D-MMLQ Algorithm for Multi-level Queue Scheduling

Manupriya Hasija TIT&S, Bhiwani
Akhil Kaushik TIT&S, Bhiwani
Satvika Kaushik TIT&S, Bhiwani
Manoj barnela TIT&S, Bhiwani

Summary
Since the advent of Operating system, the focus of the work is aimed at better resource scheduling. Handling of multiple tasks at a single time by single processor is old story now. Today is the time of multitasking as well as multiprocessing. The scheduling of processes or tasks have taken a whirlwind after the concept of multiprocessing. There are a lot of well-known scheduling algorithms for job allocation on uniprocessor system but they may not fit well under the system having multiple processors. Altogether the thought of multiprocessing systems gives overheads but still make the idea amazingly interesting. This paper takes into account the deadline concept to realize the real-time needs of a multiprocessor system with multiple queues and tries to conceptualize an innovative algorithm for the same.

Keywords
Multiprocessor scheduling, MLQ, EDF, D-MMLQ, GridSim

1. Introduction

The core idea of any system is to improve the efficiency and performance by either manipulating the inputs or modifying the implementation methods to generate better output. With the advent of computer systems, the human part has always tried to do the same and has succeeded in most of his expeditions. After the invention of Operating System, the computer era has revolutionized. A lot of research has been undertaken to study the scheduling algorithms for single processor systems earlier and now for the multiprocessor systems. However, most of the concepts of uniprocessor organization are not applicable to its multiprocessor counterpart due to reasons like availability of several processing elements, load balancing, parallel processing of data, dependency of processes on each other, etc.
Now in trend are real-time embedded systems which find applications in many diverse areas, including automotive electronics, avionics, telecommunications, space systems, medical imaging, and consumer electronics. A real-time system as defined as an information processing system which has to respond to externally generated input stimuli within a finite amount of time with the maximum accuracy. The correctness depends not only on the logical result but also on temporal accuracy to the same extent; the failure to respond in time is as bad as the wrong response [1]. For example in avionics, flight control software must execute within a fixed time interval in order to accurately control the aircraft. In automotive electronics there are tight time constraints on engine management and transmission control systems that derive from the mechanical systems that they control.
Thus for the sake of best results, the point under consideration especially for avoiding deadline misses is efficient scheduling. Multiprocessor real-time scheduling theory also has its origins in the late 1960s and early 1970s. Multiprocessor real-time scheduling is intrinsically a much more difficult problem than uniprocessor scheduling [3]. Some of the outcomes of single processor can be directly generalized to the case of multiprocessors. However, implementing multiple processors instead of single processor brings a new facet in job scheduling. An important point to note here is that a task may choose only one processor among several free processors to make scheduling complicated and amazingly interesting.

1.1. Multiprocessor Scheduling

Multiprocessor scheduling is an innovative approach to allocate several jobs to numerous processors at same time. The key idea here is to find which processor is ideal to handle which job. Working of a multiprocessor scheduler is shown in Fig. 1.
Multiprocessor scheduling can be defined as an attempt to solve following two key problems:
1) Allocation Problem: which processor should execute which task?
2) Priority Problem: which task will be executed in which order?
Broadly, scheduling algorithms can further be divided into two categories: preemptive and non-preemptive depending upon whether a process can be interrupted in between to give way to other processor. In preemptive scheduling, CPU can stop executing a process and allocate resources to another needy process; while non-preemptive scheduling does not interrupt execution of a process. Due to this reason, preemptive scheduling is a bit costlier approach.

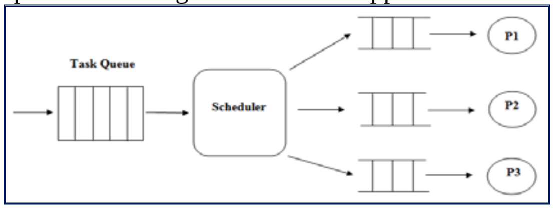

Fig1. Working of Multiprocessor Scheduler

Manuscript received July 5, 2014
Manuscript revised July 20, 2014

The mechanism or policy that is used to efficiently manage the access to and use of a resource by various processes is popularly known as resource management. This allocation and de-allocation of resources to various tasks and jobs by a processor is also called scheduling and the scheduling system is known as scheduler. An important consideration in scheduling is the consumer and resource's perspective. The consumer's outlook depends on how well the scheduler manages the resources i.e. performance. On the contrary, the resource's viewpoint is defined in the terms of how difficult or costly it is to access the resources i.e. efficiency [12].
Whenever the processor becomes idle, the operating system must select one of the jobs in the ready queue for execution. The current distributed computing era is all about the management and allocation of system resources relative to computational load of the system. In present scenario of supercomputing, large scale parallel machines are essential to meet the ever increasing needs of demanding applications. In such a context, a need for effective scheduling strategies is of vital importance, to meet the desired quality of service parameters from both user and system perspectives. Specifically, the desire to reduce response time, waiting times, processor idle time, problem of starvation and maximize the throughput, processor utilization, resource utilization etc. Scheduling algorithms demand a proper balance between fair- share and preemptions taking place.
Scheduling techniques have a significant impact on the performance characteristics of computing systems. Early strategies used queue-based approaches to schedule the tasks which later shifted towards priority- based approaches and later mixes of various dissimilar approaches overtook the market. Many different approaches and metrics of performance have been proposed to achieve the optimal solution for all resource management needs, which will be discussed in the following section.

2. Prior Work

First Come First Serve (FCFS) also referred as FIFO (First In First Out) algorithm comes under the category of Queuing algorithm and is the most basic algorithm. It treats every task equally and executes them according to their arrival times. FCFS [2] is very easy to implement, incurs low computational cost and is an optimal scheduling algorithm. However, with increase in load, the performance shows a steep downfall.
Earliest Deadline First (EDF) is a priority based algorithm which has two variations based on whether preemptions are allowed or not [2]. Non-preemptive-EDF shows comparative low execution overhead while preemptive-EDF is better with performance metrics. For preemptive tasks EDF is proved to be an optimal algorithm. But, similar to FCFS, the performance of EDF also deteriorates as the load increases.
Group- EDF (g-EDF) is a variation of EDF which groups together the tasks having almost similar deadlines and SJF algorithm is used within the group for scheduling [4]. G-EDF gives better performance in terms of success ratio (number of tasks that have been successfully scheduled to meet their deadlines). It has computational complexity almost comparable to EDF.
Shortest Job First (SJF) is a priority based non-preemptive scheduling strategy that employs the deadline constraint to schedule the tasks. The task with shortest expected execution time is given priority to those having larger execution time [5].
Backfilling [6] [7] is a concept introduced to extend FCFS to improve resource utilization. Backfilling allows a lower priority task to start before the higher one in the case when it can fill the gap that is in the queue to reduce the processors' idle time. It very effectively improves average turnaround time.
Conservative Backfilling [8] is a variation of Backfilling that focuses on elimination of Starvation problem by performing backfilling after checking that it does not cause a delay any previous job in queue.
Aggressive Backfilling/ EASY (Extensible Argonne Scheduling system) implements the aggressive version [9] of backfilling such that any job can be used to backfill provided it does not delay the first job in the queue. Since the queuing delay for the job at the head of the queue depends only on jobs that are already running, and these jobs will eventually either terminate or be terminated when they exceed their estimated runtime, starvation is eliminated.
Best Gap (BG) is similar to conservative backfilling. Conservative backfilling chooses the first gap identified in the cluster, while BG chooses the best gap on the basis of some evaluations. In case of a tie between two gaps based on the evaluation done, first gap is chosen. There are still some other variations of Best Gap like Best Gap- Earliest Deadline First [13].

3. Simulation Tool

Simulation is the imitation of real things, processes or affairs. The act of simulation generally entails representing certain key characteristics or behaviors of a selected physical or abstract system. Grid environment also can be simulated using several Grid simulators e.g. GridSim, Eclipse etc. Grid simulators enable Grid users to work on Grid like environment without worrying about the other external factors that may influence the Grid environment. The simulation tool employed to implemented D-MMLQ algorithm is GridSim. GridSim toolkit provides a modular environment composed of independent entities corresponding to the real world with the main functionality of the scheduler divided into separate parts. In GridSim it is easier to simulate different types of job, scheduling algorithms or optimization criteria by making small changes in the existing simulator. For example, to test some new scheduling algorithm only the scheduler class is to be modified. Similarly to schedule different type of jobs, only the data set used by job loader and possibly corresponding objective function in the scheduler is to be changed, rest of the classes stay intact, Hence, providing an ease to repeat the experiments with the exactly same setup. The changes are encapsulated and the results can be easily compared.

4. Proposed Solution

In this section, the proposed solution for scheduling the jobs using Deadline based- Modified Multi-Level Queue (D-MMLQ) Scheduling technique in Grid environment is briefly explained. The user submits gridlets along with the requirements to the Alea GridSim scheduling system. The submission of gridlets to the resources involves checking the suitability of the available PEs. If the requirement is satisfied, the gridlets are assigned to the respective resources. This technique uses a dynamic priority mechanism to schedule the gridlets to the system efficiently and maximize the resource utilization. The gridlets waiting for the service is placed in the waiting queue. The gridlets that are scheduled in the queue are executed.

The algorithm proposed in this paper is based on this renowned concept of multi-level queue which will reduce the problem of starvation of low priority jobs for long time despite the availability of enough resources. The concept of multi-level queue scheduling strategy maintains two separate queues where jobs are permanently assigned to the queues. The jobs are executed by applying any particular scheduling algorithm. Every queue has its own scheduling policy. The main idea behind it is to separate jobs with different characteristics. In general the scheduler is defined based on various parameters including: when to demote the priority of job, which scheduling algorithm is to being applied, the number of queues, etc. The proposed work employs the parameter of selection of the queue to be executed.

Firstly, the jobs entering are allowed to enter any queue at random basis. The selection of the queue is done on First Come First Serve (FCFS) basis as FCFS has been proved to be an optimal scheduling algorithm (i.e. FCFS will surely come up with a schedule for a set of jobs if there exists one). However, the gridlets present in the queues are executed based on EDF scheduling policy. The gridlets with the earlier deadlines are assigned the higher priority, and a higher priority request will be executed first. The gridlets having their deadlines close completes its execution quickly. All gridlets gets an opportunity to execute and thus reduces starvation of gridlets by promoting the gridlets in lower queues to a higher priority.

5. D-MMLQ Algorithm

The Deadline based Multi-Level Queue (D-MMLQ) Scheduling algorithm is basically divided into two phases. The first phase is concerned chiefly with the allocation of jobs to various queues, whereas the second phase handles the execution of jobs. Phase 1 uses Wallclock comparator for finding out which queue gets executed first, which basically uses improvised First Come First Serve (FCFS) basis. After the queue selection, jobs are executed in phase 2 on the basis of Earliest Deadline First (EDF). The significant point here is that D-MMLQ specially looks for starved jobs and makes sure the number of starved jobs is zero or as minimum as possible. The selection of cluster (of processing elements) is done automatically by GridSim simulator.

```
// Phase 1: Job Submission
1:  Queues: = 1: N.
2:  Sort N queues by using Wallclock comparator.
3:  For i: = 1 to N
4:  Set current_queue: = queues[i];
5:  Insert the jobs in the current queue at last.
6:  Sort current_queue by comparing deadlines of jobs.

//Phase 2: Job Execution
7:  For all jobs in current_queue repeat
8:  If job j can be executed then
9:  Set k: = select cluster;
10: Remove j from current_queue and send it on k;
11: End if
12: End for
13: End for
```

6. Performance Evaluation

In this section, the performance of D-MMLQ scheduling strategy through various experiments using Alea simulator, an extension of GridSim simulation toolkit is discussed. The experiment involves 5000 jobs which were executed on 14 clusters having numerous of CPUs. The simulation is implemented by providing the input data-set "metacentrum.mwf" and all the jobs submitted complete over a particular span of time. These graphs show the differences among the efficiencies of algorithms. FCFS shows poor results as per the machine usage parameter. FCFS is not able to utilize available resources when the job in the queue requires some specific and currently non-available processor(s). At this point, other jobs in the queue can be executed to improve the utilization value. This is the main motivation working behind D-MMLQ. The results show that D-MMLQ is able to show some increase the machine usage by shifting the jobs among the queues. Still, D-MMLQ as it employs a mixture of EDF and FCFS will not allow any job to starve, hence making fair-share decisions. This increases the machine utilization and efficiency. The simulation is done by providing input data set and it completes all the jobs submitted to the grid over a span of time. The following graphs show the results of D-MMLQ algorithm:

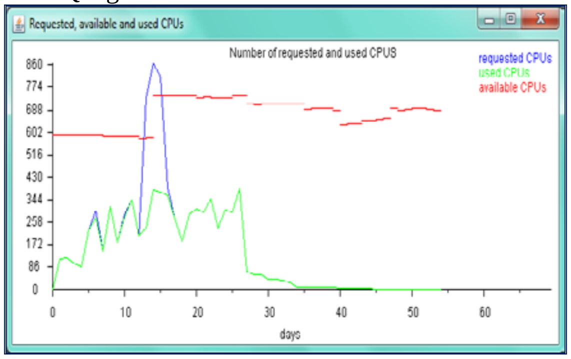

Fig 3. Numbers of requested, available and used CPUs on D-MMLQ

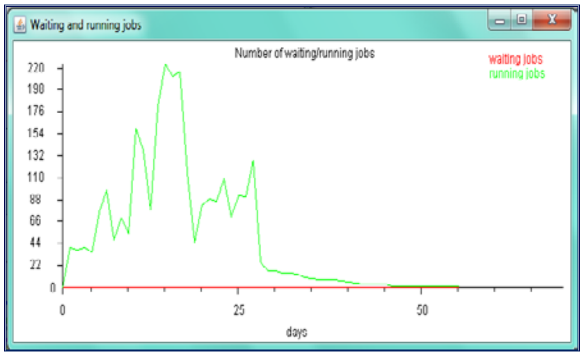

Fig 4. Number of waiting and running jobs on D-MMLQ

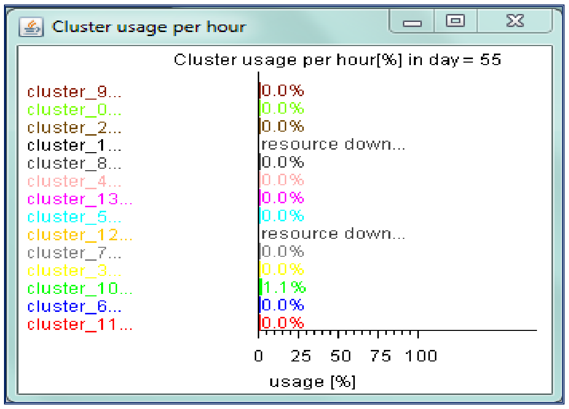

Fig 5. Cluster usage per hour on D-MMLQ

7. Results and Comparisons:

The newly proposed D-MMLQ algorithm works on two principles: Wallclock comparator for inserting jobs in the multilevel queues and then using EDF for executing jobs in each queue. This combined innovative approach proposed in D-MMLQ algorithm provides better results as compared to the EDF scheduling algorithm. The following graphs show the results of EDF algorithm implemented on the similar input set as of D-MMLQ:

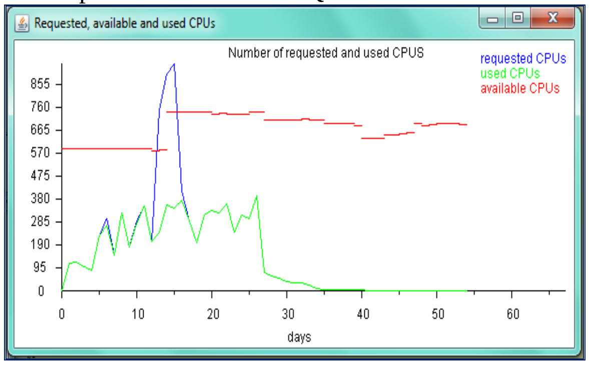

Fig 6. Number of requested, available and used CPUs on EDF

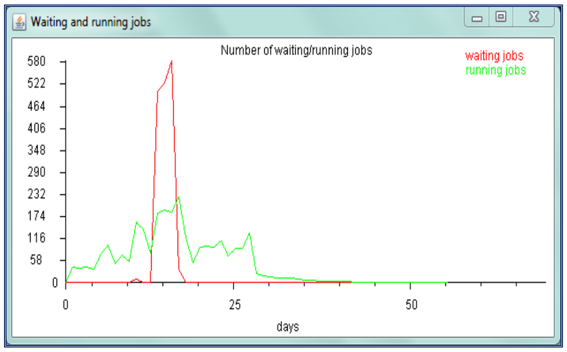

Fig 7. Number of waiting and running jobs on EDF

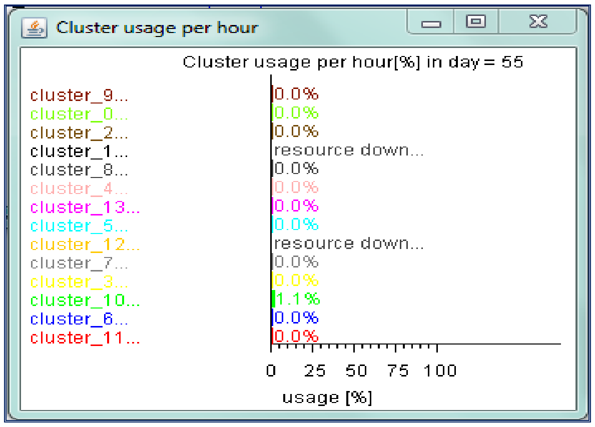

Fig 8. Cluster usage per hour on EDF

Another factor that classifies the supremacy of D-MMLQ over its counterpart EDF is Normalized User Weight (NWT). The least NWT value, the better is the scheduling algorithm. NWT value for EDF algorithm is 2.5579, which is reduced by approximately 16% by D-MMLQ algorithm to 0.422154. Hence, it is observed that D-MMLQ is supreme to EDF in all aspects of performance in multiprocessor environment.

8. Conclusion & Future Scope:

This paper describes a new and innovative scheduling algorithm named "D-MMLQ" for multiprocessor scheduling. The proposed algorithm fuses two vital concepts for handling job allocation and execution through multi-level queue. The approach proposes that the starvation problem of low priority jobs or jobs at lower end of queue, hence increasing the overall competence of multiprocessor system. The graphs show less average waiting time and better utilization of resources by D-MMLQ algorithm in comparison to traditional EDF algorithm. Furthermore, the Normalized User Weight (NWT) factor is the least possible value obtained till now by any popular scheduling algorithm. Hence, it can be concluded that the D-MMLQ algorithm proposed in the paper is the best scheduling algorithm devised till today.
The following topics are in the scope for potential work direction:

1) To analyze the algorithms further some more effective parameters like critical instant, utilization bound and bounded response time can be used.
2) The algorithm can be further improved by applying the quasi-deadline concept.
3) This concept can be further explored on heterogeneous platform.
4) Schedulability analysis of these algorithms can further prove its optimality.

Acknowledgement

The authors would like to thank Dr. Mukesh Kumar, Mr. Parveen Kumar and Mr. Deepak Singla for their extensive help and constant discussions.

References

[1] A. Burns & A. Wellings. "Real-Time Systems and Programming Languages". Addison Wesley Longmain, April 2009.
[2] Silberschatz ,Galvin and Gagne, "Operating systems concepts", 8th edition, Wiley, 2009.
[3] C. L. Liu & J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal of the ACM, Vol. 20. No. 1, pp. 46-61.
[4] W. Li, "Group-EDF- A New Approach and an Efficient Non-Preemptive Algorithm for Soft Real-Time Systems", 2006.
[5] L. Yang, J. M. Schopf & I. Foster, "Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments", SuperComputing2003, Phoenix, AZ, USA, November 15-21, 2003.
[6] D. Lifka, "The ANL/IBM SP scheduling system", JSSPP, 1995.
[7] A. W. Mu'alem and D. G. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with back_lling", IEEE TPDS, 12(6):529.543, 2001.
[8] D. Feitelson, L. Rudolph, & U. Schwiegelshohn, Parallel job scheduling - a status report, June 2004.
[9] D. Lifka, "The ANL/IBMSP scheduling system", Job Scheduling Strategies for Parallel Processing, pp. 295-303, Springer-Verlag, Lect. Notes Comput. Sci. Vol. 949, 1995.
[10] R. Buyya & M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", The Journal of Concurrency and Computation: Practice and Experience (CCPE), 14:1175{1220, 2002.
[11] S. Baruah, "Dynamic- and static-priority scheduling of recurring real-time tasks", Real-Time Systems: The International Journal of Time-Critical Computing, 24(1):99–128, 2003.
[12] A.S. Tanebaun, "Modern Operating Systems", 3rd Edition, Prentice Hall, ISBN: 13:9780136006633, pp: 1104. 2008.
[13] Dalibor Klusacek, Event-based Optimization of Schedules for Grid Jobs, Doctor of Philosophy at the Faculty of Informatics, Masaryk University, Brno, Czech Republic, 2011.

Manupriya Hasija is pursuing her Master's in Computer Science from The Technological Institute of Textiles and Sciences, India. She got her bachelor's degree from Gurgaon Institute of Technology and Management, Gurgaon, India. Her research interests are in the fields of Operating Systems and Scheduling algorithms.

Akhil Kaushik received his Master's degree in Information Technology from Central Queensland University, Australia. Since then he is being with The Technological Institute of Textile and Sciences, Bhiwani, India, Computer Science Department, where he is currently working as an Assistant Professor. His primary research interests lie in the areas of Cryptography, Steganography and Expert Systems. He has about 12 international publications and 6 national publications in the same field.