INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

THE EVALUATION OF SCHEDULING ALGORITHMS BASED ON LOAD BALANCING IN DISTRIBUTED SYSTEM

Maryam Masoudi Khorsand[1], SHahram Jamali[2], Morteza Analoui[3]

[1] *Department of Computer Engineering, Sciences and Research Branch, Islamic Azad University, Ardabil, Iran, m.masoudi@iauardabil.ac.ir*

[2] *Department of Computer Engineering, University of Mohaghegh Ardabili, Ardabil, Iran, jamali@iust.ac.ir*

[3]*Department of Computer Engineering, University of Science and Technology Tehran, Iran, analoui@iust.ac.ir*

Author Correspondence: Iran, Ardabil, Sciences and Research Branch, Islamic Azad University, 00989141508575//00984516617525, m.masoudi@iauardabil.ac.ir

Abstract

A distributed system is a set of computers that contains many different executive resources. One of the most important advantages of distributed system is the high speed of executive programs because one program is able to use several computers simultaneously foe executing. The fair division of load among computers is the goal and advantage of distributed system. Load division means that management system which is given for each program will determine the best resource for executing that program and will give the program to that computer in order to execute. Therefore, with effective use of scheduling actions and proper distribution of applied programs among processing resources, the whole loading will be divided among all computers. The current investigation will study three algorithms in distributed systems which will help to create loading balance among existing processing and will compare their advantage and disadvantages.

Keywords: Distributed System; Load Balancing; Reliability; Scheduling, Makespan.

1. Introduction

Before invention of personal computers, there was no distribution system in practice. In that period, using a computer was only sitting at a terminal and making connection with a large system. Although the terminals were located in several buildings and physical spaces, in practice there was a central computer which had the responsibility of doing all processing and saving all data. These problems led to the necessity of creation of new patterns to design computer programs and as a result, new generation of computer programs was created in the

world of software. These programs need distributed systems. A distributed system is a program that its processing potential might be provided by several physical computers and its data will be located in several physical places. Also a distributed system is a set of computers that contains many different executive resources. One of the most important advantages of distributed system is the high speed of executive programs because one program is able to use several computers simultaneously foe executing.[1]

The fair division of load among computers is the goal and advantage of distributed system. Load division means that management system which is given for each program will determine the best resource for executing that program and will give the program to that computer in order to execute. Therefore, with effective use of scheduling actions and proper distribution of applied programs among processing resources, the whole loading will be divided among all computers. So in balanced calculation and in distributed systems, task scheduling for executing different computers is one of the challenging issues that need algorithm with high efficiency. At the present study, a number of these algorithms through load balancing have been introduced and evaluated and their efficiency has been compared.

2. THE INVESTIGATION AND COMPARISON OF ALGORITHMS OF LOAD BALANCE SCHEDULING

2.1 ACO Algorithm

Colony's ant algorithm for scheduling task in distributed systems will need some information such as the number of processors, the speed of each processor and characteristics of each action.[2] Colony's ant algorithm which is an exploratory algorithm is driven from ants' cumulative life. Ants collaborate for finding the shortest way between food resource and their anthill. While the ants are moving, each ant leas a substance called pheromone. Using this pheromone, the shortest way can be found.

We will analyze this algorithm with a new way and in its improved state. The way used for load Balancing can be divided into four steps:

2.1.1 Task Evaluation

It is supposed that the processors are the same from point view of their capacity and processing ability. And only one unit of loading is given to each task. Each ant is only able to transmit one task. It is necessary to estimate the load of processor in each step of execution algorithm in order to make it clear which processor has an extra load or which tasks and how many of them should immigrate to the neighboring node to reduce gradually lack of Balancing. The load evaluation is an essential point and must be handled carefully.

Tasks are carried as capsules by the ants in each of the processors; there is line in which the ants wait for executing. The structure of the line is shown in figure 1.

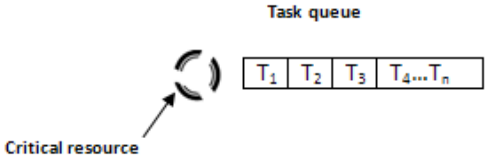

Figure 1. Line structure related to each node

In order to ease the task, we can consider graph which the number of its nuts is corresponded with its processors. Each ant waiting in the line knows about its primary load. The primary load of each ant is supposed to be one unit. As soon as

they immigrate, the information related to each ant's load will be updated and the rest of the ants will be informed. The new load is calculated by formula 1:

$$a_l \leftarrow \frac{1}{2} \times (a_l + \frac{1}{m-1} \sum_{j=1}^{m-1} oth_{ant_{l_j}}) \qquad (1)$$

m is the number of ants in the current node and oth-ant-1j is the sum of the load of ants standing in front of ant j.

2.1.2 Error Computation

After computation of loading rate, each ant will estimate local errors based on this formula:

$$est_{error} = \frac{1}{2}\left[\frac{a_l - \frac{1}{n}\sum_{i=1}^{n} neig_{hn_{l_i}}}{a - l} + \frac{a_l - \frac{1}{m-1}\sum_{j=1}^{m-1} oth_{ant_{l_j}}}{a - l}\right] \qquad (2)$$

n is the number of neighboring nuts and neigh-n-li is the load of node neighboring i. m is the number of ants in the current node and oth-ant-lj is the mean load of ant j in the current node. If an ant located in a good position compared to Balancing state, the error will be lower and in the state of final Balancing, the error will be zero.

2.1.3 Ants' Decision Mechanism for Leaving or Remaining in the Current Node

Each ant keeps its own current position in its small memory. This is corresponded with ant priority to enter and to get executed to a processor.[3]

The parameter tf is a resistant factor which its value is less than 1 and is calculated by formula (3):

$$tf = \exp(est\_error) - 1 \qquad (3)$$

If the position of ant is less than a-1×(1-tf), the probability of remaining in that node will be more than leaving the node:

$$P - leave(t) = \begin{cases} 0, & pre\_load - K(t) < 0 \\ 1, & prec\_load - K(t) > cr\_n\_l_i \\ \dfrac{prec\_load - K(t)}{cr\_n\_l_i}, & else \end{cases} \qquad (4)$$

2.1.4 The Suggested Mechanism for Immigration of Ants and the Selection of Next Node

Each ant possesses a small memory to save the needed information during traveling from one node to another one. One of this information is pheromone matrix t= {tij} holding the information which have been saved by previous ants in order to help new ants find optimal solution. In the colony algorithm of ant, there is another variable named field of view which is discovery information. A field of view which is suggested suitable for this problem is identified in this way and this value for all neighbor nodes is calculated through following formula (5):

$$\eta_{ij} = \frac{1}{L_j + 1} \qquad (5)$$

In this formula, $L_j$is the length of processor line belonged to j (neighbor of node i) plus constant value.

Each given artificial ant uses the information saved in pheromone matrix ($t=\{t_{ij}\}$) and field of view (η_{ij}) to find the better solutions during its traveling. Each ant chooses one of its neighbor nodes considering the probability function introduced in formula (6):

$$P_{ij} = \frac{t_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{k \epsilon N_i} t_{ik}^{\alpha} \times + \eta_{ik}^{\beta}} \qquad (6)$$

Ni is the collection of adjacent neighbors of node i. The parameters α and β are the constant values which have high effect on above-mentioned formula. If α has higher value, the effect of pheromone will be more, whereas if β was more than α, the effect of field of view will be increased. The simulation results indicate that the best amount for these parameters was $\alpha = 0.01$ and $\beta = 5$ in the algorithm suggested by researchers. Therefore, each ant computes the probabilities corresponded by each route and chooses the root with highest probability.

2.1.5 How to Update Pheromone

The difference in our suggested algorithm using the common method of ant colony is that changed Meta heuristic pheromone ACO was used in order to increase the effect of immigrations. Trails have dependent properties and each node owns unique pheromone table. Also, local normalization process has been included, so that trail<1,Σtrail= 1

At first, a primary amount of pheromone is placed on each route which was obtained using formula (7) and recorded in the appropriate pheromone table.

$$t_{ij}(0) = \frac{1}{d_{ij}} \qquad (7)$$

dij is the number of nodes belonged to the neighbor of node i. for example, for each node which has three neighbors , the primary amount of pheromone on each route equals to a third.

Assuming this point that the number of given node is I, each pheromone table will have d+1 lines which in the d lines belongs to neighboring nodes and 1 belongs to the current node. The amount given to trails (pheromones) is updated when an ant leaves one node or enters the line of another node. In addition, two other specific values must be selected to make change in trails. These two values are called$\Delta tr1$ and $\Delta tr2$. $\Delta tr1$ controls the changes made when one task enters and $\Delta tr2$ monitors the changes made when one task leaves.

Figure 2 displays node i which is surrounded by 3 other nodes and one task is coming in node i from one of processors.

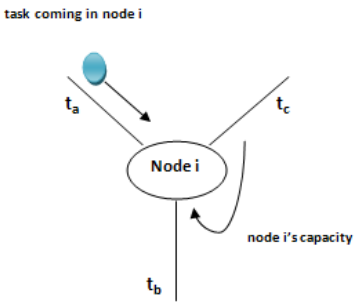

Figure 2. task coming in node i

The formula (8) describes how to update pheromone table for this node:

$$tr_a(t+1) = \frac{tr_a(t)}{1+\Delta tr_1} \qquad (8)$$

$$tr_x(t+1) = \frac{tr_x(t) + \frac{\Delta tr_1}{L+1}}{1+\Delta tr_1}$$

Trx and tra are trails corresponded by tails x and a. L and x ∈{b, c} is the length of the line belonged to the selected node by ant carrying the task. Now if one task is leaving node I, another formula set is used for updating pheromone table (figure 3).

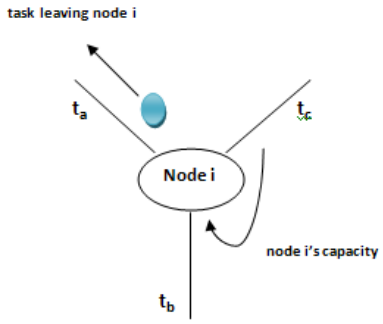

Figure 3. task leaving node i

$$tr_a(t+1) = \frac{tr_a(t)+\frac{\Delta tr_2}{L+1}}{1+\Delta tr_2} \quad (9)$$

$$tr_x(t+1) = \frac{tr_x(t)}{1+\Delta tr_2}$$

The philosophy of pheromone updating in routs follows as: if the task is coming current node from special direction, the probability of sending other task in that direction must be minimized in order to avoid task abundance in computation resource. Also, to increase the effect of task tracking through low load nodes, $\Delta tr1$ must be very bigger than$\Delta tr2$. the authors of this research combined the probable mechanisms of immigration and task leaving as well as placing pheromone using load Balancing method and suggested an optimal exploratory solution for computational systems with topology and favored graph.

2.1.6 ACO Algorithm General Administration

Message-passing method was used to administrate this algorithm. The processors were as considered graph nodes and the ants which transmit the tasks were regarded as message. [4]

2.2 Direction Processor Algorithm

With respect to a variety of topology which is applied to tie processors to each other, each processor will have a number of neighbors. [5]

For instance, in a system which in processors have been tied using mesh topology, each processor has a maximum of 4 neighbors. It is possible to define a characteristic in each processor which expresses one of processors neighboring the given processor. To make it clear, assume that this characteristic is called direction and there are a maximum of 4 neighbor processor for each processor, therefore the element direction can have 4 values in each processor like{left, right, up, down} or {1,2,3,4}.(Figure 4)

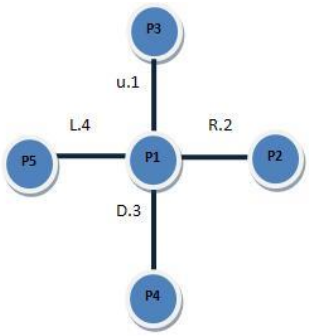

Figure 4. The characteristic amount of direction from P1 point of view

On the other hand, the idea of this algorithm was generated considering this fact that one processor can be neither free nor heavily loaded, but it is possible to have both free processor and heavily loaded processor in surrounding area. So this processor can provide opportunity for free processor and heavily loaded processor to get familiar. This processor actually can announce to its neighbor processors that I am not free but I have a neighbor that is free.

The algorithm acts in a way that once a processor was free, it would send a message to its neighbors including number of free processor, number of message of one counter, and one field to prove message accuracy.

The number of free processor is the processor itself. Each processor can experience free state. This message is not a demanded message from previous free state of processor. We use a message number. The counter is a characteristic that is added one unit whenever a message moves from one processor to another specifies and the distance of one message from primary sender processor (free).

The neighbor processor saves this message and its characteristic key considering the direction from which the message has been sent. If the processor was in the light load state, it would investigate the messages sent by its neighbors and choose the one with highest priority (a message with a distance less than 9) and send to its neighbor. Then the processor with heavy load selects a free processor and heavy load is dividend considering the process capacity in the processor. If the system is multifunctional, the load will be evaluated based on number of functions and it would be measured based on the amount of data in the case of multi data system.

About the time of algorithm performance, each processor in every state (free, heavy load, light load) can do its function toward load Balancing algorithm separately. But load immigration, that is load transmission between free processor and heavy load processor, should be carried out simultaneously.

In this procedure, there was an attempt to introduce new load Balancing algorithm. This algorithm possesses novel capabilities, for example, it doesn't need a separate tracking algorithm to track between task sender and receiver processors. This algorithm can perform in a variety of operational environment.

2.3 Most Fit Task (MFTF) algorithm

The MFTF algorithm [6] mainly attempts to discover the fitness between tasks and resources for user. It assigns resources to tasks according to a fitness value, and the value is calculated as follows:

$$\text{Fitness}(i,j) = \frac{10000}{1+|W_i/S_i - E_i|} \qquad (10)$$

where Wi is the workload of the ith task, Sj is the CPU speed of the jth node, and Ei is the expected time of the ith task. Wi/Sj is the expected execution time using this node. |Wi/Sj − Ei| is the difference of the estimated execution time and the expected task execution time. Ei is determined by the user or estimated by the machine. How to set Ei is calculated by (11).

$$E_i = A + n \times S \qquad (11)$$

where A is the average response time of the 100 latest done tasks; n is a non-negative real number and S is the standard deviation of task response time for the 100 latest done task.
When performance time is measured near Ei, it means that this node is more suitable for task to be assigned.

Since some processors may have much load and others remain free, this algorithm improves the efficiency of distributed system through load balancing and process capability of the system to smooth the periods with load traffic in the nodes.

This is done by transmitting some loads from the nodes with heavy load to other processors in order to process. Although many problems related to scheduling are solved with this algorithm, it can occur in the real environments with wrong scheduling because this algorithm doesn't consider the efficiency of resources.

3. THE COMPARISON OF MENTIONED SCHEDULING ALGORITHM

In Table 1 Scheduling algorithms listed are compared for load balancing and makespan. The benefit and the disadvantages of for each is checked.

Table1. Comparison of Scheduling Algorithms

Parameters	Advantages	Disadvantages
ACO algorithm	It causes load balance.	The amount of pheromone must be updated every moment, so there will be an increase in the time of process.
Direction processor algorithm	It does not need tracker. It is compatible. It has high resistance in dynamic environments.	It has no reliability.
MFTF algorithm	It is compatible. It has high resistance in dynamic environments.	It doesn't consider the efficiency of resources.

4. Conclusion and Future Studies

In this paper, 3 scheduling algorithm based on load Balancing in the distributed systems were investigated. although there have been a lot of studies concerning task scheduling in distributed and parallel systems, a new challenge still can be interesting and many research projects can be done. The present study was an attempt to focus on current scheduling algorithms and the mentioned algorithms were compared considering a variety of aspect and advantages and disadvantages of each one were explained. In the future, these algorithms can be tested on heterogeneous processors.

5. REFERENCES

[1] Foster, I., Geisler, J., Nickless, W., Smith, W., and Tuecke, S. 1997. Software infrastructure for the i-way high performance distributed computing experiment. In Proc. 5th IEEE Symposium on High Performance Distributed Computing, 562–571.

[2] Chang, R. SH., Chang J. SH., and Lin, P.SH. 2009. An ant algorithm for balanced job scheduling in grids, Future Generation Computer Systems, 20–27.

[3] Craus, M., Bulancea, C. 2004. An Agent Based Model for Real-time Load Balancing in non-Uniform Connected Computing Environments, Third European Conference on Intelligents Systems and Technologies ECIT2004, 21-23.

[4] Nagdeforusfha, M., and torogi hagigat, A. 2007. The presentation of one new algorithm for providing process load Balancing in the distributed systems using ants' colony. 8th conference of intelligent systems in Mashad Ferdosi University, 3-5.

[5] Rostami, M., and Sadegzadeh, M. 2012. Theinvestigation of load balancing in parallel computation and presentation of one new load balancing algorithm. National conference of computer and IT in Azad University of derud branch, 6-7.

[6] Wang, SH.D., Hsu, I.T., Huang, Z. Y. 2005. Dynamic scheduling methods for computational grid environments, International Conference on Parallel and Distributed Systems, 22–28.