ISSN:2320-0790

An International Journal of Advanced Computer Technology

A-MMLQ Algorithm for Multi-level Queue Scheduling

[1] **Manupriya Hasija**, [2]**Akhil Kaushik**, [3]**Parveen Kumar**

[1]Research Scholar, [2]Assistant Professor, [3]Research Scholar
[1,2]TIT&S, Bhiwani, India,
[3]BITS, Pilani

Abstract: This being the era of advancement in computing domain, the emphasis is on better resource scheduling. Scheduling is not confined to dealing multiple tasks by a single processor. It's a dawn with multiprocessing and multitasking. Although multiprocessor systems impose several overheads but still make the concept amazingly interesting. The scheduling field has taken a whirlwind after the notion of multiprocessing. Many of the uniprocessor algorithms do fit well under the multiprocessor systems but, still necessitating a further development aiming solely on multiprocessor scheduling. This paper thus sketches a new idea to modify and extend the well-known multi-level queue scheduling, taking into account the arrival time/ arrival sequence to conceptualize an innovative scheduling algorithm.

Index Terms: Multiprocessor scheduling, Multi-Level Queue Scheduling, FCFS, A-MMLQ, GridSim.

I. INTRODUCTION

Today, real-time embedded systems find applications in many diverse areas, including automotive electronics, avionics, telecommunications, space systems, medical imaging, and consumer electronics. Real-time systems are driven by a profit motive and they are in huge demand due to rapid technological developments in mostly applications all around the world. 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 [2]. 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.

A. 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 (Fig.1). Hence, 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?

On the basis of above criteria the scheduling may further be classified as:

B. Allocation/Migration:

1) No migration (i.e., task partitioning)

2) Migration allowed, but only at job boundaries (i.e., dynamic partitioning at the job level)

3) Unrestricted migration (i.e., jobs are also allowed to migrate).

C. Priority:

1) Static

2) Dynamic but fixed within a job

3) Fully dynamic

Scheduling algorithms can be further classified as follow:

1) Preemptive: Tasks can be preempted by a higher priority task at any time.

2) Non-preemptive: Once a task starts executing, it will not be preempted and will therefore execute until completion.

3) Cooperative: Tasks may only be preempted at defined scheduling points within their execution. Effectively, execution of a task consists of a series of non-preemptable sections.

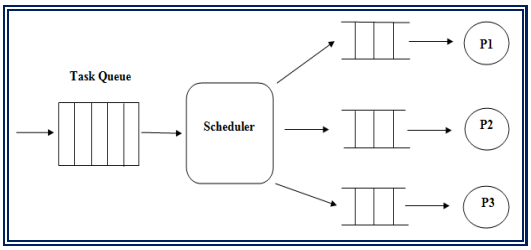

Fig. 1: Working of Multiprocessor Scheduler

The mechanism that is used to effectively manage the access to and use of a resource by various processes is commonly known as resource management. This allocation and de-allocation of resources to various jobs by a processor is also called *scheduling* and the scheduling system is known as *scheduler*. A chief consideration in scheduling is the consumer and resource's perspective. The consumer's viewpoint is defined in terms of how well the scheduler manages the resources i.e. performance. On the contrary, the resource's outlook depends on how difficult or costly it is to access the resources i.e. efficiency [12].

As soon as the processor becomes idle, one job must be selected from the ready queue for execution. The present-day distributed computing era is all about how well the system resources are allocated and managed relative to computational load on the system. In current state of supercomputing, large scale parallel machines must critically meet the ever increasing needs of demanding applications. In such a context, a need for effective scheduling strategies is vigorously important, to meet the desired quality of service parameters from both user and system angles. 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 an appropriate balance between fair- share and preemptions taking place.

Scheduling techniques have a significant impact on the performance characteristics of computing systems. Earlier strategies in trend were queue-based approaches to schedule the tasks while later the fad demanded priority- based approaches and then mixes of various dissimilar approaches overtook the market. Many diverse approaches and metrics of performance have been proposed to achieve the finest solution for all resource management needs, which will be discussed in the following section.

II. PRIOR WORK

First Come First Serve (FCFS) also referred as FIFO (First In First Out) algorithm is categorized under Queuing algorithm and is one of the elementary algorithms. It gives every task equal importance and executes them according to their arrival times. FCFS [2] is very easy to implement, invites little computational cost and is an optimal scheduling algorithm. However, its performance depends inversely on load placed.

Earliest Deadline First (EDF) is a priority based algorithm with two famous variants grounded on whether preemptions are allowed or not [2]. Non-preemptive-EDF spectacles comparative low execution overhead while preemptive-EDF is

healthier with performance metrics. For preemptive tasks EDF is verified to be an optimal algorithm. But, similar to FCFS, the performance of EDF also worsens as the load surges.

Group- EDF (g-EDF) is a variant of EDF that groups together the tasks having more or less 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 practically comparable to EDF.

Shortest Job First (SJF) is a kind of 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 conception introduced to extend FCFS to improve resource utilization. Backfilling allows a lower priority task to start before the higher one when it can fill the gap that is in the queue to reduce the processors' idle time. It very successfully improves average turnaround time by folds.

Conservative Backfilling [8] is a type of Backfilling that is motivated to eradicate *Starvation* problem by performing backfilling after testing that it does not cause a delay to any previous job in the queue.

Aggressive Backfilling/ EASY (Extensible Argonne Scheduling system) gears the aggressive version [9] of backfilling such that any job can be used to backfill as long as 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 executing by this time, and these jobs will eventually either terminate or will be forcefully terminated when they overdo their estimated runtime, starvation is eliminated.

Best Gap (BG) is alike conservative backfilling. Conservative backfilling picks the first gap identified in the cluster, while BG selects the best gap on the basis of some evaluations. In case evaluation results a tie between two gaps, first gap is elected. BG's success story is itself voiced by the accomplishment of modifications and extensions of *Best Gap* like *Best Gap- Earliest Deadline First* [13].

III. SIMULATION TOOL

Simulation is the imitation or reproduction of the appearance, character, real entity, process, affair or conditions. The act of simulation generally involves representing certain significant characteristics or behaviors of a selected physical or abstract system. Grid environment can also be simulated using several Grid simulators e.g. GridSim, Eclipse etc. Grid simulators enable Grid users to work on Grid alike environment without worrying about the other external factors that may influence the Grid environment. The simulation tool employed to implement A-MMLQ algorithm is *GridSim*. GridSim toolkit provides a modular environment composed of self-governing entities corresponding to the real world with the main functionality of the scheduler divided into distinct parts. In GridSim it is easier to simulate diverse kinds of job, scheduling algorithms or optimization criteria by making small changes in the existing simulator environment. For instance, to test some new scheduling algorithm only the scheduler class needs 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 altered, rest of the classes stay intact, Hence, providing an easiness to reiterate or repeat the tests with the exactly same setup. The modifications are encapsulated and the results can be easily compared.

IV. PROPOSED SOLUTION

In this section, the proposed solution for scheduling the jobs using Arrival based- Modified Multi-Level Queue (A-MMLQ) Scheduling technique in Grid environment is briefly explained. The user submits gridlets along with their requirements to the Alea GridSim scheduling system. The submission of gridlets to the resources involves checking whether available PEs fit with the gridlets. 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 and reduce starvation. The gridlets waiting for the service are 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 well-known concept of multi-level queue scheduling which will reduce the problem of starvation of low priority jobs for long time despite the availability of enough resources. In multi-level queue scheduling strategy there are two separate queues where jobs are permanently assigned to the queues. The jobs are

executed by applying certain scheduling algorithm. Every queue has its own scheduling policy. The main motive 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 first.

Firstly, the jobs entering are allowed to enter any queue randomly. 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).

Alongside, the gridlets present in the queues are also executed based on FCFS scheduling policy. The gridlets that arrived first are assigned the higher priority, and are picked to be executed first. This selection for execution follows the First In First Out policy. All gridlets get an equal opportunity to execute and thus reduces starvation of gridlets. This algorithm respects the fair-share policy.

V. A-MMLQ ALGORITHM

The Arrival based Multi-Level Queue (A-MMLQ) Scheduling algorithm is mainly split into two phases. The first phase concerns with the allocation of jobs to various queues, whereas the second phase manages the execution of jobs. Phase 1 uses *Wallclock comparator* to pick the queue to be executed first, which principally uses improvised *First Come First Serve (FCFS)*. After the queue selection, jobs are executed in phase 2 on the same basis of *First Come First Serve (FCFS)*. The significant point here is that A-MMLQ specially looks for fair share among all jobs and also 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 arrival times 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

VI. PERFORMANCE EVALUATION

In this section, the performance of A-MMLQ scheduling strategy through various experiments using Alea simulator (GridSim simulation toolkit) is discussed. The experiment involves 5000 jobs which were executed on 14 clusters having abundant 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 variances among the efficacies of algorithms. FCFS shows poor results as per the machine usage parameter. FCFS alone cannot utilize available resources when the job in the queue requires some specific and currently non-available resource(s). This is the main motivation working behind A-MMLQ. The results show that A-MMLQ is able to show some increase in the machine usage. Still, A-MMLQ as it employs a FCFS will not allow any job to starve, hence making fair-share decisions. This increases the machine utilization and efficiency. The following graphs show the results of A-MMLQ algorithm:

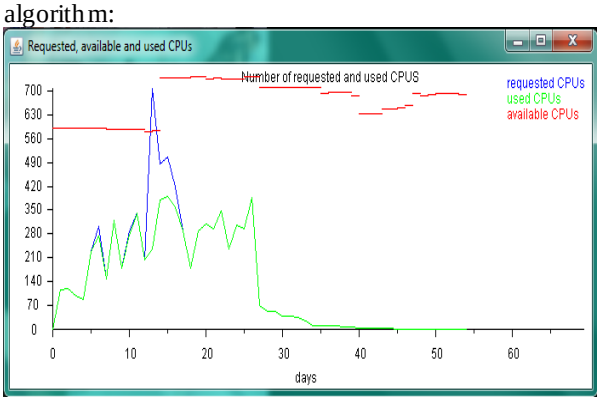

Fig. 3: Numbers of requested, available and used CPUs on A-MMLQ

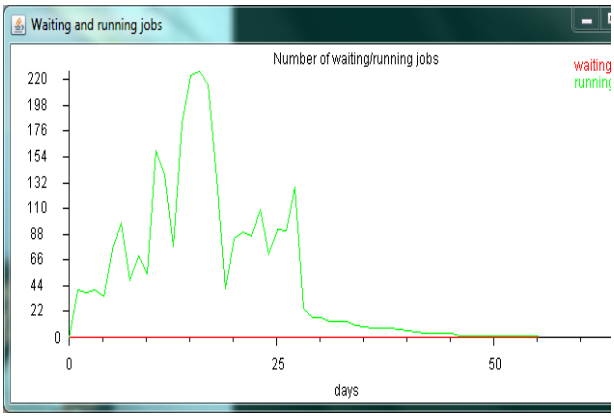

Fig. 4: Number of waiting and running jobs on A-MMLQ

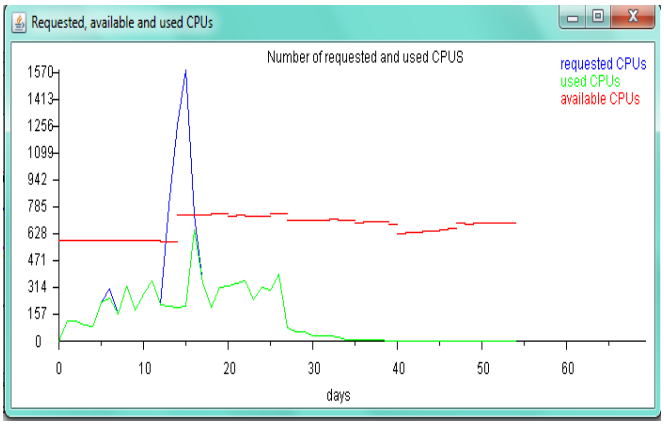

Fig. 6: Number of requested, available and used CPUs on FCFS

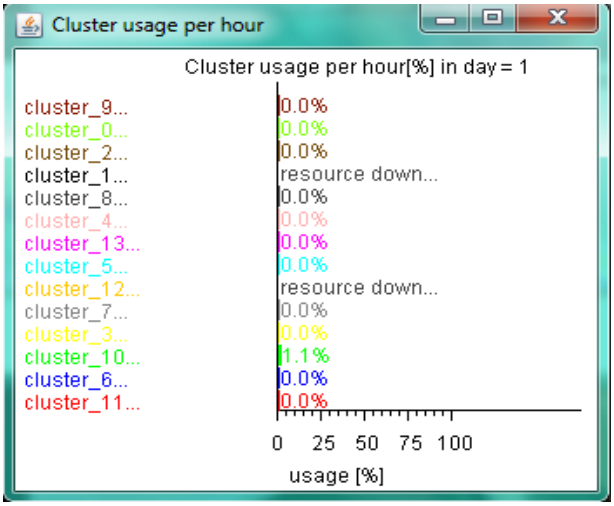

Fig. 5: Cluster usage per hour on A-MMLQ

Fig. 7: Number of waiting and running jobs on FCFS

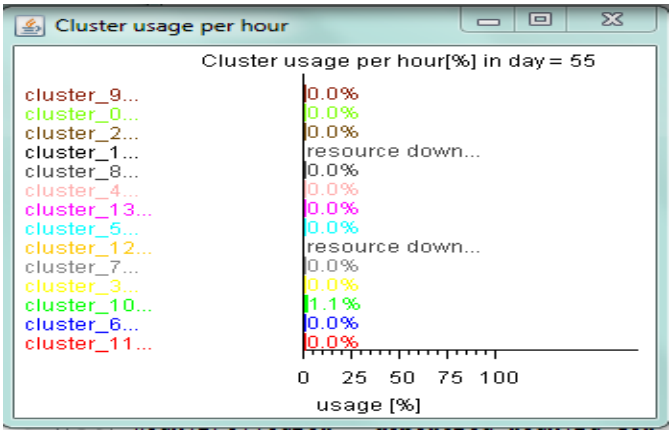

Fig. 8: Cluster usage per hour on FCFS

VII. RESULTS & COMPARISONS

The newly proposed A-MMLQ algorithm works on Wall clock comparator for inserting jobs in the multilevel queues as well as for executing jobs from each queue. This innovative approach proposed in A-MMLQ algorithm provides better results as compared to the FCFS scheduling algorithm in some aspects. The following graphs show the results of FCFS algorithm implemented on the similar input set as of A-MMLQ:

Another factor that classifies the supremacy of A-MMLQ over its counterpart FCFS is Normalized User Weight (NWT). The least NWT value, the better is the scheduling algorithm. NWT value for FCFS algorithm is 2.5579, which is reduced by folds with A-MMLQ algorithm to 0.1320. Hence, it is observed that A-MMLQ is equivalent to FCFS in various aspects of performance in multiprocessor environment.

VIII. CONCLUSION & FUTURE SCOPE

This paper describes a new and advanced scheduling algorithm named "A-MMLQ" for multiprocessor scheduling. The proposed algorithm uses the same concept for handling job allocation and execution through multi-level queue. The approach proposes that the starvation problem and waiting time of low priority jobs or jobs at lower end of queue are effectively managed, hence increasing the overall competence of multiprocessor system. The graphs show very less average waiting time and better utilization of resources by A-MMLQ algorithm in comparison to traditional FCFS 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 A-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) The further analysis of the algorithm can be done using various parameters.
2) The algorithm can be further improved by fusing it with other existing scheduling algorithms.
3) This concept can be further explored on heterogeneous platform.
4) Schedulability analysis of these algorithms can further prove its optimality.
5) Backfilling strategy if blended with this algorithm may also lead to success.

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 backfilling", *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, 2011. Her research interests are in the fields of Operating Systems, Scheduling algorithms and cryptography.

Akhil Kaushik received his Master's degree in Information Technology from Central Queensland University, Australia, 2006

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.