ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

A Study on Hybrid Scheduling for Homogeneous Multicore Architecture

Vinuta A Patil

Assistant Professor, Dept. of CSE, SJBIT, Bangalore, Karnataka, India

ABSTRACT: This paper proposes a study on different scheduling algorithms used to schedule processes to different processors in a system and gives advantage of using hybrid scheduling algorithm over other algorithms. Multiprogramming computer systems execute multiple programs concurrently .In a multiprogramming environment the main objective is to share a resource efficiently in a system and to do this a hybrid scheduling algorithm is more effective compare to other algorithms. This paper is a study on hybrid scheduling for real time and non-real time applications running parallelly over a homogeneous multicore architecture.

KEYWORDS: multiprogramming, scheduling, multicore

I. INTRODUCTION

Scheduling is a process of assigning a different PCB (Process Control Block) of a processes to a processor. Scheduling is a fundamental function of an operating system. In this paper we are considering a hybrid scheduling for scheduling the different tasks of an application, that application can be a real time or non-real time.

Hybrid scheduling uses two level scheduling policy. In the top level a spordiac server is used which implements all the different scheduling algorithms for a different tasks of an application.At the bottom level rate monotonic OS scheduler is implemented.

In this paper we study a hybrid scheduling algorithm to be implemented on multicore processors. A multicore processor combines two or more independent cores in a single Integrated Circuit(IC) chip.

All these multicores may share a single cache or cores may have its own cache .In order to improve the performance a parallelism should be implemented and a concept of multithreading has to be implemented between the cores

The rest of the paper is structured as follows: Section II discusses a different scheduling algorithm. Section III discusses task model .Section IV discusses architecture of a hybrid scheduling. Section V discusses a load balancing algorithm. Finally in section VI some concluding remarks are made.

.

II. SCHEDULING ALGORITHMS

There are different scheduling algorithms [1, 2, 3] implemented to schedule the different tasks in an application. In this section we study those algorithms as well as compare them with each other.

A. First Come First Served Scheduling Algorithm (FCFS)

FCFS is the simplest scheduling algorithm. For this algorithm the ready queue is maintained as a FIFO queue. A PCB (Process Control Block) of a process submitted to the system is linked to the tail of the queue. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

B. Shortest Job First Scheduling Algorithm (SJF)

For this algorithm the ready queue is maintained in order of CPU burst length, with the shortest burst length at the

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

head of the queue. A PCB of a process submitted to the system is linked to the queue in accordance with its CPU burst length. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

C. Shortest Remaining Time First Scheduling Algorithm (SRTF)

For this algorithm the ready queue is maintained in order of CPU burst length, with the shortest burst length at the head of the queue. A PCB of a process submitted to the system has its CPU burst length compared with the remaining time of the PCB being executed. If the newprocess requires less time than that remaining of the 'active' process then preemption occurs and it becomes the new PCB's turn for execution, otherwise it is linked to the queue in accordance with its CPU burst length. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

D. Round Robin Scheduling Algorithm (RR)

For this algorithm the ready queue is maintained as a FIFO queue. A PCB of a process submitted to the system is linked to the tail of the queue. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. Processes being executed are pre-empted on expiry of a time quantum, which is a system-defined variable. A pre-empted process's PCB is linked to the tail of the ready queue. When a process has completed its task, i.e. before the expiry of the time quantum, it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

E. Priority Scheduling Algorithm

For this algorithm the ready queue is maintained in the order of system-defined priorities. A PCB of a process submitted to the system is linked to the last PCB in the queue having the same or a higher priority. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

F. Hybrid Scheduling Algorithm

This algorithm uses two level scheduling policy. In the top level a spordiac server algorithm is implemented for all the tasks of an application program. Each spordiac server will have its own scheduling algorithm. At the bottom level a rate monotonic OS scheduler is implemented to schedule top level spordiac servers

The authors of [4] have used several simulators for the performance evaluation of all the algorithms and proved that Hybrid scheduling works well from a system perspective as it gives minimum Average Waiting Time and Minimum Turnaround Time.Hybrid scheduling algorithm is well suited for multicore architecture.

III.TASK MODEL

Task model in [5] is summarized as follows:
Real time application A(r, d) is characterized by a release time r and a relative deadline d. Each real time task is characterized by T(r, c,e,l,d) where r is release time, c is computation time, e is earliest start time, l is latest start time and d is relative deadline. The release time and computation time is known at the release time of its application .The earliest start time, the latest start time andrelative deadlines are computed according to precedence relationship between tasks.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

IV. ARCHITECTURE OF A HYBRID SCHEDULING MODEL

The authors in [3] have explained the architecture of a hybrid scheduling model and it is as shown

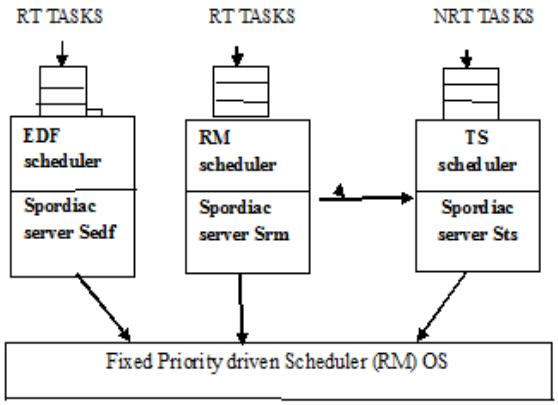

fig1: Scheduling architecture in processing core

The workload in the real-time system consists of real-time (hard and soft) and non-real-time tasks. In each processing core, there are several sporadic servers with a CPU budget *c* and a period *p* at the top level. Each sporadic server is associated with a ready queue contains ready tasks to run on the server. Each server has a scheduler associated with it. The server scheduler uses a scheduling algorithm, such as EDF or RM or time sharing algorithm, to schedule tasks and order tasks in the ready queue of sporadic server. All the real-time tasks with the earliest deadline first (EDF) algorithm are executed on the sporadic server S_{EDF}, all the real-time tasks with the rate monotonic (RM) algorithm are executed on the sporadic server S_{RM}, and so on. In addition, all the non-real-time tasks, which adopt the time sharing scheduling policy, are executed on the sporadic server S_{TS}.

At the bottom level, there is a fixed-priority driven scheduler (RM scheduler) called the operating system scheduler, which is adopted to maintain all the top level sporadic servers.

V. LOAD BALANCING ALGORITHM

Since we are using multicore architecture the load on processing cores should be balanced for a better performance. Hence we make use of load balancing algorithms. The authors of paper [3] use a load balancing algorithms which has two sub algorithms: Most-Demand-First Algorithm (MDFA) and Idlest-Fit Algorithm (IFA). The main idea of MDFA is that the task with the most CPU utilization demand is allocated first,and the tasks in an application are sorted in the decreasing order of their reserved CPU utilizations. The idea of IFA is that theprocessing cores are sorted in the decreasing order of their available CPU capacity, and the idlest processing core is always searched first when an allocation decision is made.

The steps of MDFA and IFA as proposed in paper [3] are as follows:

Step 1 Sort the tasks in the just submitted application A_x in the decreasing order of their reserved CPU utilizations.

Step 2 Sort the processing cores in the decreasing order of their available CPU utilization capacities.

Step 3 Compare the reserved CPU utilization u of the first task with the available CPU utilization capacity C of the first processing core. If $u \leq C$, do schedulability test, or reject the application A_x and end the allocation for the application A_x. If the task is schedulable, pre-allocate the task to the processing core and go to step 2, or reject the application A_x and end the allocation for the application A_x.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

VI. CONCLUSION

This paper is a study of different scheduling algorithms and the advantages of using hybrid scheduling in multicore architecture, which allows real time application to run with non real time applications concurrently and supports parallelism among tasks within an application. In this paper load balancing algorithm is discussed which is used to balance the load among the different cores which are homogeneous.

REFERENCES

[1]. A. Silberschatz, P. B. Galvin, G. Gagne, "Operating System Concepts", 7th ed., John Wiley & Sons, 2005.
[2] Milan Milenkovic, "Operating System Concepts and Design", Second Edition McGraw Hill International, 1992.
[3]. PengliuTan,Jian Shu and Zhenhua Wu "A hybrid real-time scheduling approach on multi-core architectures" , vol. 5, no. 9, september 2010.
[4]. Syed Nasir Mehmood Shah, Ahmad Kamil Bin Mahmood, Alan Oxley"hybrid scheduling and dual queue scheduling"978-1-4244-4520-2,2009 IEEE
[5] Gopalakrishnan T R Nair ,Research Associate, Real Time Systems Group ,RIIc,Sr.Lecturer, toce,"critical task re-assignment under hybrid scheduling approach in multiprocessor real time systems"

BIOGRAPHY

VINUTA A PATIL is working as Assistant professor in the department of Computer science and engineering, SJBIT, BANGALORE, Karnataka, India