Survey of different Task Scheduling Algorithm

[1]Viral Patel, [2]Milin Patel
[1]Student, [2]Assistant Professor
[1]Master in Computer Engineering, Parul Institute of Engineering & Technology,
Vadodara, Gujarat, India
[2]Computer Engineering, SardarVallabhbhai Patel Institute Of Technology,
Vasad, Gujarat, India
[1]viralpatel15591@gmail.com, [2]milin2784@hotmail.com

***Abstract*—In distributed real-time systems, both computation and communication are critical factors. The timeliness of activity relies on both communication and computation. This property requires that both computation and communication be integrated into the system and interplay between them. Models that are employed in current distributed real-time systems are constructed solely on computation. They are not considering the communication cost. To extend the previous research we propose a new model, which integrates both computation and communication. The interplay between computation and communication is characterized in the model. Here in this paper explain different method of task scheduling. Paper describes task scheduling method that only considers the computation cost.**

***IndexTerms*—Task Scheduling, Real time system, Rate Monotonic, Earliest Deadline first, Directed acyclic graph**

1. INTRODUCTION

In distributed real-time systems, communication costs are no longer negligible. Whether activities can be completed in time depends on whether both the computation and the communication involved in them can be completed in a timely way. The timeliness of computation relies on that of communication, and vice versa. Hence, communication is as important a factor as computation in terms of meeting timing constraint.

There are lot of work was done for scheduling the task in the distributed system. The scheduling is done according to minimizing the execution time of computation or minimizing the communication cost or both. There are many algorithm for scheduling the task in the distributed system like RMS [5], EDF [2] etc. But these entire algorithms are schedule according to computation cost. But the timeliness of computation relies on that of communication and vice versa [1]. So in the new model we believe that communication should be integrated with computation and interplay between computation and communication should be fully captured. So we proposed the model that will fully capture communication and computation cost and interplay between them. In this paper, describe many algorithms for scheduling the task.

2. DIFFERENT TASK SCHEDULING TECHNIQUE

There are many techniques for scheduling the task in the system. Various techniques are proposed in literature to deal with the task scheduling. In following section categorize the different techniques available in existing literature.

2.1 Rate Monotonic Scheduling (RMS)

Liu and Layland published the paper [5] which provides the theoretical foundation to all modern scheduling algorithms for real-time systems. In their analysis, the authors limit themselves to independent, periodic, preemptable, hard real-time tasks executing on a single processor. The task which has highest priority is always executing first by the processor. Although author only address scheduling of real-time tasks for a single Processor, their results turned out to be applicable to distributed systems as well. The first scheduling model introduced by Liu and Layland is the RMS. RMS is a static and fixed-priority based scheduler in which the priority of a task is given by using rate monotonic so the task which has higher the rate, the higher the priority. In RMS the priority is decide using period of task, the task which has smaller period get highest priority. Once the priorities are assigned, it is up to the processor to execute the available task with the highest priority.

Example: suppose two tasks are schedule using the RM scheduling, here for RM scheduling the two tasks has period and completion time.

Example: T1=5 C1=1
T2=7 C2=3

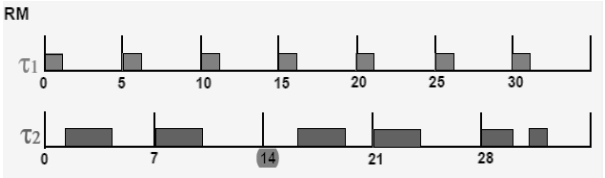

Fig 2.1 Rate Monotonic scheduling

2.2 Earliest Completion Time First (ECF)

In this algorithm the tasks are scheduled according to its completion time. The task which has smaller completion time gives highest priority. The ECF [5] is same as RMS algorithm .This algorithm is fixed priority algorithm so task must be schedule according to decided priority smaller completion time or smaller computation cost has highest priority. So according to this algorithm if we schedule the task it is possible to miss deadline of some task which has smaller priority. ECT scheduling is also a fix priority based scheduling but it schedules the task according to its completion time. Suppose we have three tasks for scheduling. First task has completion time 5, second has 10 and third has 3. Here in our example task third has highest priority. Second has priority less than third and greater than first task. So according to priority all three tasks are schedule. This algorithm is preemptive so when highest priority task comes lower one resumes.

2.3 Dependent Activity Scheduling Algorithm (DASA)

DASA (Dependent Activity Scheduling Algorithm) [4] is another example of a utility accrual resource scheduler used in supervisory control systems. DASA is implemented by The second generation of the Alpha OS. DASA has the twofold objectives of maximizing system- wide accrued utility while minimizing the number of missed deadlines. Relative dependencies can be developed by Concurrent tasks as they serially access devices, channels, and other exclusively shared system resources: a task requesting a resource is considered dependent on the task currently holding that resource. Given its objectives, DASA makes appropriate Scheduling decisions according to dynamic dependencies and the corresponding precedence relationships. The algorithm assumes that possible deadlock resulting from cycles in the dependency graphs can be detected and resolved. A simplified version of the algorithm, DASA/ND [4], can be used in systems where tasks are known to be independent. DASA considers all deadlines as hard, and exclusively relies upon rectangular utility functions.

1. Create an empty schedule.
2. Determine dependencies among tasks.
3. Calculate potential utility density for each task.
4. Resolve deadlocks if detected.
5. Sort tasks by potential utility densities.
6. Examine each task in decreasing order of potential utility density:
   6.1. Tentatively add the task and its dependent tasks to the schedule in deadline order.
   6.2. Test schedule feasibility.
   6.3. If not feasible, remove task from schedule.
   6.4. Optimize schedule if possible.
7. Execute task in schedule with earliest deadline.

Steps of DASA Algorithm

2.4 EDF Scheduling (Earliest deadline first)

The EDF scheduling [2] algorithm is a priority driven algorithm in which higher priority is assigned to the request that has earlier deadline, and a higher priority request always preempts a lower priority one. So EDF is preemptive scheduling algorithm. This scheduling algorithm is an example of priority driven algorithms with dynamic priority assignment in the sense that the priority of a request is assigned as the request arrives. EDF is also called the deadline-monotonic scheduling algorithm. For the EDF algorithm, we assume everything that same as RM [5] scheduling except that the task is not periodic. EDF is an optimal uniprocessor scheduling algorithm. Earliest deadline first scheduling (EDF) schedules the task according to its deadline. EDF is dynamic priority based scheduling algorithm. In the EDF the task that has smaller deadline has highest priority. Let us understand it with one example.

Example: C1= 2 **T1=D1=5**
C2= 4 **T2=D2=7**

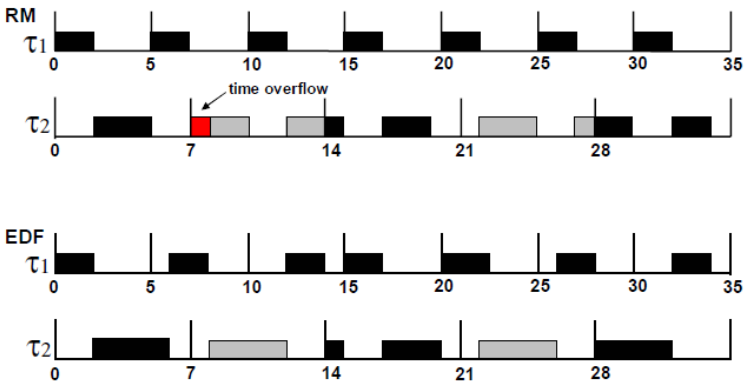

Fig 2.2 scheduling using EDF and RM

Here in this figure when we schedule the task using RM scheduling the deadline is missed and time overflow occur. And when we schedule the task using EDF scheduling the deadline is meets.

2.5 Fault Tolerant Earliest Deadline First Scheduling Algorithm

In the EDF scheduling algorithm [2] when fault is occur the scheduling of the faulty task stopped and other tasks are scheduled. The general approach to fault tolerance in uniprocessor systems is to maintain enough time redundancy in the schedule so that any task instance can be re-executed in presence of faults during the execution. In this algorithm a scheme is presented to add enough and efficient time redundancy to the Earliest-Deadline-First (EDF) scheduling policy for periodic real-time tasks. This scheme can be used to tolerate transient faults during the execution of tasks. We describe a recovery scheme which can be used to re-execute tasks in the event of transient faults and discuss conditions that must be met by any such recovery scheme.

The general approach to fault-tolerance in uniprocessor systems is to make sure that if a fault occurs during its execution there is enough slack in the schedule to allow for re-executing of any task instance. Tasks are executed according to the EDF scheme if no faults occur. However, when a fault occurs in a task, a recovery scheme is used to re-execute that task. In the EDF policy with utilization bound less than 100%, there is a natural amount of slack in uniprocessor. But this natural slack does not enough for re-executing faulty tasks. To have one efficient fault-tolerant mechanism in the schedule it is necessary that additional slack time is added to the schedule. The insertion of slack and recovery mechanism is called the FT scheme. The recovery mechanism ensures that the reserved slack can be used for task re-executing before its deadline, without causing other tasks to miss deadlines. When fault is detected at the end of some task, the system goes into recovery mode. In this mode, task will re-execute at its own priority.

For example consider the task C1=2, T1=d1=5, C2=2, T2=d2=8.In figure 1.a no fault has occurred while the tasks are executed. In figure (b) a fault has occurred when T12 (T12 denotes the second period of T1) was being executed and has been detected at time 7, then recovery scheme has been called, i.e., new copy of T1 has been added to the schedule (figure (c)), and has been completed using the slack time (time redundancy).

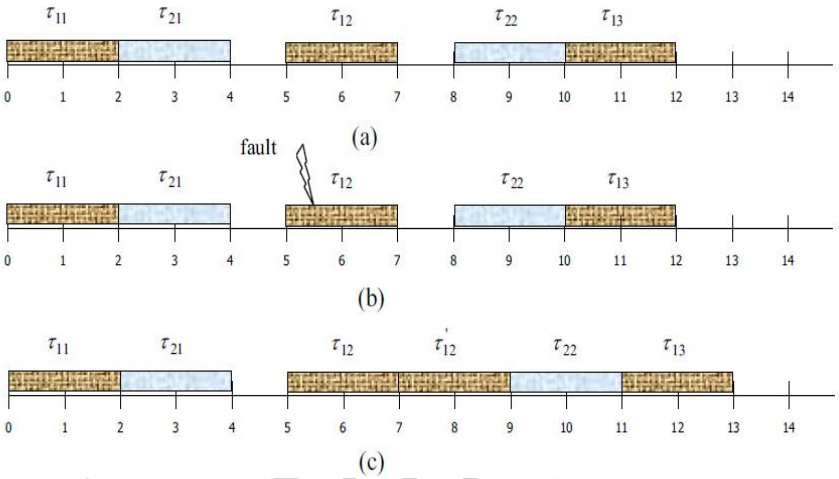

Fig 2.3.Scheduling for FEDF (a) tasks are scheduled in not present faults, (b) T_{12} has encountered with a fault , (c) the recover scheme has been called and T_{12} has been re-executed using time-redundancy

2.6 CAUSA (Communication Aware Utility Accrual Scheduling Algorithm)

In this paper we are going to see the scheduling the task according to the CAUSA algorithm that consider both computation and communication cost into the system. The algorithm work like following manner.

Algorithm

Here for scheduling let us take one example. We consider one DAG (directed acyclic graph) for dependent task and schedule these tasks according to the proposed algorithm. Let us see with example.

Input: Directed Acyclic Graph

Output: scheduling result;

Step 1: Prepare a predecessor element set for all node.

Step 2: Select the set with maximum unprocessed predecessor node.

Step 3: Allocate processor to the initial node and repeat the allocation until all node in the selected element set gets processor and mark node as processed node.

Step 4: Repeat step 2 and step3 until all element set gets their processor.

Example

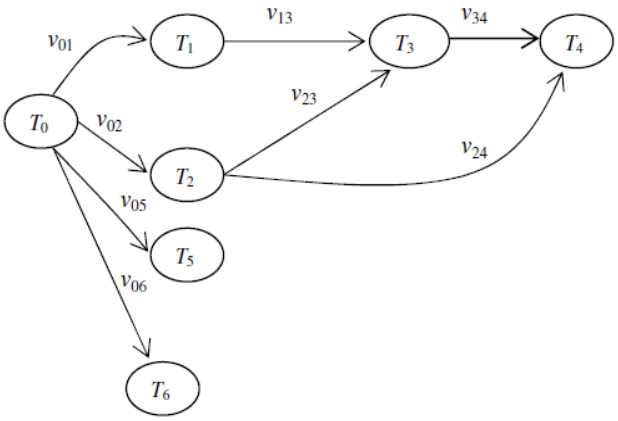

Fig 2.4 Task graph

	1	2	3	4	5	6
0	**(3,2)**	**(3,2)**			**(3,3)**	**(3,3)**
1			**(4,2)**			
2			**(3,2)**	**(3,2)**		
3				**(3,2)**		
4				**(4,0)**		
5					**(5,0)**	
6						**(6,0)**

Table 2.1 Parameter for the Task graph

Here, first value is computation time and second is communication between two task i.e. (3,2) means task T0 has computation time 3 and communication cost between T0 and T1 is 2.

Analysis

Step 1: Prepare a predecessor element set for all node.
Step 2: Select the set with maximum unprocessed predecessor node.

- In our example, we first see that task T4 has its predecessor element set {E01, E02, E13, E23, E24, E34, E44}. So T4 is selected first.

Step 3: Allocate processor to the initial node and repeat the allocation until all node in the selected element set gets processor and mark node as processed node.

- So, task T0 is picked because it has no predecessor. T0 is can be dispatched to processor P0 and completed at time 3.
- After T0 is processed T1 and T2 can be picked.
- Suppose T1 is picked and it is dispatched to processor P0 for avoiding communication cost between T0 and T1. So T1 is released at time 7.
- Next, T2 can be picked. This can dispatched to either P1 or P2 because its completion time on P1 or P2 will be less than that on P0 due to T1.T2 will take time 3(computation) + 2(communication)=5 for released.
- After T2, T3 can be picked and dispatched to P1 for avoiding communication cost between T2 and T3. But communication cost between T1 and T3 is 2 so T3 is start at time 9.
- Next T4 is dispatched by P1 to avoid communication cost between T3 and T4.

Step 4: Repeat step 2 and step3 until all element set gets their processor.

- At second round, task T6 is selected and dispatched to processor P2. And T6 is start at time 6 because communication cost between T1 and T6 is 3.

- And at the last round T5 is selected and dispatched to processor P0 because its completion time on P0 is less than on P1 and P2.

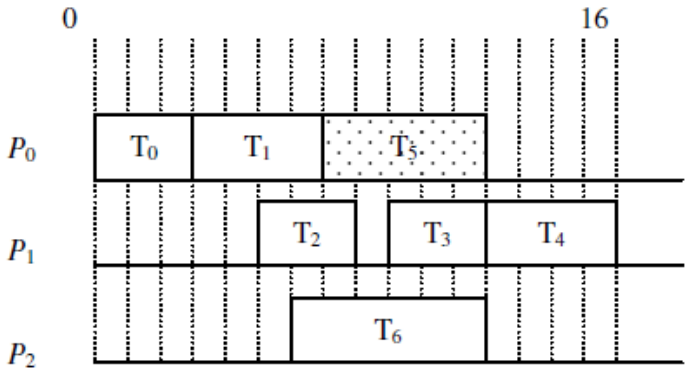

Fig 2.5 Scheduling result

3. CONCLUSION

In this paper, we see different task scheduling algorithm. And we conclude that RMS is a static priority based algorithm and EDF is dynamic priority based algorithm. But both RMS and EDF are not considering communication cost for task scheduling. They only consider computation cost of a task for scheduling. But CAUSA algorithm considers both computation and communication cost for scheduling. And according to communication and computation cost CAUSA schedule the tasks.

ACKNOWLEDGMENT

I would like to express the deepest appreciation to Mr. JwalantBaria who has continuously guided me through my research work.

REFERENCES

Xinfa Hu, Joseph Y.-T. Leung "Integrating Communication Cost into the Utility Accrual Model for the Resource Allocation in Distributed Real-Time Systems" The 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, IEEE 2008.

HakemBeitollahi, SeyedGhassemMiremadi Geert Deconinck "Fault-TolerantEarliest-Deadline-First Scheduling Algorithm" IEEE Transaction on Computers, Sep. 2007.

Technical Report No. 2005-499 Scheduling Algorithms for Real-Time Systems ArezouMohammadi and Selim G. Akl School of Computing Queen's University Kingston, Ontario Canada K7L 3N6 arezou July 15, 2005

Peng Li, BinoyRavindran "Fast, Best-Effort Real-Time Scheduling Algorithms" IEEE TRANSACTIONS ON COMPUTERS, VOL. 53, NO. 9, SEPTEMBER 2004

Utility Accrual Scheduling with Real time javaSahroozFaizabadi, WiliamBeebee,BinoyRavindran and Peng.Li and Martin Rinard.

Scheduling Real-Time Tasks in Distributed Systems: A Survey Alex Gantman Pei-NingGuo James Lewis FakhruddinRashidUniversity of California, San Diego Department of Computer Science

http://www.cse.unsw.edu.au/cs9242/08/lectures/09real-timex2.pdf

Liu C. and Layland J.W. Scheduling Algorithms for multiprogramming for hard real time environment. Journal of ACM Vol. 20(1) 46:61,1973