)$$

For $\gamma=3$:

Equation result of formulation above can be rewritten as:

$$(\alpha_0 N)^2 - A(m)_3(\alpha_0 N) - B(m)_3 = 0 \quad (28)$$

Using the same method which we use to get the approximate solution of α_i, and we can get the root below:

$$\alpha_0 N = \frac{A(m)_3 + \sqrt{A(m)_3^{\,2} + 4B(m)_3}}{2} \quad (29)$$

$$A(m)_3 = \frac{N}{1+\sum_{i=1}^{m}(\prod_{k=1}^{i} f_k)^{1/3}} \quad (30)$$

$$B(m)_3 = \frac{\sum_{i=1}^{m}\frac{\beta_i}{3(\prod_{k=1}^{i} f_k)}}{1+\sum_{i=1}^{m}(\prod_{k=1}^{i} f_k)^{1/3}} \quad (31)$$

Now we can have got the solution for order 1 to 3.

3. Closed-form expression for processing time

As we show in computation model in figure 2, the processing time for each child processor is the sum of its computing time and communication time and for processor P_0, it equals to the processing time. From equation (6),(7), combined with the condition that the start-up delay, Θ_{cm} and Θ_{cp} are all very small, these small values will be ignored ,we can get the expression of T_i:

$$T_i = \begin{cases} A_0(\alpha_0 N)^{\gamma}, i = 0 \\ G_i(\alpha_i N) + A_i(\alpha_i N)^{\gamma}, i = 1, \ldots, m \end{cases} \quad (32)$$

Our target is to get the optimal processing time T in the single level tree computing model. In this case, all the processing time of the child processors is equal to the computing of processor P_0, and then we can the solution of Tmin as follows:

For order 1 and order 2, the solutions have been got clearly. So now we focus on the order 3, substitute equation (29) to (32):

$$T_{\min} = A_0(\alpha_0 N)^3 = A_0\left(A(m)_3 + \frac{B(m)_3}{A(m)_3}\right)^3 \quad (33)$$

4. Speed-up for the order 3 case

Now let us calculate the speed up of our formulation to get a main image to the effort of case order 3. This is a very important parameter to value the result of our work and also it is a good mark for practice application. In the discussion, we talk about the homogeneous case which means that all the processors own same conditions:

$$Speed\_up = \frac{T_{normal}}{T_{\min}} = \frac{A_0 N}{A_0\left(A(m)_3 + \frac{B(m)_3}{A(m)_3}\right)^3} \approx (1+m)^3 \quad (34)$$

From the result of the speed-up, we can see that this computation model has a very good effort to calculate the distributed load.

Ⅳ. Conclusion

In our paper, we describe a new computation model to the distributed load computation, and get a good closed-form formulation of the model. From the speed-up result, the effort of the model is really good as the developing of the DLT theory. Also this work has a good future as we can continuously find the result of high order and get a generous result for all the case.

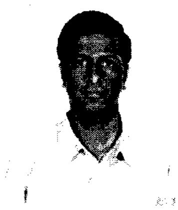
[References]

[1] Bharadwaj, V., Ghose, D., and Mani, V., "Optimal sequencing and arrangement in distributed single-level networks with communication delays", *IEEE Transactions on Parallel and Distributed Systems*, 5 (1994).

[2] Bharadwaj, V., Ghose, D., Mani, V., and Robertazzi, "T. G. Scheduling Divisible Loads in Parallel and Distributed Systems". *Los Almitos, CA: IEEE*

Computer Society Press, 1996.
[3] Stankovic, J. A., Spuri, M., Ramamritham, K., and Buttazzo,G. C. "Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms.", Boston: *Kluwer Academic Publishers*, 1998.
[4] Suresh,S.,Mani,V. and Omkar,S.N. "The effect of start-up delays in scheduling divisiable loads on bu networks: an alternate approach," *Compute Appl.* 40(10), pp. 1545-1557
[5] H.J.Kim and V.Mani, "Divisiable load scheduling in single-level tree network in nonblocking mode of xommunication", *Computers Math. Applic.* 46(10/11), pp. 1611-1623.

Run Cui
2008 B.S in Computer and technology, Harbin Institute of technology. 2008- for Master and Doctor degree, Multimedia and Information Security Lab, Korea University. Research Interesting: Distributed Load Computation
Email: cuirun@korea.ac.kr



Suresh Sundaram
1999, B.E in Electrical and Electronics Engineering from Bharathiyar University, Tamilnadu, India. 2001, M.E (Control and Dynamics) in Aerospace Engineering, Indian Institute of Science, Bangalore, India. 2005, Ph.D in Engineering, Indian Institute of Science, Bangalore, India. Research Interesting: Distributed Load Computation, Machine Learning
E-mail: Sundaram.Suresh@sophia.inria.fr

Hyoung Joong Kim
1978, B.S., in Electrical Engineering Seoul National University. 1986, M.S., Control and Instrumentation Engineering Seoul National University. 1989, Ph.D, Control and Instrumentation Engineering Seoul National University. Research Interesting: Multimedia Computing, Multimedia Security, Semantic Analysis, and e-Learning applications.
E-mail: khj-@ korea.ac.kr