ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

Dead Line Scheduling In Grid Computing Environments

R. Ananthi Lakshmi[1], S.Vidhya[2]

M Phil Scholar, Computer Science, Kg College of Arts and Science, Coimbatore, India[1]

Assistant Professor, Computer Science, Kg College of Arts and Science Coimbatore, India[2]

Abstract: Grid, it is extensively used in science and technology. It fully provides a resource sharing among different organizations. Now days, grid's usage is rapidly increasing in industry also. In Dead Line Scheduling creates a fundamental in choosing the right order of task scheduling. In the traditional dead line scheduling such as Earliest Deadline First (EDF), provides a results in consuming more resources. On the other hand, we can use the tasks in decreasing order of their deadlines in Latest Deadline First (LDF).

Keywords: Dead Line Scheduling, Computational Model, Task Model, Redundant Scheduling.

I. INTRODUCTION

Grids infrastructures, large scale computational platforms and it are a cycle sharing systems have grown in running the areas High Energy Physics, climate production and bio formatics. These systems provide high computational power in distributed around the internet.
Many of these applications are fully composed by computational tasks that require specific deadlines to meet for the successful applications. These deadlines are time-constrained or real time nature of applications first. The nodes in such a systems are very highly speed. In addition, the capacity of individual nodes is also varying loads.
In a redundant scheduling, the tasks assigned to multiple nodes to improve the completion and employed in such systems. These systems provide a scalability and enormous computational power by idle processing cycles from computing hosts distributed around the internet. Their low deployment and operational cost in addition to their scalability has made these infrastructures attractive for hosting large scale time-critical applications.

II.PROPOSED WORK

The redundant scheduling, it creates a dilemma in choosing the right order of task scheduling. It gives a low dead line asks. As in dead line based scheduling algorithms such as Earliest Deadline First (EDF), it consumes more resources. Since this tasks have more level deadlines and need more resources for their timely completion. On the other hand, the tasks decreasing order of their dead lines such as Latest Dead line First (LDF) will provide better resource utilization. It provides a tighter deadline tasks.

In this paper, we propose a new scheduling algorithm called Limited Resource Earliest Deadline (LRED).That is specifically designed to address the dynamic computational environments. LRED, it couples redundant scheduling with the dead line driven scheduling. LRED by limiting the number of resources consumed per tasks. An important feature of LRED is that it can achieve the desired work scheduling.

The design of the LRED algorithm has resulted in the following key research contributions. We define a statistical notion of timeliness for a computational node which can incorporate both inter node heterogeneity as well as intra node dynamism .LRED uses these timeliness values to couple redundant scheduling with dead line driven scheduling in a seamless manner. LRED is a generalization of EDF and LDF. So that, by tuning this parameter, LRED reduces to EDF in one extreme and to LDF in another extreme.

III.LIMITED RESOURCE EARLIEST DEADLINE SCHEDULING

In this section, we present the Limited Resource Earliest Deadline Scheduling (LRED): In a general deadline driven scheduler, explicitly incorporates the redundant scheduling, and it provides a flexible way to exploit the computation system. The shorter deadline tasks consumes a larger number of resources and more likely to fail with the passage of time. LRED uses these insights to exploit the throughput fairness trade off. We first present the high level intuition behind the algorithm it is followed by the key concepts used by this algorithm and then describe the algorithm working in detail.

N: Total no. of tasks in the task pool
L: Total no. of workers available at the scheduler
S_k: Set of tasks with ascending deadline values each of which can be completed with probability TSR by the workers{W1…WK}for k=1,2,…,max .Sk could be empty.Sx=Set of tasks

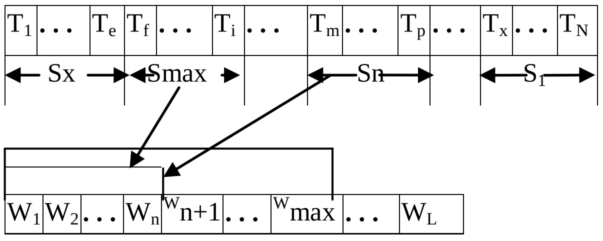

T_1 . . . T_e T_f . . . T_i . . . T_m . . . T_p . . . T_x . . . T_N
Sx
Smax
Sn
Sr
W_1 W_2 . . . W_n $^{W}n+1$ . . . ^{W}max . . . W_L

Figure 1.Partitioning of the task list by LRED

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

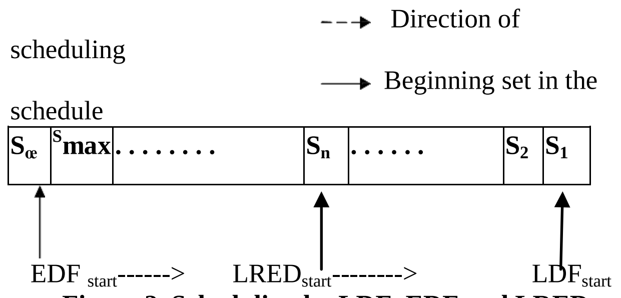


Figure 2. Scheduling by LDF, EDF and LRED

The LRED works by limiting the number of resources consumed per task while scheduling the selected tasks in earliest deadline order. To achieve this goal, we can sorts the task pool in LRED for increasing the order of deadlines. So that shorter deadline tasks requires more resources compared to the higher deadline tasks. Then the LRED schedules the tasks earliest deadline first order starting from the first task we required. The LRED to control the throughput fairness tradeoff the system.

IV.KEY CONCEPTS

Consider a set of N workers and L tasks in the systems. Let us assume that the task list is sorted in increasing order of task deadlines. Then we assume the workers queue is sorted in decreasing order of the mean timeliness of the workers.

Definition 1: k-dependent task
A task is said to be k-dependent if it needs exactly the k most timely workers in the worker queue to work successfully.

Definition 2: k-dependent task (S_k)
The set of all k-dependent tasks in the task queue. Similarly the tasks T_f to T_i belongs to the set Smax, while tasks Tx to TN belongs to the set S1.S represents the tasks that cannot be successfully completed with any number of works from{W1….WL} and with the current worker pools. Where max is the maximum number of workers required by any task in the task list. Note that the size of these set Sk could be zero. Which means that no tasks which means completely within the tasks.

ALGORITHM 1 LRED

```
 1: W ⟵ Set of all available workers
 2: Sort W in decreasing order of T
 3: Sort the task pool in increasing order of D
 4: While W is non-empty do
 5: Organize the task pool into the list {$S_1$,$S_2$,…Smax}
    Based on T of workers in W
 6: V ⟵ Set of all tasks in the list{Sn,…,$S_2$,$S_1$}
 7: if V is non-empty then
 8: T ⟵ First task from the first non-empty set $S_k$ in V
 9: Schedule T to k most timely workers
10: Update W by removing the k assigned workers
11: else if n <max then
12. LRED(n+1)
```

Algorithm 1 shows the pseudo code for LRED.It takes a parameter n which corresponds to the set Sn to be used as the set pointer LRED $_{start}$. The basic algorithm works by scheduling the group of the n most timely workers among the available worker list to the shortest deadline task T in Sn. The value of n signifies which task among all the tasks in the task pool will be chosen to be scheduled first. When n=1, the execution of LRED(1) corresponds to LDF. When n=1, the execution of LRED(1) corresponds to LDF

When n=max, it schedules tasks from Smax until either Smax becomes empty or all the capable workers are moving on to Smax-1.This corresponds to an execution of EDF. Also,to make the algorithm work conserving once it exhausts all tasks in the sets Sk for k=1,……….n,it recursively calls LRED(n+1).

	Task pool	Deadline	LRED(2)	LRED(1)
S1	T_1	D_1	$T_1 € \{W_1, W_2\}$	
	T_2	D_1		
S2	T_3	D_1		
	T_4	D_2	$T_4 € \{W_3\}$	$T_4 € \{W_1\}$
	T_5	D_2		$T_5 € \{W_2\}$
	T_6	D_2		$T_6 € \{W_3\}$

An example schedule by LRED (n) for values of n=1,2.L=6,n=3

The figure gives an illustration of these schedule created by the LRED (N) algorithm for values of n=1,2.The figure shows that a higher value of n=2 produces a lower throughput but completes more short deadline tasks, whereas with a smaller value of n=1,short deadline tasks strave while increasing the throughput .

But completes more short deadline tasks whereas with a smaller value of n=1, short deadline tasks strave while increasing the net throughput. We next provide detailed quqntitave evaluation of this algorithm using a simulation study.

V. THROUGHPUT FAIRNESS TRADEOFF

Figure a and b plot the fairness index FI and throughput respectively for the different scheduling algorithms. As expected, the fairness of LRED increases as n increases. While through put decreases as shown in the figures.EDF shows the highest fairness and lowest throughput.

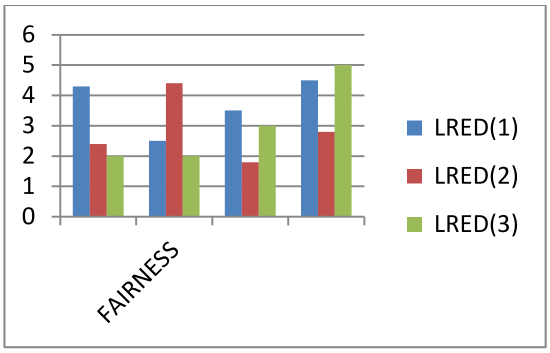
It consists the benefits of rescheduling to availability of newly added resources. The algorithmic scheduling is helps to achieve the minimum execution time of the application. In [1], the author proposed the job scheduling algorithm to schedule the data intensive jobs.

The figure shows the Work Flow Scheduling (WFS) Architecture [4]. It consists Resource Discovery &Monitoring, Grid Information Server, Work Flow task Scheduler, Execution Manager. Grid Information Server is used to maintain the available grid resources. Resource Discovery and Monitoring is to monitor and discovered the grid resources. Execution Manager is to rescheduling the unexecuted tasks. The Resource Discovery and Monitoring component notifies either the overload or new resources to the Execution Manager. The new

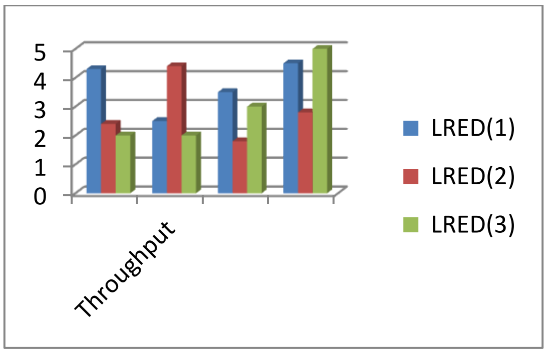
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

schedules generated are submitted to the Execution Manager.

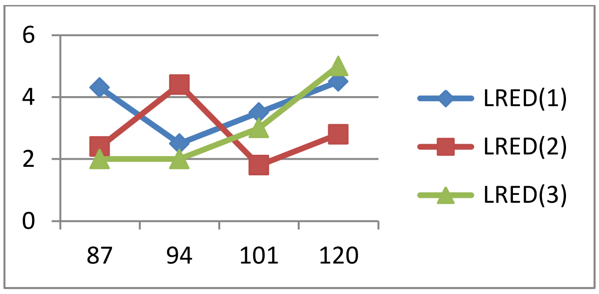


(a) FAIRNESS

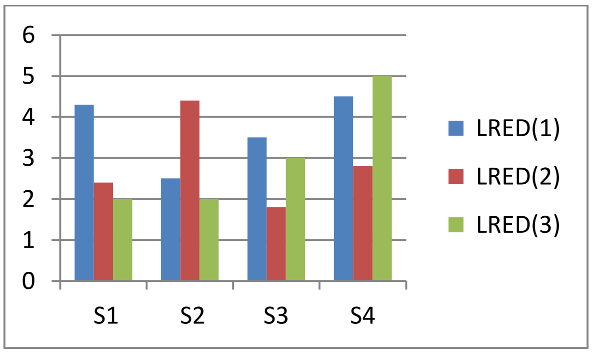


(b) THROUGH PUT

Comparing F1 and Throughput of LDF and Rand with LRED for n=1, 4, 7 in a low TE.



(a) Ratio of completed tasks in deadline bins



(b) Ratio of tasks completed in k-dependent sets

Figure a and b plot the fairness index F1 and throughput respectively for the different scheduling algorithms. As expected ,the fairness of LRED increases as n increases, while throughput decreases as shown in figures.EDF shows the highest fairness and lowest throughput.LDF has the lowest fairness, however its throughput is lower than that of LRED(1) which demonstrates the benefit of scheduling tasks in the increasing order of deadlines within a k-dependent set (S1 in this case).Rand shows slightly higher fairness and lower throughput than LDF because it happens to schedule a greater number of lower deadline tasks than LDF due to the randomness in choosing tasks.EDF does not show any dramatic improvement over LRED(1),because the majority of the tasks that could be finished with unlimited number of workers needed only a size of 7 at maximum.

VI.CONCLUSION

In this paper, we examined the problem of deadline driven task scheduling in a grid environment. We propose a new algorithm called Limited Resource Earliest Dead line(LRED).Our results show that load and the timeliness level of the underlying environment have a significant impact on the throughput fairness tradeoff of task scheduling .We find that LRED provides a powerful mechanism to achieve desired throughput or fairness under high loads and low timeliness environments, where these tradeoffs are most critical.

REFERENCES

[1]. E. Bagarinao, L. F. G. Sarmenta, Y. Tanaka, K. Matsuo, and T. Nakai. Real-Time Functional MRI Analysis Using Grid Computing. High Performance Com-putting and Grid, 2004.
[2] P. Bonetto, M. Guarracino, and F. Inguglia. Integrating Medical Imaging into a Grid Based Computing Infrastructure. Computational Science and Its Applications - ICCSA, 3044:505–514, Apr 2004.
[3] K. Budati, J. Sonnek, A. Chandra, and J. Weissman. RIDGE: Combining Reliability and Performance in Open Grid Platforms. HPDC, Jun 2007.
[4] E. Caron, P. K. Chouhan, and F. Desprez. Deadline Scheduling with Priority for Client-Server Systems on the Grid. ACM GRID, Nov 2004.
[5] C. Germain, V. Breton, P. Clarysse, Y. Gaudeau, T. Glatard, E. Jeannot, Y. Legre, C. Loomis, J. Montagnat, J.-M. Moureaux, A. Osorio, X. Pennec, and R. Texier. Grid-enabling medical image analysis. CC- Grid May 2005.
[6] A. K. F. Khattab and K. M. F. Elsayed. Channel- Quality Dependent Earliest Deadline due Fair Scheduling Schemes for Wireless Multimedia Networks. ACM MSWiM, 2004.
[7] T. Lam and K. To. Performance guarantee for Online Deadline Scheduling in the Presence of Overload. ACM SODA, 2001.
[8] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the Association for Computing Machinery 20, 1:46–61, Jan 1973.
[9] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems. Heterogeneous Comput-ing Workshop, Apr 1999.
[10] A. Takefusa, S. Matsuoka, H. Casanova, and F. Berman. A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid. HPDC, 2001.