International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 08, August-2015

Performance Based Scheduling Real-Time Application over Multicore Reservation

Ishwar Chaudhary, Radhakrishna Naik, Pratap Suryawanshi
MIT, Aurangabad

Abstract- **Now days, computational control existing in multicore platform involves the software to be quantified in terms of parallel performance flows. Task set considered as application with precedence relation among them and it is communicated by directed acyclic graph. Till now in directed acyclic graph tasks are partitioned into number of flows for reduce bandwidth necessities and scheduled according to precedence relationship only. This paper suggests multicore performance contribution factor scheduling algorithm (MPCF). On the other hand tasks are scheduled considering precedence relationship, performance contribution factor (PCF) as well as deadline of each task in the task set. Proposed scheduling algorithm is proved by scheduling real-time periodic task set on quad core processor by using Earliest Deadline First (EDF) scheduler. Simulating the proposed MPCF scheduling algorithm through the analysis of three case studies and gives optimal improvement solution for requirement of execution time and bandwidth are less for executing an application on core. Results are compared with Branch and Bound algorithm.**

Keywords: Performance Contribution Factor (PCF), Earliest Deadline First, Multicore reservation, Real- Time Scheduling, Branch and Bound Algorithm, IRIS.

I. INTRODUCTION

In real-time systems use of the available resources while achieving a desired performance is a serious design goal. Day by day continuously increase of complication and requiring the higher performance industry moving towards the multicore platforms.

Multicore architecture gives effective solution to the problem of increasing the processing speed with controlled power dissipation. If only operating frequency of single processor increase then it would cause the serious heating problems and the problem of power consumption. In multicore platform tasks are executing parallel. Therefore, the tasks are allotted to processor considerably affects the number of active cores required for running the application. In multicore system researchers are still working to produce the new theoretical results. Control power available in multicore system requires the programming structure; in which application optimize the allocation of tasks on different cores [1].

In real-time systems results are required within a specified deadline. Now days real time system are more difficult and more correctness of consistent results are predictable. In real-time systems tasks are interact with each other to achieve the output of the system. The collection of tasks is called as tasks set. In the tasks set tasks are depend on each other to achieve the goal of system, effective resource consumption, and create the precedence relationship among the tasks. The task in the task set considering the precedence relationship in which successor tasks should need the predecessor tasks result to achieving the output of the system.

The processing platform affects the scheduling strategies. Consequently it can be well-defined the scheduling algorithm by Liu and Layland [2] namely Rate Monotonic (RM), Earliest Deadline First (EDF) are optimum scheduling algorithm on uniprocessor processing platform but it is not the case processing platform moved to multiprocessor. Now day’s huge amount of changes in processing platform. In the early stage of the processing platform progress which was characterized by uniprocessor, then shifted towards the multiprocessor platform, distributed architecture, and now multicore platform. Modern tendency turns around multicore processing platform, in which multiple cores mounted on single chip [3].

In real-time systems, scheduling strategy must consume the strength of the processing platform and this paper claims involvement of each task in task set throughout the schedulability of whole task set. Lower significant task must have less priority that the higher significant task. Performance contribution factor (PCF) of each task indicates that quantity of involvement of tasks to achieve output of system. Achievement level must be considered if the task performs partly. Involvement of task can be calculated whether it completes execution partly or successfully in MSS.

This paper suggests preemptive scheduling algorithm intended for multicore architecture. The algorithm proposed scheduling strategy will schedule task set by considering the precedence relation among the task, Performance contribution factor (PCF) as well as deadline of each task in task set. Task can completes full of its instruction or insufficient of its instruction is considered as contribution factor of each task in task during scheduling of task. The accumulative performance of each task can be calculated by calculating the probability of each state and achievement level of each state is assumed. In the task set task priorities are allocated on the basis of two factors such as Performance Contribution factor (PCF) and Deadline of each task in task set.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

Task set considered as an application and is expressed by Directed Acyclic Graph (DAG). Till now, in directed acyclic graph tasks are scheduled according to the precedence relation among the tasks only. Due to that higher priority tasks in directed acyclic graph are wait in the queue because of the resources acquire by the lower priority task. So, requirement of the bandwidth for performing application is more and the performance of the system reduces. Therefore, the algorithm proposed an innovative scheduling strategy for scheduling task in directed acyclic graph for decreasing the bandwidth requirement and increasing the system performance.

Organization of the Paper:

The rest of the paper is organized as follows. Section II represents the related work. Section III represents the system model with terminology and recalls the some background concepts. Section IV represents the proposed work for scheduling the application according to the precedence relation along with considering the PCF and Deadline. Section V illustrates the performance evaluation of the proposed algorithm. Section VI represents the performance analysis of the proposed work. Section VII states our conclusion and possible extensions for a future work.

II. RELATED WORK

In real-time systems deadline is the prime constraint. Therefore, the tasks are executed in the proper order with respect to the deadline constraints. Many researchers and academicians has been done lot of work right from first result publish in 1973, by Liu and Layland[2] in the field of real-time scheduling. In the first paper of real-time scheduling Liu and Layland expressed optimum fixed priority algorithm i.e. Rate monotonic (RM) and Earliest Deadline First (EDF). In RM algorithm priorities are assigned to task in the task set on the basis of their period whereas in EDF priorities are assigned to tasks in task set on the basis of their deadline.

Partitioning a real-time periodic task in task set by using branch-and-bound algorithm proposed by Peng and Shin [4]. Proposed method is optimum but requirements of bandwidth for executing a task graph are not considered. To assign and schedule the real-time periodic task in task set proposed by Ramamritham [5] by considering precedence relation and communication among the tasks.

Baruah and Fisher [6] proposed, heuristically partition a set of deadline constrained sporadic tasks set on multiprocessor but they are not consider the precedence relation among the task. Chetto et al [7]. addressed on the problem of precedence relation among the real-time tasks, who proposed a method for setting the activation time and deadline to each task in the task graph to convert that graph into precedence graph.

In precedence constraint problem of assigning tasks is generally a type of NP hard problem. Issues involve in scheduling of precedence constraint tasks in real-time system addressed in Performance contribution and Deadline algorithm [8]. In this proposed method of scheduling a task in task graph by considering the Performance contribution factor and deadline of each task in task set. Increase reward with increase service type of imprecise computation model was proposed to permit for the trade-off the quality of computations in favor of meeting the deadline constraints [9]. In this model tasks are logically divided into two parts one is mandatory part another one is optional part of the task. In this paper for scheduling of task we are using the mandatory and optional part of the task strategy for increasing the system performance and decreasing the bandwidth consumption of task.

2.1 *Real Time Scheduling on Multicore Processing Platform:*

Multicore processing platform also called as single-chip multiprocessor. In the multicore processor there is only a single chip and multiple processing units are mounted on that chip [10]. The multicore platform has two categories, first one is homogeneous multicore platform and second one is heterogeneous multicore platform. In global scheduling algorithm, consist of global queue and it is associated with multiple processing unit. In partitioning scheduling algorithm there is separate queue associated with separate processing unit. Migration of task not allowed in partitioning scheduling [10]. Once the task allocated to any core preemption method is used in scheduling of task set then tasks mandatory as well as optional part executed on that core only.

There are many challenges for scheduling a task on multicore to succeed goal of system along with reducing bandwidth consumption, use the less number of cores due to that power consumption is less, increase the processor speed. Therefore, now day's different parameter is used in designing of scheduling strategy in multicore platform.

2.2 *Performance Contribution Factor of Task:*

In real-time systems each task in task set contribute to the achieving the output of the system. But in MSS it may not happen some tasks may not execute completest but contributes up to definite amount. Therefore, in the scheduling of task set contribution of each task must be considered [8]. In some task set task are scheduled according to precedence relation only. Due to that resources are not available for scheduling of higher priority tasks, all the resources are acquire by the lower priority tasks. Therefore, contribution of each task in task set with respect to the other task must be considered. Tasks are divided into mandatory and optional part. Mandatory part is the compulsory part of the task in execution and optional part of not affects too much on system if it is not executed at the right time. In precedence relation graph tasks are connected to each other so without the execution of predecessor task; successor task not starts its execution. Priorities assign to the classification of task in task set on the basis of two parameters Performance Contribution

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

factor (PCF) and Deadline. Then the task is scheduled by considering the precedence relation with PCF and Deadline of each task.

PCF of each task can be calculated as,

$$E_G = \sum_{k=0}^{k} P_k \cdot G_k \qquad (1)$$

Where,
E_G = Expected MSS Performance.
P_k = Probability of task in state k.
G_k = Performance of task in state k.

The probability that task extents to particular state can be calculated by,

$$P_k = L_{in} / L_t \qquad (2)$$

L_{in} = Total number of inputs received tasks from the other task.
L_t = Total number of inputs associated with that task.

In the precedence relation graph successor task depend on the predecessor task. So the probability of task execution having only one preceded task can be calculated by the conditional probability. If the task having more than one preceded task then probability of task execution can be calculated by using Bayes rule.

III SYSTEM MODEL:

In real-time system task set is considered as an application and is expressed by Directed Acyclic Graph (DAG) [1]. In Directed Acyclic Graph tasks are scheduled according to the precedence relation only. Due to that higher priority tasks are blocked by lower priority tasks. To prevent the blocking of higher priority tasks calculate Performance Contribution Factor (PCF) and Deadline of each task in task set.

3.1 Related Terminologies:

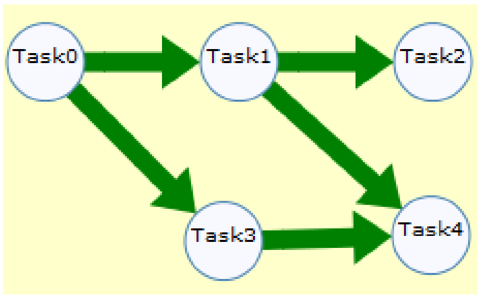

Figure 1: Directed Acyclic Graph

a) *Application A*: It is set of tasks expressed by Directed acyclic graph (DAG).
b) *Task set*: It is collection of task in one group and set the precedence relation among the task set.
c) *Precedence Relation R*: It is defined as partial ordering of task in task set. $R \subseteq A \times A$. In the application Task 0 is the predecessor of Task 1 and Task 1 not start its execution before the completion of Task 0
d) *Path P*: It is subset of task set order according to the precedence relation R. $P \subseteq A$.
e) *Predecessor Task*: It is tasks whose output needed in the execution of the other tasks in the precedence graph. In the above fig.1 Task 0 is the predecessor task of Task T1 and Task 3.
f) *Successor Task*: It is task which is not start its execution before the completion of predecessor task in the precedence graph. In the above fig.1 Task 1 is the successor task. Task 1 not start its execution before the completion of Task 0
g) *Sequential Execution Time C^s*: It is the minimum time needed to complete the total application in sequentially. Tasks are executed sequentially one after the other. In the above figure 1 Task 0 to Task 4 are executed sequentially.
h) *Parallel Execution Time C^p*: It is the time needed to complete the application on parallel architecture with four numbers of cores.

3.2 Partitioning Application into Flows:

This section describes the optimal method for partitioning application into flows. Fig.1 considered as one application and partitioning that application into number of possible ways. Partitioning application into flows find out the number of required core and how much bandwidth is required for executing the application. If the numbers of fragmentation are less then only the required numbers of cores are also less. There are the two possible methods are used for the optimal partitioning application into flows [1].

1) Branch and Bound Algorithm:

This algorithm is used for partitioning application into number of possible flows [1]. The main aim of this algorithm is to minimize the bandwidth requirement for the execution of application. This algorithm gives best solution for partitioning application. This algorithm concentrates only on the partitioning not on scheduling application. It schedules the application according to the precedence relation among the tasks in the application.

The aim of this algorithm is to minimize the bandwidth consumption and is calculated by,

$$B = \sum_{k=1}^{m} B_k = \sum_{k=1}^{m} \left(\alpha_k + 2\sigma \frac{1-\alpha_k}{\Delta_k}\right) \qquad (3)$$

Where,
B = Total bandwidth of the application
B_k = Bandwidth of each flow.
m = Number of flows.
σ = Context Switch Overhead.

Branch and bound algorithm generates search tree for finding out the optimal partition. At the starting point of the search tree i.e. parental node of the tree Task T_1 associated with the flow F_1. Then branch and bound search the level 2 nodes, at that level Task T_2 associated with the already created flow F_1 or its create new flow F_2. Branch and Bound algorithm search the tree until the last element of the tree. In the Branch and Bound Algorithm if the element with flow has bandwidth greater than one then pruning condition is apply on that flow. Application starts execution according to the precedence relation.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

2) Heuristic Partitioning:

To compact with large number of tasks in task set heuristic partitioning is used for partitioning real-time application into number of flows [1]. Branch and bound algorithm is not possible if the task set contains the more than 15 - 20 tasks.

Heuristic algorithm starts with making the longest possible path into the flow F_1. At the start critical path of the precedence graph taking into flow F_1. The remaining task of the graph algorithm tries to fir them into the already existing flows by using the best fit strategy. If this not possible to fit the task in that flow then algorithm makes new flow. Algorithm schedules the tasks according to the flows and precedence relation of the precedence graph.

3.3 Demand Bound Function:

Demand bound function is used to estimate the amount of required computational resources for execution of the application. The application demand is calculated for core reservation for executing the application on core [11].

The processor demand of a task T_i that has activation time a_i, computation time C_i, period T, and relative deadline d_i, in any interval $[t_1, t_2]$ is defined to be the amount of processing time $g(t_1,t_2)$ requested by those instance of T_i activated in$[t_1,t_2]$ that must be completed in $[t_1,t_2]$

$$g_i = \left(\left\lfloor\frac{t_2-d_i}{T}\right\rfloor - \left\lceil\frac{t_1-a_i}{T}\right\rceil + 1\right)_0 C_i \quad (4)$$

IV PROPOSED WORK

4.1 Architecture:

In this section describes the how the proposed algorithm works .This is the architecture of the proposed Multicore PCF Scheduling algorithm as shown in fig.2.

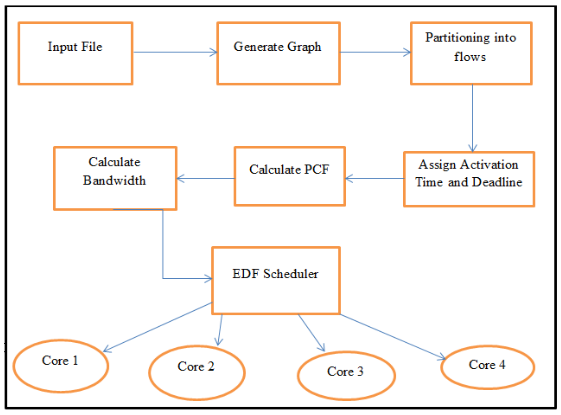

Figure 2: Architecture of MPCF Scheduling Algorithm

In the previous algorithm, concentrate on only partitioning an application into number of flows but not on scheduling. Tasks are schedule according to the precedence relation only. So, higher priority tasks are blocked by lower priority tasks. Due to that performance of the system reduces. Tasks take too much time for execution. Therefore, in the proposed algorithm calculating the performance contribution factor of each task in task set and schedule the task by setting the priority in task set on the basis of PCF and Deadline of each task. Then task are schedules on different core by Earliest Deadline First (EDF) scheduler.

Proposed Multicore PCF Scheduling algorithm also gives some novel idea for scheduling task in precedence graph. In the precedence graph successor task not start its execution before the completion of predecessor task. In precedence graph successor task need output of the predecessor task for execution. Every task has mandatory and optional part. Mandatory part is the compulsory part for execution of task. Proposed algorithm makes assumptions that if successor task have higher PCF value than the predecessor task then that time only execute mandatory part of the predecessor task. After the execution of mandatory part of predecessor task start execution of higher PCF task. Because of that strategy higher priority task are starts their execution as early as in the precedence graph.

4.2 Set Activation Time and Deadline:

Fig.1 considered as one application. Before the application start execution we know the activation time and deadline to each task in task set [12]. After the assignment of deadline and activation time then overall computation requirement of the application can be find out.

First algorithm start with assigning the activation time to the application. Assume that the application start time $t = 0$. So in the application all the task starts at time $t = 0$.

At the second step algorithm assigns deadline to each task of the application. First assume the total application deadline. Then the last task of the application means those tasks that are not the predecessor for any task in the application whose deadline equal to the total application deadline. For Example, in above figure Task 2 and Task 4 are last task of the application. So their deadline equal to the total application deadline.

Then assign deadline to remaining task in the graph for which all the successor have been considered. Therefore, deadline assign to such task is,

$$d_i = \min(d_j - c_j) \quad (5)$$

Here, T_i is the predecessor of task T_j. So deadline of predecessor task T_i is calculated by subtraction of computation time of successor task T_j through deadline of successor task T_j.

4.3 Calculation of PCF:

Let A be the application considered as task set expressed by Directed Acyclic Graph (DAG).

T be the set of n tasks i.e. $T = \{ T_1, T_2, T_3 \ldots\ldots \}$
C is the set of classes to classify the task from the task set. i.e. $C = \{ C_1, C_2 \}$
I) Calculation of PCF of each task in task set by using the some following criteria.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

In figure Task 0 is the root node of the DAG, so PCF value of that task can be calculated as,

$$E_G = \sum_{k=0}^{k} P_k \,.\, G_k$$

$$P_k = \frac{\textit{Total no of Outgoing Links}}{\textit{tal no of outgoing nodes associated with that ROOT Node}}$$

G_k = Assume.

For the last node of the DAG i.e. those tasks are not the predecessor for the any other successor task. In figure Task 2 and Task 4 PCF value can be calculated as,

$$E_G = \sum_{k=0}^{k} P_k \,.\, G_k$$

$$P_k = \frac{\textit{Total no of Incoming Links}}{\textit{Total no of nodes associated with that link}}$$

G_k = Assume.

For the other nodes of DAG i.e. in figure Task 1 and Task 3 PCF value can be calculated as,

$$E_G = \sum_{k=0}^{k} P_k \,.\, G_k$$

$$P_k = \frac{\textit{Total no of Incoming Links}}{\textit{Total no outgoing links}}$$

G_k = Assume.

II) Classification of tasks done on the basis of two parameters PCF and Deadline of each task.

Table1 Classification based on PCF and Deadline.

Priority Levels	PCF (EG)	Deadline (D)
Class I	High	Low
Class II	Low	High

4.4 Scheduling Strategy:

Task assignment to core is dynamically depending upon the precedence relation among the task and availability of core. Once task is allocated to particular core, task is not migrated to any other core. All the tasks are executed in preemptive manner which means higher priority task are scheduled first, they may not need to wait in queue.

Assumptions:

1. Task set can be represented by Directed Acyclic Graph concerning the precedence relation.
2. Tasks are divided into mandatory and optional portion.
3. Data required for transmission as to successor are tasked by mandatory portion.
4. Tasks are scheduled according to the EDF scheduler.
5. For scheduling dependent task, if the predecessors are from class I, then its mandatory as well as optional portion is executed.
6. If the predecessors PCF value is low than the successor task then only execute mandatory part of the predecessors' tasks.
7. Class II contains all the higher deadline tasks of the application.
8. All cores are homogeneous *i.e.* identical
9. Intercommunication among tasks is done by message passing.

Multicore PCF Scheduling Algorithm:

1. Take the input of tasks containing computation time and precedence relation among tasks.
2. Generate directed acyclic graph (DAG) consider as application.
3. Partition the application into number of flows by using branch and bound algorithm or heuristic partition.
4. Set the deadline and activation time to each tasks of the application.
5. Calculate PCF of each task of the application.
6. Calculate demand bound function of application for reservation of core.
7. Calculate total bandwidth required for the application to schedule on core.
8. Tasks are schedule on the core by EDF scheduler, considering the precedence relation among the tasks, PCF and deadline of each task.

V PERFORMANCE EVALUATION:

In real-time systems gives the correct output of system in specified time known as deadline. In real-time scheduling task must execute on or before its deadline. In task set task assigned to core in specific time for execution of task has to be done on or before its deadline.

In order to evaluate the performance of proposed scheduling strategy, simulation is done on three case studies.

Case Study 1:

Fig.3shows the task set along with precedence relationship among the task and considered as application. Table 2 shows the details of the entire task in task set i.e. Task Id, Computation time, Activation time, Deadline and PCF, requirement of mandatory and optional part of the task for scheduling.

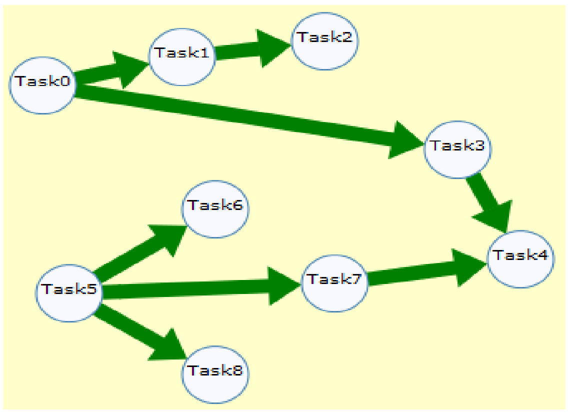

Figure 3: Directed Acyclic Graphs

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

Table2: Case Study 1 Task Table

Task	C.T	Activation Time (r_i)	Deadline (D)	PCF	M_i	O_i
Task 0	2	0	12	0.70	1	1
Task 1	3	0	15	0.90	3	0
Task 2	5	0	20	0.40	5	0
Task 3	3	0	16	0.80	3	0
Task 4	4	0	20	0.60	4	0
Task 5	3	0	11	0.60	2	1
Task 6	6	0	20	0.30	6	0
Task 7	5	0	16	0.70	5	0
Task 8	6	0	20	0.25	6	0

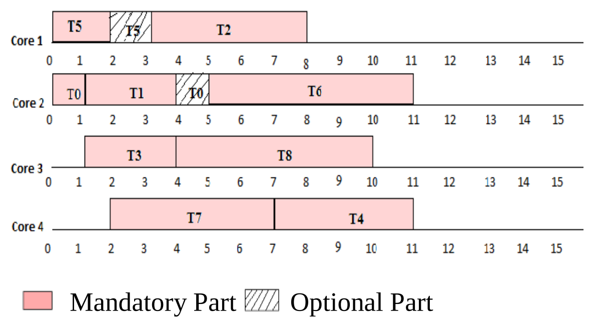

Figure 4: Scheduling Graph of Case Study I

Fig.4 shows the scheduling graph of the case study 1. This Proposed scheduling strategy managed by EDF Scheduler. Tasks are allocated to core by considering precedence relation, PCF of each task and deadline of each task. The scheduling strategy also considers the Increase reward with increase service type of imprecise computation model. Tasks are scheduled in mandatory and optional part. If the PCF value of the predecessor is low and deadline also low than the successor task then schedule only the mandatory part of the predecessor task. In the above scheduling graph task *T0* have PCF value lower than the task *T1*, so only schedule the mandatory part of the task *T0* then task *T1* start its execution. Because of this strategy higher priority task start their execution as early as possible. Application completes its execution one unit before as compared with branch and bound algorithm. Core reserves less time and due to that bandwidth required for execution of the application as compared to Branch and bound algorithm are less.

Case Study II:

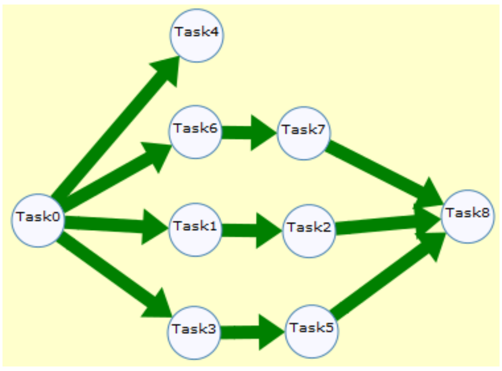

Figure5: Directed Acyclic Graph

In the case study II Fig. 5 shows the application consisting of 9 tasks expressed in Directed Acyclic Graph. Table 3 shows the details of all the tasks of the application.

Fig.6 shows the scheduling graph of case study II. In this case study proposed algorithm try to create Increase reward with Increase service type imprecise computation model scenario. Here, assumes that predecessor task completed its mandatory part then the next task in the precedence relation starts its execution. In this case study tasks are executing according to their precedence relation, PCF and Deadline of each task in task set

Table 3: Case Study 2 Task Table

Task	C.T	Activation Time (r_i)	Deadline (D)	PCF	M_i	O_i
Task 0	2	0	5	0.90	2	0
Task 1	3	0	10	0.80	3	0
Task 2	6	0	16	0.60	6	0
Task 3	3	0	13	0.60	3	0
Task 4	5	0	20	0.40	5	0
Task 5	3	0	16	0.60	3	0
Task 6	5	0	10	0.70	5	0
Task 7	6	0	16	0.50	6	0
Task 8	4	0	20	0.375	4	0

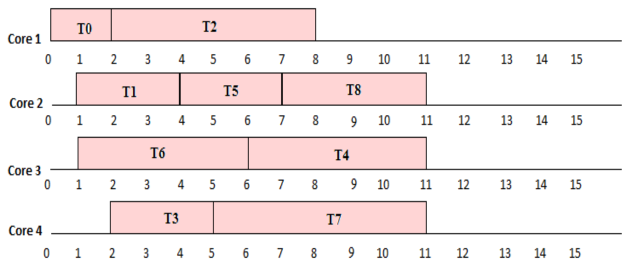

Figure 6: Scheduling Graph of Case Study II

Case Study III

Fig. 7 shows the task set consisting of 12 tasks and it is considered as an application. Table 4 shows the task table containing the details of all the tasks.

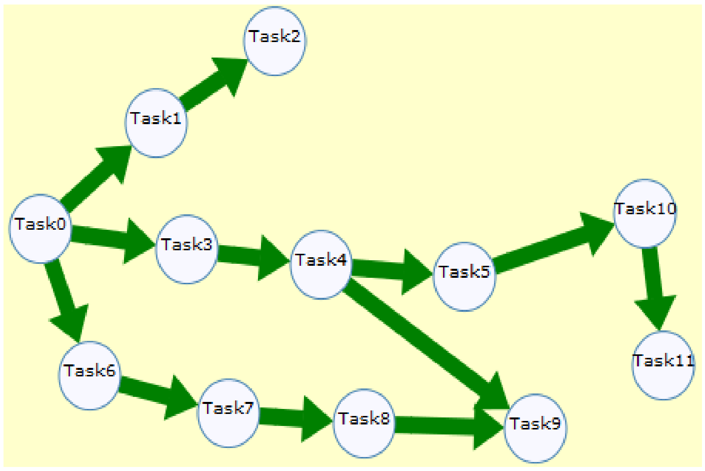

Figure 7: Directed Acyclic Graph

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

Table 4: Case Study III Task Table

Task	Computation Time	Activation Time (r_i)	Deadline (D)	PCF	M_i	O_i
Task 0	2	0	10	0.90	2	0
Task 1	3	0	25	0.60	3	0
Task 2	5	0	30	0.40	5	0
Task 3	4	0	14	0.80	4	0
Task 4	3	0	17	0.45	2	1
Task 5	6	0	23	0.80	3	3
Task 6	3	0	18	0.70	2	1
Task 7	4	0	22	0.80	2	2
Task 8	3	0	25	0.90	3	0
Task 9	5	0	30	0.39	5	0
Task 10	3	0	26	0.90	3	0
Task 11	4	0	30	0.35	4	0

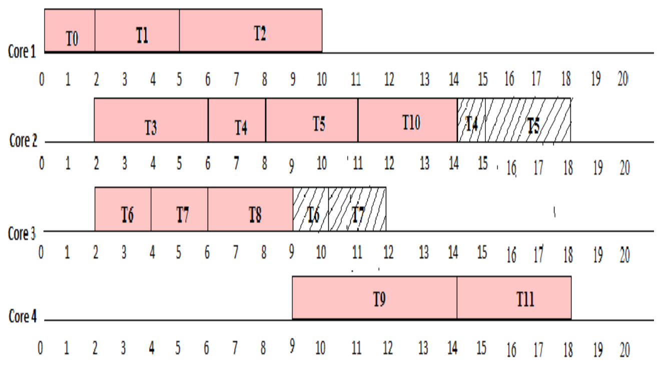

Figure 8: Scheduling Graph of Case Study III

Fig.8 shows the scheduling graph of Case Study III. In this scheduling graph proposed algorithm shows how to schedule the task if the middle of the task set having high PCF value than the other task.

VI PERFORMANCE ANALYSIS:

The proposed Multicore PCF scheduling algorithm compared with Branch and bound algorithm, in MPCF scheduling execution time require for executing an application on core less than the branch and bound algorithm. If the application completes its execution as early as possible then cores are free as early, due to that cores are not busy with one application, and it is then reserves for another application. In MPCF scheduling algorithm requirement of the bandwidth for executing an application on core are less as compared to the branch and bound algorithm.

Table 5: Comparative performance of the algorithm

Case Study	Algorithm	Execution Time	Bandwidth Requirement
Case Study I	Branch and Bound	12	2.842
	PCF Scheduling	11	2.524
Case Study II	Branch and Bound	17	2.435
	PCF Scheduling	11	1.89
Case Study III	Branch and Bound	22	3.23
	PCF Scheduling	18	2.75

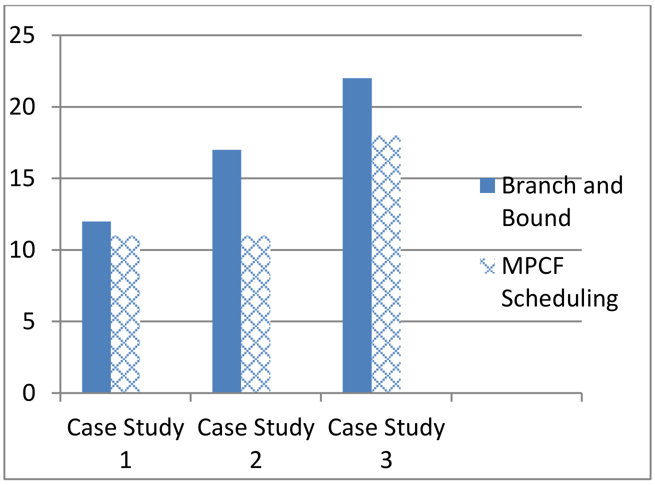

Figure 9: Execution Time Analysis

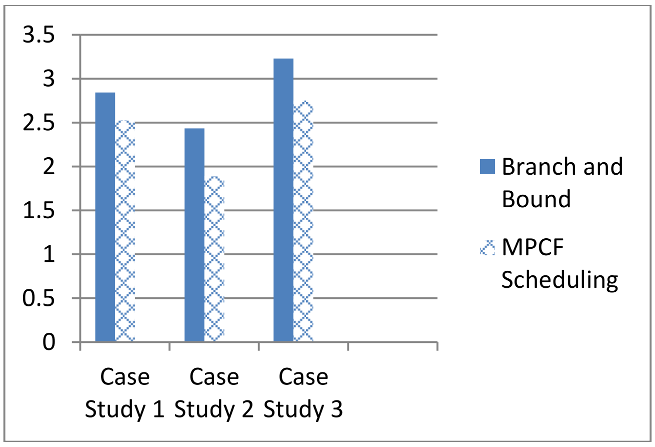

Figure 10: Bandwidth Requirement Analysis

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

VII CONCLUSION AND FUTURE SCOPE:

Problem of precedence constraint tasks can be solved more effectively with this approach.

In proposed Multicore PCF scheduling algorithm addresses problem of precedence constraint task scheduling and it consider precedence relation, performance contribution and deadline of each task taken into account of the scheduling of task on multi-core processing platform which has more than one core.

Simulating the three case studies with MPCF scheduling algorithm and then compared with branch and bound algorithm, it gives 20.6 % improvement in execution time and 16.14 % bandwidth are less utilize.

By using the MPCF scheduling algorithm require Execution time is less. Cores are idle as early as possible. Bandwidth requirements for executing an application on the core are low. Thus, the performance of the system increases.

REFERENCES

[1] Giorgio Buttazzo, Enrico Bini, “Partitioning Real Time Application Over Multicore Reservation,” IEEE Trans. Ind. Inf. 7 (2) (2011) 302–315.

[2] C. Liu, J. W. Layland, “Scheduling algorithm for multiprogramming in hard real-time environment,” Journal of ACM 20, 1973, pp. 46-61.

[3] Hitesh P. Daulani et al. “Precedence Constraint Task Scheduling for Multicore Multikernel Architecture,”Journal of IOSR, volume 16, Jul-Aug 2014, pp. 43-53.

[4] D.T.Peng and K. G. Shin, “Static allocation of periodic task with precedence constraint in distributed real-time systems,” in 9th international conference on Distributed Computing Systems, Newport Beach ,CA, USA, June 1989, pp. 190-198.

[5] K. Ramamritham, “Allocation and scheduling of precedence-related periodic tasks,” IEEE Transactions on Parallel and Distributed Systems, vol. 6, April 1995,pp. 412-420.

[6] S. Baruah and N. Fisher, “The partitioned-multiprocessor scheduling of deadline-constrained sporadic task system,” IEEE transaction on Computers, vol. 55, no 7, 2006,pp.918-923.

[7] H. Chetto, M.Silly and T. Bouchentouf “Dynamic scheduling of real-time tasks under precedence constraints,” Real-Time Systems, vol. 2, no.3, , Sep.1990, pp. 181- 194.

[8] Radhakrishna Naik, R. R. Manthalkar, “A new approach to schedule the precedence constraint tasks in real-time systems,” Journal of IJCSIT, vol.3, 2012, 3436-3443.

[9] Radhakrishna Naik, R.R Manthalkar, “Instantaneous utilization based scheduling algorithms for real-time systems, ” Journal of IJCSIT, vol.2, 2011, 654-662.

[10] Fan Ming, ”Real-Time scheduling of FIU Embedded Application on Multicore Platforms,” Electronics Thesis and Dissertation.Paper 1243. 2014.

[11] A.Rahni, E. Grolleu and M. Richard, “Feasibility analysis of non-concrete real-time transactions with edf assignment priority,” in proceedings of the 16th conference on Real-Time and Network Systems, Rennes, France, Oct. 2008, pp. 109- 117.

[12] Yifan Wu, Zhigang Gao, “ Deadline and Activation time assignment for partitioned real time application on multiprocessor reservation,” Journal of System Architecture, 60, 2014, 247-257.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)