Reputation Aware Reliable Distributed Grid Scheduler for Mixed Tasks

Ram Mohan Rao Kovvur
Department Of Computer Science And Engineering, Vasavi College Of Engineering, Hyderabad, Andhra Pradesh, India.

Vijayakumar Kadappa
Department Of Computer Applications, Bms College Of Engineering, Bangalore, Karnataka, India.

Ramachandram S
Department Of Computer Science And Engineering, Osmania University, Hyderabad, Andhra Pradesh, India.

Govardhan A
Department Of Computer Science And Engineering, Jntuh, Hyderabad, Andhra Pradesh, India.

ABSTRACT
Grid scheduling is one of the vital tasks in grid environment, which maps tasks to resources. More recently, a Reputation based scheduling method based on Reliability was proposed for workflow applications to overcome the deficiencies of the existing reputation methods. The method was focused on only computational-intensive tasks. Other recent effort to improve reliability of the scheduling include RDGS (Reliable Distributed Grid Scheduler), which attempts to enhance the Successful Schedule Rate of the mixed tasks by using rescheduling concept. The RDGS method considers various parameters (Priority, Deadline, and CCR) for both computational and communication intensive mixed tasks (Hard, firm, and soft). In this paper we propose a novel method which exploits the merits of both Reliability based reputation method and RDGS. We conducted exhaustive simulation experiments to prove the superiority of the proposed method as compared to other existing methods (GDS, RDGS). The proposed method shows its merit in terms of successful schedule rate, task queuing time and overall time.

Keywords
Grid, Scheduling, Reliability, Reputation, Priority, Deadline, Distributed and Rescheduling.

1. INTRODUCTION
Grid computing and its technologies mainly emerged as the next-generation parallel and distributed computing methodology for fulfilling the mounting demand of the scientific computing community for more computing power. A Grid computing environment is comprises of distributed computers and resources inter-connected locally, nationally or across countries and continents to achieve high performance computing and resource sharing. Thus a computational grid is hardware and software infrastructure is able to provide a dependable, consistent, pervasive and unlimited computing capacity for every user associated in the grid [1] [2] [3].
Grid scheduling is a process of mapping grid tasks to grid resources under multiple criteria and grid environment configuration. The grid scheduler has five phases, which consists of resource discovery, resource selection, task selection, task execution and task monitoring. The responsibility of a scheduler is selecting resources and scheduling tasks in such a way that the user and application constraints are satisfied, in terms of overall execution time and cost of the resources utilized [4].

In general, reliability is an ability of a system to perform and continue its functions in routine circumstances, as well as hostile or unexpected circumstances [5]. The reliability of a grid scheduling scheme depends upon the following three important factors:

- Task execution time: The time taken by the task to complete its execution.
- Communication time: The time consumed in communication in order to obtain the required resources from the various nodes of the grid.
- Rate of failure: The rate of failure of elements of grid computing system such as grid nodes, communication channels.

Recently, a lot of effort kept for fault avoidance and removal has been investigated to improve grid reliability. EunJoung Byun et al. [6] proposes Markov job scheduler based on availability for improving performance and reliability of selecting volunteers according to the needs of the application in Grid computing environment Fiaz Gul Khan et al. [7] Presents a company study of four different fault tolerant techniques such as check pointing, retrying, alternative resource and alternative task to understand the behavior and performance of these fault tolerant techniques in grid computer environment. Mohammed Amoon et al. [8] Brings out a fault tolerant scheduling strategy of the system is to select resources with lowest tendency to fail for computational grids to improve grid performance in terms of throughput, unavailability, turnaround time and fail tendency. Suchang Guo et al. [9] presents an in depth study on grid service reliability modeling and analysis of fault recovery mechanism as Local Node Fault Recovery (LNFR) by resuming the subtask execution on the failed node until the failed node is recovered. Young Choon Lee et al. [10] presents a rescheduling for reliable job completion with the support of clouds resources to reduce delay in job completion. These

methods improved reliability in the grid systems by various fault recovery mechanisms, but however they make grid scheduling decisions centrally.

Mustafizur Rahman et al. [11] Proposes a cooperative and decentralized workflow scheduling in dynamic and distributed sharing environment, the participants in the system, such as the workflow brokers, resources and users who belong to multiple control domains, work together to enable a single cooperative resource sharing environment. Katia Leal et al. [12] Proposes a decentralized model for scheduling independent tasks in federated grids, to implement a mapping strategy on the meta scheduler at each grid infrastructure of the federated grids to reduce makespan of application and increase the performance of the grid infrastructure. In contrast to these methods, Cong Liu et al. [13] developed a general distributed scalable grid scheduler (GDS) for independent tasks with different priorities and deadlines. GDS has three phases, which consists of a multiple attribute ranking phase, a shuffling phase, and peer-to-peer dispatching phase. However, these distributed schedulers do not consider the reliability factor, which is vital in the context of grid environment. Thus there is no guarantee that the task will be scheduled successfully if the system is not reliable.

Recently, we proposed a distributed Grid Scheduler with reliability factor with respect to failure of grid nodes for (i) independent tasks [14] (ii) Mixed tasks RDGS–MT [15]. These two methods indeed considered failure of resource, by rescheduling the tasks to another resource but the real time failure of resource is not monitored.

To further improve reliability of distributed grid scheduler, we considered reputation of resource. Reputation is generally said or believed about a person or things [16]. The reputation is a represent of trust building, as one can trust another based on a good reputation. Thus a reputation is a measure of truth worthiness, in the sense of reliability. Abdul-Rahman et al., [17] states that a reputation is computed based on the information of past behavior. Gheorghe Cosmin Silaghi et al. [18] propose a reputation–based trust management systems and their applicability to grids for the usage of reputation systems for enhancing grids with fault-tolerance in computational grids to improve resource management in traditional grids.

Recently, Xiaofeng Wang et al. [19] propose a method, reliability-driven (RD) reputation using Look-Ahead Genetic Algorithm (LAGA) which utilizes the RD reputation for workflow application to optimize both makespan and reliability. It consists of four components namely resource manager, task scheduler, task monitor and reputation manager. This method focuses on workflow application consisting of only computational intensive tasks.

In this work, we propose a novel distributed grid scheduler (RRDGS) for mixed tasks by exploiting merits of RD reputation and RDGS [15] [19]. The merits of the proposed method include: i) It considered failure of nodes by monitoring resources and tasks. ii) It consists both computational and communication intensive tasks. iii) It considered hard, firm and soft tasks iv) It computes reputation of nodes during scheduling of initial set of grid nodes. v) It is computationally efficient.

The rest of the paper is organized as follows. In section 2, we outline the grid model used in this work. In Section 3, we review the existing distributed grid scheduling methods. Section 4 describes the proposed scheduling algorithm. Our experimental results are presented in section 5. Finally we conclude in section 6.

2. GRID MODEL

We consider the grid model as shown in Fig.1, for our investigation. The grid model consists of geographically distributed sites which are interconnected through WAN. At each site, there is a Grid Resource (GR) consisting of several machines of different processing capabilities and a grid user have many tasks to be scheduled by the grid scheduler. The communication within the site (intra-site) is 10Mbps as well as the communication across the sites (inter-site) is also 10Mbps.

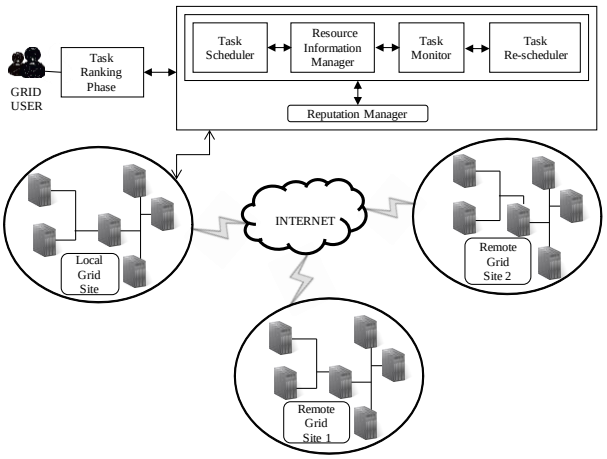

Fig 1: Grid Model

3. REVIEW OF DISTRIBUTED GRID SCHEDULING METHODS

3.1 GDS

GDS [13] consisting of three phases, which consists of multi attribute ranking phase, a shuffling phase and a dispatch phase. In the ranking phase the task are initially sorted the decreasing order of priority, the decreasing order of CCR and lastly by increasing order of deadline. In the second phase the shuffle procedure is used to assign each task to a specific resource and finally the dispatch phase unscheduled computational tasks are assigned to remote sites for a suitable resource matching.

The GDS computes CSSR as number of mission-critical tasks meeting deadlines to total number of mission-critical tasks and also computes overall successful schedule ratio as number of tasks meeting deadline to the total number of tasks.

GDS assumes resources are available all the time i.e. no resource failure, although this is unrealistic for most of the Grids.

3.2 RDGS

RDGS is an improved version of GDS, which considers the resource failure while scheduling of task. RDGS consists of three phases namely ranking phase, Scheduler phase and Re-scheduler. In the ranking phase the tasks are sorted by decreasing priority and then by decreasing CCR value and then by increasing order of deadline. In the second phase scheduler assign a task to a specific resource and in the last phase, the re-scheduler submits the tasks which are failed due to resource failure to other resource by keeping tasks in a queue. The following are the design goals of RDGS [15]:

- RDGS-MT assigns priorities as high, normal, and low to the tasks which correspond to hard, firm and soft tasks.
- RDGS-MT is based on Communication to Computing Ratio (CCR), which is used to decide local or remote site for task scheduling.
- RDGS-MT maximizes the total number of tasks completing execution and meeting their deadlines.
- RDGS-MT exploits reliability factor with respect to failure of nodes.
- RDGS-MT makes use of re-scheduling concept.

In RDGS, Overall successful schedule ratio is computed as the ratio of the number of tasks meeting deadline plus number of tasks executed on a reliable resource and the total number of tasks. The computed critical successful schedule ratio is computed as number of mission-critical tasks meeting the deadline plus number of mission-critical tasks executed on a reliable resource and the number of mission-critical tasks.

RDGS indeed considered the failure of resource, by rescheduling the tasks to another resource but the real time failure of resource is not monitored. Thus it effects overall successful schedule ratio as well as critical successful ratio.

4. REPUTATION AWARE RELIABLE DISTRIBUTED GRID SCHEDULER FOR MIXED TASKS (RRDGS)

We consider three kinds of tasks: hard, firm and soft. The proposed method uses such a task taxonomy which considers the consequence of missing deadlines and the importance of task property. A hard task cannot tolerate any deadline miss, since a single job that finishes after its deadline could disturb the entire system. A soft task can tolerate jobs that finish after their deadlines, whereas a firm task can tolerate only some job failure. Typically, a firm job should either finish before its deadline or not execute at all. In other words a soft job that misses its deadline can still do some useful work, while a firm job that misses its deadline is useless, though it does not jeopardize the system [15].

RRDGS consists of six phases: task ranking phase, resource manager, task scheduler, task monitor, reputation manager and task re-scheduler. **The task ranking phase** consists of two sub-phases, in the first sub-phase tasks are sorted in decreasing order of priority, then by decreasing order of CCR and then by increasing order of deadline. In the second sub-phase tasks queue are divided into 'n' sub-queues. **The resource manager** works as an agent for the existing computing resources in the system. A computing resource R = $\{r_1, r_2 \ldots r_m\}$ can be local or remote grid service provider. Each resource r_i is associated with two values such as computing speed of resource and failure frequency of the resource.

The task ranking phase initially will submit n^{th} sub-queue consists of soft tasks to the **task scheduler** for execution and retain a list of failed nodes with tasks and then it submits first sub-queue consist of mission critical tasks based on the feedback obtained from previous sub-queue. The same process will repeated for subsequent sub-queues. **The task monitor** check the status of each task whether it is successful or not and further un-successful tasks are maintained in a queue 'U' for re-scheduling. **The reputation manager** obtains feedback from each task sub-queue, which can be used for next sub-queue. **The re-scheduler** schedules the tasks in the queue 'U' to other available resources.

To further improve successful schedule ratio, we propose RRDGS where the scheduler considers resource failure by monitoring both task and resource entities. The following are the design goals of RRDGS:

- To maximize the reliability of scheduler i.e. the probability that all tasks complete successfully.
- To minimize the resource failure factor using reputation.
- To minimize the task failure by re-scheduling
- To minimize the execution time of tasks.
- To minimize the waiting time of tasks in queue.

The following notation is used in this paper:

T_i : i^{th} task
Pi : Priority of the task t_i
d_i : Deadline of task t_i
CCR_i : Communication to computation of the task t_i
Q : Task Queue
Q_i : i^{th} subqueue
U : Queue of tasks assigned to a failed node
S_i : i^{th} site with a number of machines
r_i : A resource in the system
R_{ri} : The instruction execution time of resource r_i
n_j : Number of machines within a site s_j
m_j : Number of neighboring sites for a grid site s_j
N= $\{N_1, N_2 \ldots\ldots N_n\}$: Available grid node list
L = List of failed nodes
T_{ij}: i^{th} task in subqueue, Q_j
TT_{ijk} : Transmission time of i^{th} task from site s_j to s_k
CC_{jk}: Computing capacity of $machine_k$ at site s_j

Each site contains a number of machines. The average computing capacity of a site s_j is defined as:

$$\overline{cc}_j = \sum_{k=1}^{n_j} cc_{jk} / n_i$$

The average transmission time of task T_j, from a site S_j to all its neighbors:

$$\overline{TT}_j = \sum_{k=1}^{m_j} TT_{ijk} / m_j$$

A task is composed of execution code, input data, output data, priority and CCR. Tasks are assigned high, normal, low priorities which correspond to hard, firm and soft respectively. The ratio of communication time to computational time (CCR_i) of a task T_i is defined as follows [13].

$$\mathrm{CCR}_i = \frac{\overline{TT}_j}{(instruction\ size\ of\ T_i / \overline{CC}_j)} x \frac{output\ data}{input\ data}$$

The reliability based scheduling of an application is to maximize the reliability and minimize the makespan of the application. To maximize the reliability of scheduling S, we need to minimize its task failure rate and reduce number of reschedules.

Procedure RRDGS-MT (Q, N) // at a site

Begin

1. Sort the Task Queue Q
   i) In descending order of priority, and
   ii) In increasing order of deadline.
2. Divide the task queue, Q into 'n' logical sub queues Q_1, Q_2…… Q_n such that low priority tasks occupy rear side queues.
3. Call RRDGS (Q_n, N, L)
4. For each Task subqueue, Q_i; i=1,2… n-1: Call RRDGS (Q_i, N, L)

End

Procedure RRDGS (Q_i, N, L)
Input: Subqueue, Q_i
Begin
1. For each task T_i in Q_i;
   Begin
   CCR_i = Compute–CCR (T_i);
   Begin
   If (CCR_i < 1)
   Begin
   1.1 Assign T_i to remote site, s_i
   1.2 Call RRDGS–Execute (T_i, S_i, N,L) for execution of T_i
   End
   Else
   Begin
   1.3 Assign T_i to local site, s_j
   1.4 Call RRDGS–Execute (T_i, S_j, N, L) for execution of T_i
   End
   End
   End

2. Compute Task Failure Percent.
3. Compute Mission Critical Task Failure Percent.
4. Update available grid node list N, by removing failed nodes at each site.

End

Procedure RRDGS–Execute (T_i, S_k, N, L)

Begin

1. Select a node, N_i randomly at grid site, S_k.
2. Check the status of the node N_i.
3. If (status of N_i is 'failed')
   Begin
   3.1 Insert T_i in queue U
   3.2 Insert N_i in list L
   3.3 Re-schedule T_i by calling once RRDGS–Execute (T_i, S_k, N, L)
   End
   Else
   Begin
   4.1 T_i is scheduled to node N_i
   End

End

5. EXPERIMENTAL RESULTS AND ANALYSIS

5.1 Experimental Setup

We used the following parameters in our experimental study: Task ID, Task length, Task file size, and Task output size, Priority, Deadline and Communication to computational Ratio (CCR).

We assumed the number grid resources as 10 percent of the mixed tasks under consideration in our analysis. We varied Resource failure rate percentage as 5, 8, 10, 16, 20 of grid resources under consideration and obtained results on increasing task size of {24, 48, 74, 100, and 125} x 10^5 MI. We computed Average Overall Successful Schedule Percentage (OSSP) using number of tasks successfully scheduled and total number of tasks and also Average Critical Successful Schedule Percentage (CSSP) using number of mission critical tasks successfully scheduled and total number of mission critical tasks.

We used GridSim [20] simulator for simulating Grid environment and the experimental results are shown in Figs. (2) - (7).

5.2 Experimental Results

5.2.1 Experiment 1: Computing OSSP on increasing task size of {24, 48, 74, 100, and 125} x 10^5MI by varying resource failure rate.

We compared RRDGS with RDGS and GDS with respect to the Average overall Successful Schedule percentage on increasing task size of {24, 48, 74, 100, and 125} x 10^5 MI by varying resource failure rate i.e. 5%, 8%, 10%, 16% and 20%, it is observed that RRDGS performed better in terms overall successful schedule percent than the RDGS by 2% at minimum task size and 4% at maximum Task size, Where as

GDS shows 12% lesser OSSR at minimum task size and 14% lesser OSSR at maximum task size.

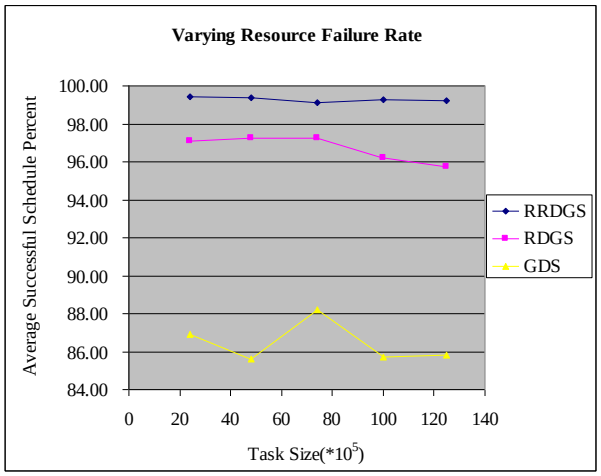

Fig 2: Average Successful Schedule Percent of GDS, RDGS and RRDGS on increasing task size

5.2.2 Experiment 2: Computing Average Mission Critical Task failure percent on increasing Number of Mission Critical Tasks by Varying Resource Failure Rate.

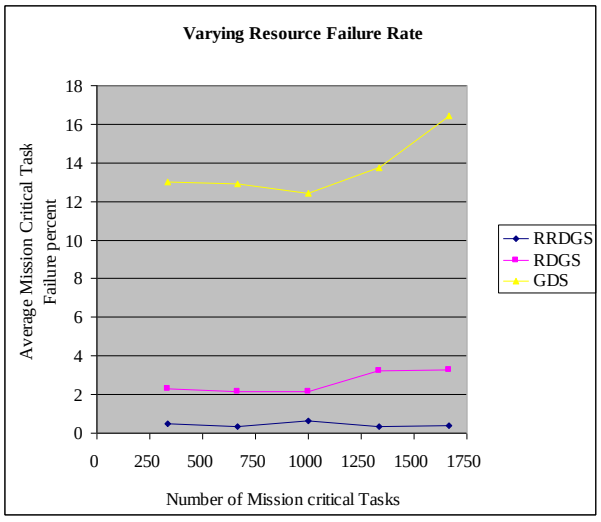

Fig 3: Average Mission critical task failure percent of GDS, RDGS, & RRDGS percentage on increasing number of mission critical task

We compared Average Mission Critical Tasks failure percent on increasing number of mission tasks on increasing Number of Mission Critical Tasks by Varying Resource Failure Rate i.e. 5%, 8%, 10%, 16% and 20%, it observed that the failure of mission critical tasks of RRDGS is less than 1% of mission critical tasks, where as failure of mission critical tasks of RDGS is 2 to 3% and GDS is 13% to17% on increasing number of mission critical tasks.

5.2.3 Experiment 3: Computing the Computational time/ Makespan of the schedulers by varying task size of {24, 48, 74, 100, and 125} x 10^5MI.

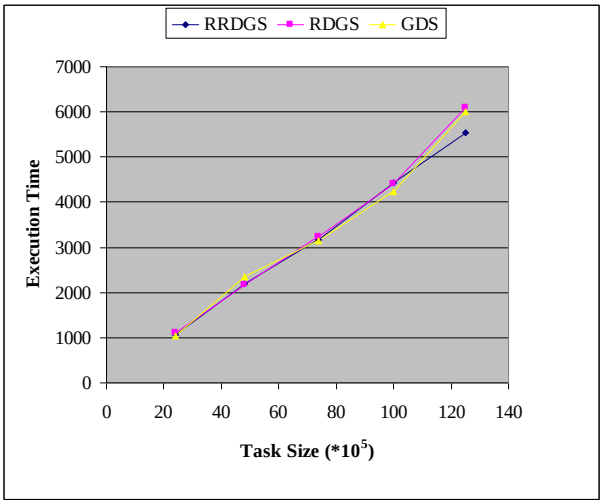

Fig 4: Computational time of GDS, RDGS & RRDGS by varying task size

We Computed the Computational time/ Makespan of the schedulers by varying task size of {24, 48, 74, 100, and 125} x 10^5MI , it is observed that at low and medium task size the execution time remains constant for the three schedulers, where as at high task size the execution time of RRDGS is lesser by 6% to 9% with respect to RDGS and GDS Schedulers.

5.2.4 Experiment 4: Computing task Waiting Time of tasks in reschedule Queue by varying number of tasks.

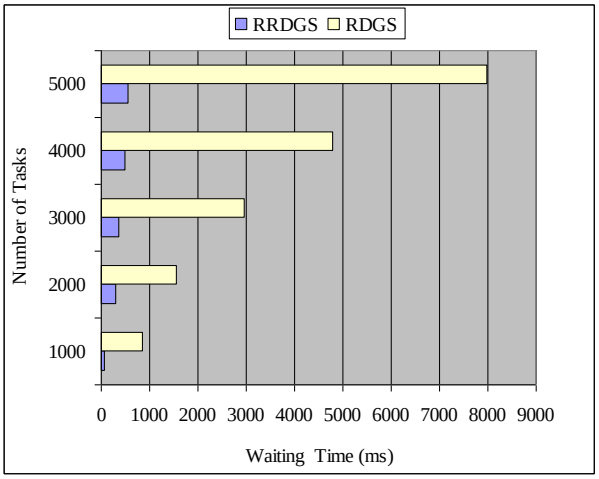

Fig 5: Computing Task waiting time of RDGS & RRDGS by varying number of tasks

We Computed task Waiting Time in rescheduled Queue by varying number of tasks. It is observed that the waiting time of tasks w.r.t. RRDGS in reschedule queue 'U' is minimal which varies from 60 to 540 milliseconds where as it varies from 918 to 8400 milliseconds for small (1000 tasks) and large number (5000 tasks) of tasks w.r.t. RDGS respectively.

5.2.5 Experiment 5: Impact of Rescheduling.

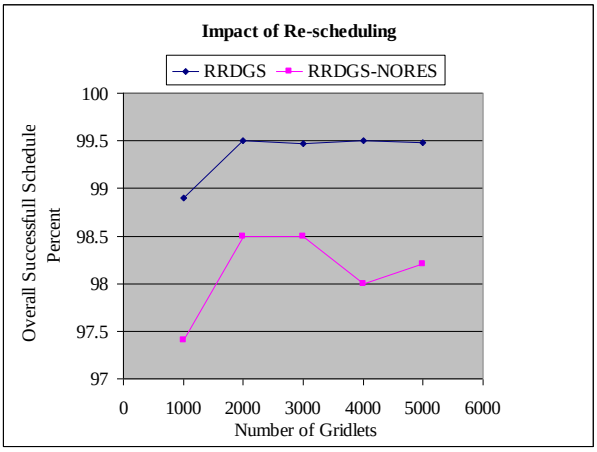

Fig 6: Overall Successful Schedule Percent of RRDGS with and without Rescheduling

In this experiment, we examine the usage of rescheduling for task monitoring component of RRDGS. To do so we use RRDGS-NORES, a scheduler, obtained upon removing task monitoring component. From above fig (6), it is observed that overall successful schedule percent of RRDGS is higher by 1% to 1.5% with respect to RRDGS-NORES on increasing number of tasks.

5.2.6 Experiment 6: Impact of Communication to Computational (CCR).

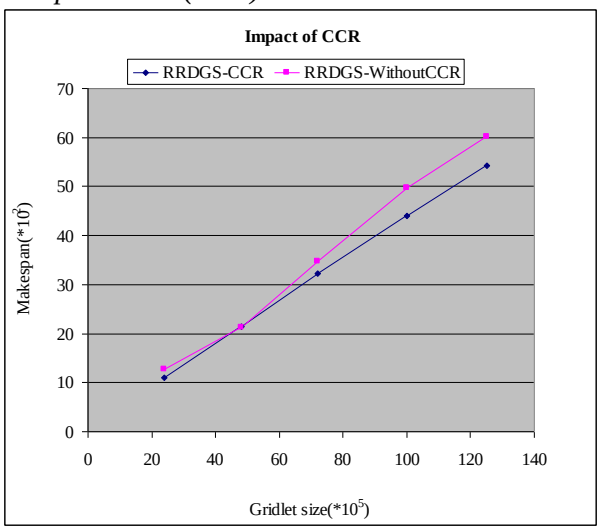

Fig 7: Makespan of RRDGS with and without CCR

In this experiment, in order to see the advantages of task ranking by CCR-type, we used another algorithm named RRDGS-NOCCR, which is same as the RRDGS with out CCR, sorts the tasks by priority and deadline, but nor CCR-type. We observe that there is improvement in execution time of 7% for high task size, 3% for medium size tasks and 1.5 % for low task size.

6. CONCLUSION

In this paper, we propose a novel reputation based distributed grid scheduler for independent mixed tasks in order to increase successful schedule percent, to reduce waiting time of task in the queue and to improve overall computational time. The proposed scheduler minimizes failure percent of mission critical tasks i.e. less than 1% and also incorporates task monitoring component to further improve successful schedule percent (i.e. 1.5). In future we extend this work by considering more topics including resource discovery algorithms, advance resource reservation and migration of task on execution.

7. REFERENCES

[1] Fatos Xhafa, Ajith Abraham, Computational models and heuristic methods for Grid scheduling problems, Future Generation Computer Systems, August 2009.

[2] Barry Wilkinson, Grid Computing Techniques and Applications, CRC Press, Taylor & Francis Group, A CHAPMAN &HALL BOOK , 2011.

[3] Fangpeng Dong, Selim G.Akl, Scheduling algorithms for grid computing : state of the Art and open Problems, A Technical Report No. 2006-504, 2006.

[4] Maozhen Li, Mark Baker, The Grid Core Technologies, A John Wiley & Sons, Inc.,, 2005.

[5] Wikipedia (Visited Feb 2009) Reliability, [online], http://en.wikipedia.org/wiki/Reliability.

[6] EunJoung Byun, Sung.Jin Choi, MaengSoon Baik, JoonMin Gil, Chan Yeol Park, Chongsun Hwang, Markov job scheduler based on availability in desktop grid computing environment, Future Generation Computer Systems 23 (2007) 616-622.

[7] Fiaz Gul Khan, Kalim Qureshi, Babar Nazir, Performance evaluation of fault tolerance techniques in grid computing system, Computers and Electrical Engineering, May, 2010.

[8] Mohammed Amoon, A fault-tolerant scheduling system for computational girds, Computers and Electrical Engineering, December 2011.

[9] Suchang Guo, Grid Service Reliability Modeling and Optimal task scheduling considering fault recovery, IEEE Transactions on reliability VOL. 60, No.1, March, 2011.

[10] Young Choon Lee, Albert Y. Zomaya, Rescheduling for reliable job completion with the support of clouds, Future Generation Computer Systems, March, 2010.

[11] Mustafizur Rahman, Rajiv Ranjan, Rajkumar Buyya, Cooperative and decentralized workflow scheduling in global girds, Future Generation Computer Systems, July, 2009.

[12] Katia Leal, Edurado Huedo, Ignacio M. Llorente, A decentralized model for scheduling independent tasks in Federated Grids, Future Generation computer systems, March, 2009.

[13] Cong Liu, Sanjeev Baskiyar, A general distributed scalable grid scheduler for independent tasks, J. Parallel Distrib. Comput. 69(2009) 307-314.

[14] Kovvur Ram Mohan Rao, S Ramachandram, Kadappa VijayaKumar and A Govardhan , A Reliable Distributed Grid Scheduler for Independent Tasks, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.

[15] Ram Mohan Rao Kovvur, S. Ramachandram, Vijayakumar Kadappa, A. Govardhan, A Reliable

Distributed Grid Scheduler for Mixed Tasks, PDCTA 2011, CCIS 203, pp. 213 -233, 2011.

[16] A. Josang, R. Ismail and C.Boyd. A survey of trust and reputation systems for online service provision. Decision support systems, 43 (2):618-644, March 2007.

[17] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In HICSS'00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6, Page 6007, Washington, DC, USA, 2000. IEEE Computer Society.

[18] Gheorghe Cosmin Silaghi, Alvaro E. Arenas, Luis Moura Silva, Reputation based trust management systems and their applicability to grids. Core GRID TR -0064, Feb 23, 2007.

[19] Xiaofeng Wang,, Chee Shin Yeo, Rajkumar Buyya, Jinshu Su, Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm, Future Generation Computer Systems, Volume 27,Issue 8, October 2011, Pages 1124-1134.

[20] Buyya R K, Murshed M, Anthony S, Marcos D de A, Agustin C, GridSim Tool kit 4.1: A Grid simulation toolkit for resource modeling and application scheduling for parallel and distributed computing (2007).