2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

Real Time Tasks Scheduling in Distributed Systems

Najmeh Danesh[1,a], Hossein Shirgahi [2,b+]

[1] Sari Branch, Islamic Azad University, Sari, Iran
[2]Jouybar Branch, Islamic Azad University, Jouybar, Iran

Abstract. Multi processor systems and distributed systems are used a lot in the real time applications these days. Cost reduction of these systems is one of the popularity reasons of them. We can access two-processor systems with low price these days. The distributed workstations such as networked personal computers are prevalent too. Some other reasons of the real time systems' attraction in multi processors and distributed systems are quicker reaction time and fault tolerance. In this paper we describe some of the most proper tasks real time scheduling algorithms in multi processor and distributed systems and we express some advantages and disadvantages of them.

Keywords: Real time scheduling, multi processor systems, distributed systems.

1. Introduction

Tasks scheduling in the multi processor systems and the distributed systems is more difficult than doing it in the single processor systems. As we know, the time complexity of optimal scheduling algorithms for a set of independent real time tasks is polynomial. (The integer numbers of the tasks should be scheduled). However determining an optimal algorithm for a set of real time tasks in a multi processor system or a distributed system is a NP-hard problem [1, 2]. Multi processor systems are known as tightly coupled systems. It means there is a common physical memory in the system. Distributed systems are known as loosely coupled and there is no common physical memory. In a tightly coupled system, the inner process communication (IPC) is cheap and it can be ignored rather than tasks' run times. It is possible because IPC is achieved by reading and writing in the shared memory. However it is not true in a distributed calculation systems which inner task communication times are comparable with task run times. Therefore, a multi processor system might use a centralized distributed scheduling, whereas distributed systems can't use it. A centralized scheduler needs maintenance the status of different tasks of a system in a centralized data structure.

In needs different processors in the system. When the task status is changed, it makes it upgrade. So it leads to a high over communication loading. Tasks real time scheduling in distributed multi processor systems has a problem as bellow:

Tasks assignment to the processors and tasks scheduling on the separated processors: tasks assignment is related with dividing a set of tasks and their posture assignment to the processors. Task assignment can be static or dynamic. In a static assignment algorithm, tasks assignment to the nodes is contiguous and it isn't changed during the time. Whereas, in a dynamic task scheduling, the tasks are assigned to the nodes when the task s are accrued. Therefore, different samples of a task might be assigned to the different nodes. After a successful assignment of the tasks to the processor, we can consider tasks to each processor separately.

+ [a]Najmeh.Danesh@Gmail.com, [b]Hossein.Shirgahi@gmail.com

Task assignment to the processors in distributed environments is a NP-hard problem and the complexity of determining an optimal answer is exponential. Therefore, most used algorithms are hierarchy. Tasks assignment algorithms can be categorized in static or dynamic algorithms. In static algorithms, all tasks are divided to sub systems. Each sub system is devolved to a separated processor but ready tasks for running are put in a priority common queue and when the processor is available, the tasks are distributed in the processors for running. Therefore, various samples are possible in the periodic running of the tasks on different processors. The most difficult real time systems are made based on data and the most of them will be static in the near future. However, a dynamic real time system can use available processors efficiently.

2. Tasks Assignment to Multi Processors

We consider some algorithms for static assignment of real time tasks to the processor of the multi processor systems in this section. We know an efficient static assignment algorithm. The assignment is done before the time of running and the assignment will be valid during a complete running of the system. As we told before, task assignment algorithms do not try to minimize the communication costs as an inner connection time. Because despite of multiprocessors communication time is equal to the memory access time. As this reason, assignment algorithms which we analyze here, do not work desirable in the distributed environments. These algorithms are centralized. In continuance we analyze some important task assignment algorithms in the multi processor systems.

2.1. Beneficiary Balance Algorithm

This algorithm shows the tasks ascending based on their beneficiary. it removes tasks from the front of the queue one by one and it assigns the tasks to the processor which is used less than the others. Its goal is beneficiary balance between different processors. Beneficiary U_i of each processor is equal to the total beneficiary of all processors $\overline{u}$ of the system in a completely balanced system. So, beneficiary of a processor p_i is the sum of all assigned tasks beneficiary to this processor. If ST_i is the all assigned tasks to the processor P_i, then the beneficiary of P_i is $U_i = \sum_{J \in ST_i} U_{Tj}$. U_{Tj} is the beneficiary of performing task T_j by Pi processor. However, the use of this algorithm is too difficult for achieving a complete balance of beneficiary in different processors for an arbitrary task set. In another words, it is difficult for P_i processor to do U_i=U. the optimal results are achieved from this algorithm. Every beneficiary balance algorithm's aim is to minimize the $\sum_{i=1}^{n} \left| (\overline{u} - u_i) \right|$, n is the number of processors in the system, $\overline{u}$ is the average of processors' beneficiary and U_i is the beneficiary of P_i processor. If the number of the processors is fixed in a multi processor system, this algorithm works well. Also when the tasks are scheduled in the separated processors by EOF, this algorithm can be used. [5]

2.2. Next Fit Algorithm for RMA

In this algorithm the tasks are divided as each class is scheduled on a single processor by RMA scheduling. This algorithm tries to use minimum number of processors and against the beneficiary balance it does not need to know the number of system processors before running the algorithm. It categorizes different tasks in different classes based on tasks beneficiary [6].

Each class involves the same beneficiary which is scheduled on the same processor. It categorizes the tasks based on their beneficiary following this rule: the tasks are categorized to m classes, the task T_i belongs to class j, if 0<=j<=m,

$$\left(2^{\frac{1}{j+1}} - 1\right) \prec e_i / P_i \leq \left(2^{\frac{1}{j}} - 1\right) \quad (1)$$

Suppose that we want to categorize a system tasks beneficiary to 4 classes. Therefore, different classes are categorized based on the tasks' beneficiary by Eq. 1 as follow:

Class (1): $(2^{\frac{2}{1}} - 1) \prec C_1 \leq (2^{\frac{1}{1}} - 1)$

Class (2): $(2^{\frac{3}{1}} - 1) \prec C_2 \leq (2^{\frac{1}{2}} - 1)$

Class (3): $(2^{\frac{4}{1}} - 1) \prec C_3 \leq (2^{\frac{1}{3}} - 1)$

Class (4): $(2^{\frac{5}{1}} - 1) \prec C_4 \leq (2^{\frac{1}{4}} - 1)$

So, the beneficiary network for the different Classes is as follow:

Class (1): (0.41, 1]

Class (2): (0.26, 0.14]

Class (3): (0.14, 0.26]

Class (4): (0, 0.14]

Equation (1) is such as defined networks on the tasks beneficiary map. Each task is assigned to a network based on its beneficiary. It is not difficult to understand that when the beneficiary values are greater, the networks sizes will be bigger. Class (1) sizes are 0.41 to 1, whereas the Class (4) sizes are 0 to 0.14. The simulations show that the next fit algorithm's usage is needed for 2.34 times more than the optimal numbers of processors at most. Now we describe this task assignment method's application with an example: table (1) shows the running times (micro second) and the periods (micro second) of a set with 10 real time tasks.

Suppose that the tasks need to be runned on a multi processor system with 4 processors. Assign the tasks to the processor by next fit algorithm. Suppose that the separated processors are scheduled by RMA algorithm. Table 1 shows the task set of this example.

Table 1: The task set as an example.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
e_i	5	7	3	1	10	16	1	3	9	17
p_i	10	21	22	24	30	40	50	55	70	100

Now we can determine the beneficiary of different tasks and we assign different tasks to different classes based on their beneficiaries. Table 2 shows the calculation of each task beneficiary and it shows the task beneficiary to different classes.

Table 2. The solution of the example.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
e_i	5	7	3	1	10	16	1	3	9	17
p_i	10	21	22	24	30	40	50	55	70	100
u_i	0.5	0.33	0.14	0.04	0.33	0.04	0.02	0.05	0.13	0.17
Class	1	2	4	4	2	2	4	4	4	3

3. Implementations and Results

We implement the expressed method of this paper by Delphi 7 in windows XP. The implemented program is able to receive 100 processors and 1000 tasks independently. The tasks, the resources and their parameters are received randomly. We run beneficiary balance algorithm 5 times for 10 processors. Fig. 1 shows the difference between the beneficiaries of each processor with the average beneficiary. We run next fit algorithm for RMA 5 times for 20 processors and 4 classes. Fig. 2 shows the average beneficiary of each class.

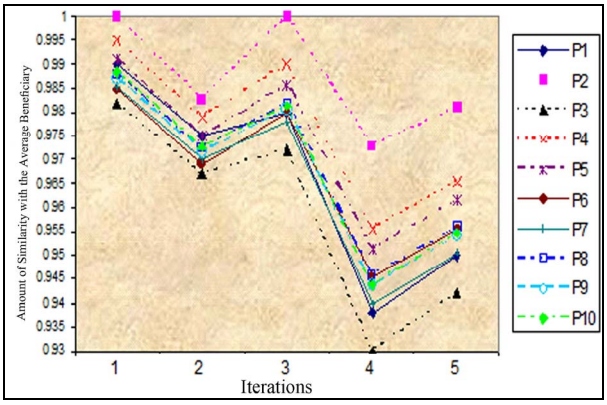

Fig. 1: The amount of similarity with the average beneficiary in beneficiary balance algorithm running.

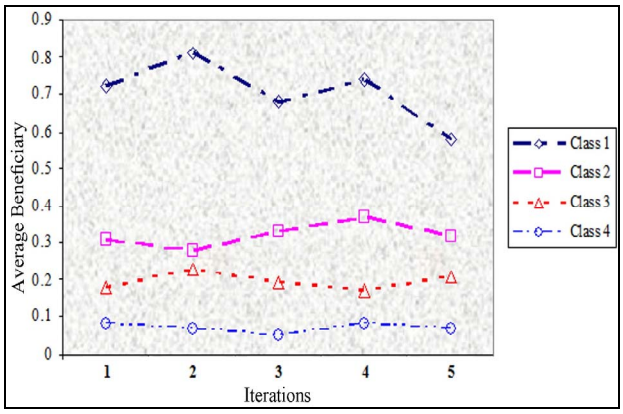

Fig. 2: The average beneficiary of classes in next fit algorithm running for RMA.

4. Conclusions

The beneficiary balance algorithm can compliance the beneficiary balance all processors in the system with 99 percent assurance factor. The next fit algorithm for RMA can compliance 0.73 of beneficiary for class 1, 0.33 of beneficiary for class2, 0.21 of beneficiary for class3 and 0.7 of beneficiary for class 4 in average. We propose the combination of some evolutionary expansion algorithms such as GA, PSO and GELS with the expressed method in real time scheduling of distributed systems.

5. References

[1] T. F. Abdelzaher and V. Sharma. A synthetic utilization bound for a periodic tasks with resource requirements. In: *Proc. of the 15th Euro micro Conference on Real-Time Systems (ECRTS 2003)*, Porto, Portugal, July 2003, p. 141–150.

[2] A. Amin, R. Ammar and A. E. Dessouly. Scheduling real time parallel structure on cluster computing with possible processor failures. In: *Proc. of the Ninth IEEE International Symposium on Computers and Communications (ISCC 2004)*, July 2004, p. 62–67.

[3] Y.Amir, B. Awerbuch, A. Barak, R. S. Borgstrom and A. Keren. *An opportunity cost approach for job assignment in a scalable computing cluster*. IEEE Transactions on Parallel and Distributed Systems, 11(7), 2000, p. 760-768.

[4] R. A. Ammar and A. Alhamdan. Scheduling real time parallel structure on cluster computing. In: Proc. of the Seventh IEEE International Symposium on Computers and Communications (ISCC 2002), Taormina, Italy, July 2002, p. 69–74.

[5] ATLAS (AToroidal LHC Apparatus) Experiment, *CERN (European Laboratory for Particle Physics)*. Atlas web page. http://atlas.ch/.

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