International Journal of Computer Applications (0975 – 8887)
Volume 44– No.23, April 2012

Robustness of Heuristic Resource Allocation Techniques in Grid Computing System

Sampa Sahoo
Department of CSE
PCE Rourkela
Rourkela, India

Bibhudatta Sahoo
Department of CSE
NIT Rourkela
Rourkela, India

Ashish Chandak
Department of CSE
NIT Rourkela
Rourkela, India

ABSTRACT

Grid computing system consists of machines with varied computational capabilities. These systems assist in the computing of large amounts of complicated tasks in scientific and engineering areas. It may operate in an environment where system performance features degrade due to unpredictable changes, inaccuracies in the estimation of task execution times etc. These systems need robustness. The robustness guarantees limited degradation in system performance. The following research is based on the requirement of robustness for resource allocation in grid computing environment. Four heuristic techniques for resource allocation are used to compare the robustness.

General Term

Grid Computing System

Keywords

Grid computing system, robustness, makespan heuristic resource allocation.

1. INTRODUCTION

Grid computing consists of machine sets with varying computing capabilities. It solves problems by allocating idle computing resources across geographically distributed area. The key goal of grid computing is to design a system that can provide improved efficiency and a platform for proper utilization of all computing resources within an enterprise or extended enterprise to meet end user demands [3, 8]. Grid is a decentralized heterogeneous system in which resources belong to multiple organizations. It does not enforce absolute control over these resources. From user's perspective, grid computing is a collaborative problem-solving environment in which one or more user jobs can be submitted without knowing where the resources are or whom it is allocated to. A grid computing system must guarantee the quality of service of a job's execution. It utilizes the network and combines idle resources scattered in every region for distributed applications [4]. A grid environment aggregates the rich computing resources from every part of the world to form a powerful computing capability. Traditional parallel scheduling problem deals with scheduling the subtasks of an application to the parallel machines in order to reduce the turnaround time [6]. In large grid computing system it is unwieldy for an individual to select the resources manually. So resource management and scheduling of tasks into machines are required for better performance. Grid computing systems should be able to assign the tasks of different users to the different avail resources efficiently and utilize the resources of unused devices (known as load balancing/job scheduling/resource allocation). Purpose of resource allocation is to improve the performance of the grid computing system through an appropriate distribution of the user's application tasks [2].

The rest of the paper is organized as follows. Section 2 describes system model and defines the resource allocation problem. Section 3 provides the work dealing with robustness and some robustness metrics. Section 4 presents some experiments and their results that highlight the usefulness of the robustness metric. Related work is given in section 5. Section 6 concludes the paper.

2. GRID-COMPUTING SYSTEM MODEL AND RESOURCE ALLOCATION

Grid is a system having a number of independent sites as shown in Fig. 1 and task execution in grid system is shown in Fig.2. A site may have either a single computing node or a number of computing nodes connected in a distributed manner. Resources in a site are not exclusively dedicated for grid usage. Sites can freely participate in grid computing by offering resources. We represent a grid as two tuple G = <S, TM> where S is the set of sites and TM is the set of tasks. We further represent the set S as S = { $S_1^{N_1}$, $S_2^{N_2}$,……… $S_i^{N_i}$,……, $S_n^{N_n}$ } where $S_i^{N_i}$ is the ith site have N_i number of resources, TM = { Tj | j ε 1,2 …. ,M}, the set of tasks to be executed in the grid. The resources at site S_i can be of data, computational or I/O type. Each site S_i is associated with few attribute. They are status St_i of the site (whether working or not working) and maximum capacity Cap_i of the site. A site S_i can be represented in three tuple S_i = < R_i, St_i, Cap_i >. The resource R_i at site S_i can be represented in three tuple R_i = <I/O_i, C_i, D_i> where

I/O_i = Set of resources of I/O type,

Ci = Set of resources of computational type and

Di = Set of resources of data type.

The QoS for I/O resources is characterized by speed and latency. The QoS for computational resources is characterized by computational speed and load. The QoS for data resources is characterized by space and disk bandwidth. So, the total no of resources available at all sites at a point of time t_j for a task is:

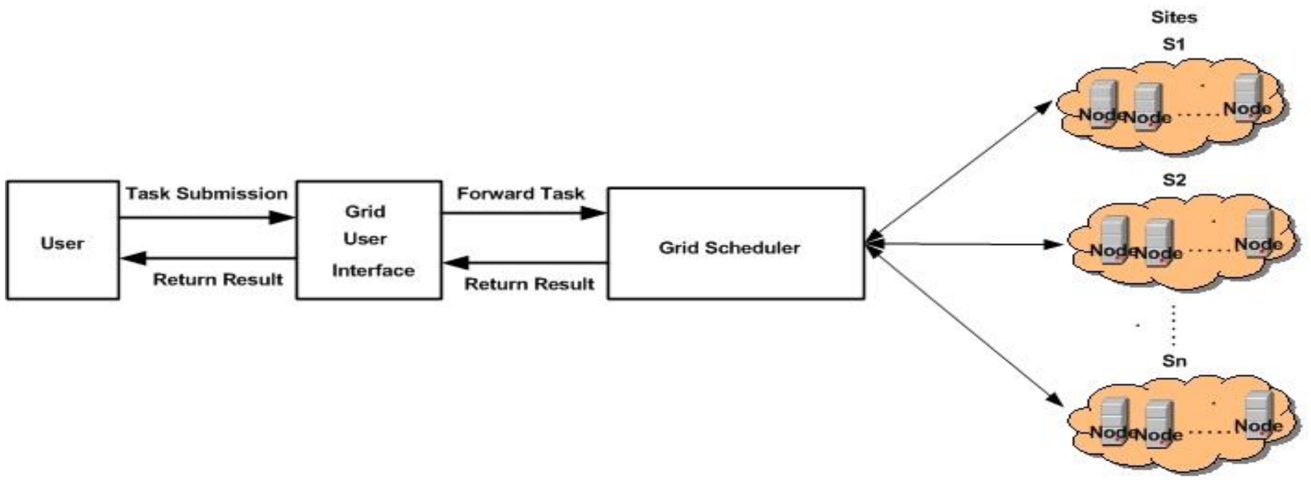

Fig 1: Grid system

R= {C_1, C_2, C_3......., C_n, I/O_1, I/O_2, I/O_3......., I/O_n, D_1, D_2, D_3......., D_n}

We represent the grid system as a M/M/s: N/FCFS queuing model as shown in Fig. 2 where: M - represents exponential inter arrival times between tasks, M - represents exponential execution time of tasks, s - represents number of computing sites, N - represents capacity of system i.e maximum task allowed in the system (this includes executing task plus waiting task), FCFS – represents First Come First Serve queue discipline.

Let λ_i be the rate of arrival of task from each grid user i at the grid scheduler. Assuming that there are j numbers of grid user, the total rate λ at which task arrive at the grid scheduler $\lambda = \sum_{i=1}^{j} \lambda_i$ Let μ_i be the rate at which a task is served at each site i. We assume that the service rate is independent and identically distributed. The combined service rate of all sites in a grid is $\mu = \sum_{i=1}^{n} \mu_i$.

Our queuing model is characterized by following parameters:-

n - Number of tasks in the system.

λ - Arrival rate of tasks.

μ - Service rate of tasks.

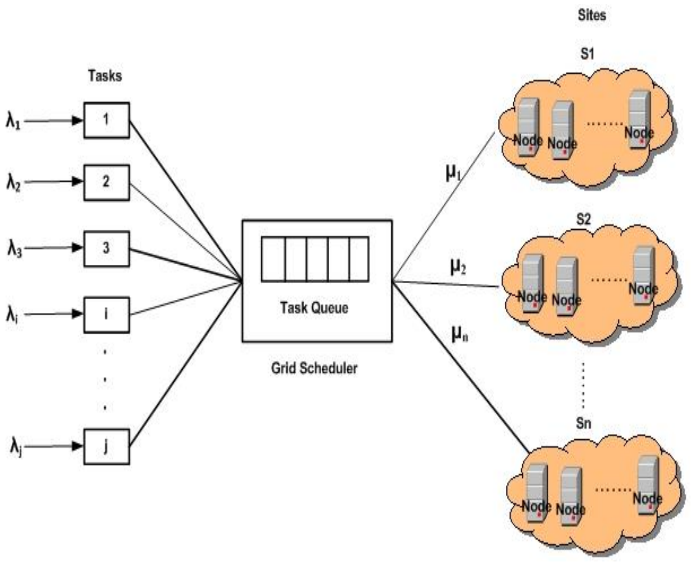

Fig 2: Grid system task model

The process of assigning each task to a machine and scheduling the execution of the tasks on each machine is known as *resource allocation/mapping/resource management* [3]. The goal of resource allocation is to achieve high system throughput. It also matches the application needs with available computing resources satisfying the required QoS [6]. A resource allocation is defined to be robust if system degradation is limited in the presence of unpredictable circumstances such as machine failure, higher than expected system load etc [8]. The degree of robustness is the maximum amount of collective uncertainty in perturbed system parameters. This should guarantee user-specified level of system performance. In robust system actual makespan (i.e completion time for an entire set of tasks) under the perturbed conditions does not exceed the required time constraint [3]. Following simple heuristics are used for resource allocation.

FCFS: FCFS (First come First serve) means first come first serve. As the name specifies the scheduler executes the jobs in the order of their submission i.e. job submitted earlier will be executed earlier.

Random: Tasks are selected randomly among all tasks that are submitted but not yet started for execution and this schedule is non-deterministic.

Min-min: The Min-min heuristic requires two steps. In first step, machine with minimum completion time is selected for each task. Second step, from all tasks, task with minimum completion time for execution is sent for execution.

Max-min: The Max-min heuristic method's first step is same as Min-min's but sends the task with maximum completion time for execution. This strategy is useful in a situation where completion time for tasks varies significantly.

3. JUSTIFYING THE USE OF ROBUSTNESS AS THE PERFORMANCE METRIC

Resource allocation that maximizes the robustness of a system in heterogeneous computing environment is an important research problem [5]. Heterogeneous computing (HC) systems utilize various resources with different capabilities to satisfy the requirements. It is very difficult to measure the performance of the system-using throughput as performance parameter since, it consists of different machines scattered here and there with different capabilities. These systems often operate in an environment where certain desired performance features degrade due to unpredictable circumstances, such as higher than expected work load or inaccuracies in the estimation of task and system parameters. .Thus when resources are allocated to tasks it is desirable to do this in a way that makes the system performance on these tasks robust against unpredictable changes [3]. The robustness of a computing system can be defined as the measure to which a system can perform correctly in the presence of parameter values different from those assumed [5]. This can be measured using different techniques described in section 2. Unpredictable changes may happen in any situations, which results system failure in grid computing environment. This may lead to degradation in system performance. So robustness can be used as performance metric to guarantee limited degradation in system performance in grid computing environment.

4. SIMULATION RESULTS

Grid scheduling is NP complete problem [3]. Various heuristics have been developed to solve this Grid scheduling problem. The four basic heuristic are economic heuristic [4, 5, 6], meta-heuristic [7, 8], population based heuristic [9, 3, 10, 11, 12], hybrid heuristic [13, 14, 15, 16, 17]. A grid scheduler acts as an interface between the user and distributed resources. It hides the complexities of the computational grid from the grid user [3]. This paper presents a brief discussion on various heuristics and their importance in grid scheduling.

We developed a simulation application in matlab to carry out the experiments. This is used to evaluate performance of grid scheduler and tasks are schedule using simple heuristic viz. Max-Min, Min-Min and FCFS heuristics. Each simulation experiment ends when 30-50 tasks executions are completed. The simulation model consists of four nodes each having different computing capability. The arrival of tasks is modeled as Poisson random process.

Different arrival rates are low (λ=1), moderate (λ=3), high (λ=5). For each mapping 20% delay in execution time was allowed i.e. actual completion time could be no more than 1.2 times of estimated value and the robustness metric was evaluated. Fig 3, Fig 4, Fig 5, and Fig 6 shows the robustness of a mapping against the number of tasks, with different arrival rate (λ). Fig 3, Fig 4, Fig 5 shows that max-min is better than fcfs, random, min-min even if we increase the arrival rate i.e. a system is more robust using max-min resource allocation strategy than the rest three. Fig 6 shows that for the same arrival rate if we increase the number of task robustness also increases, still max-min performs better than the rest three. The experiment also shows that in every case min-min is the worst strategy to implement for robust system.

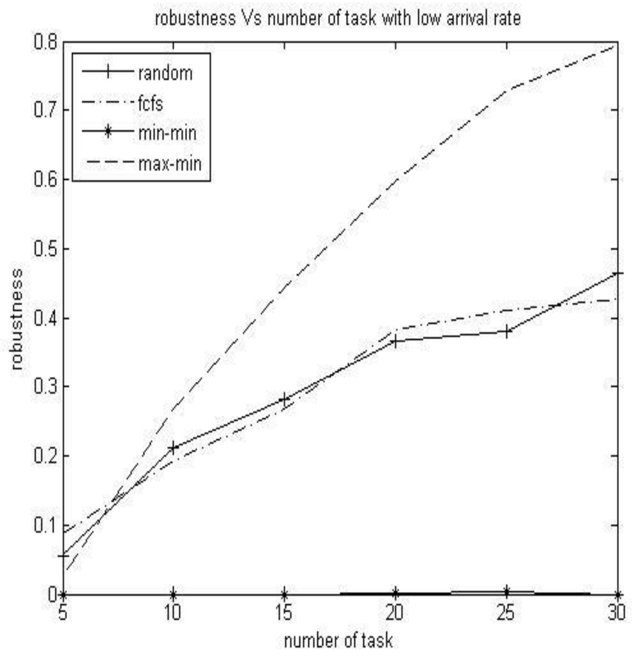

Fig 3: robustness V/s number of tasks with low arrival rate

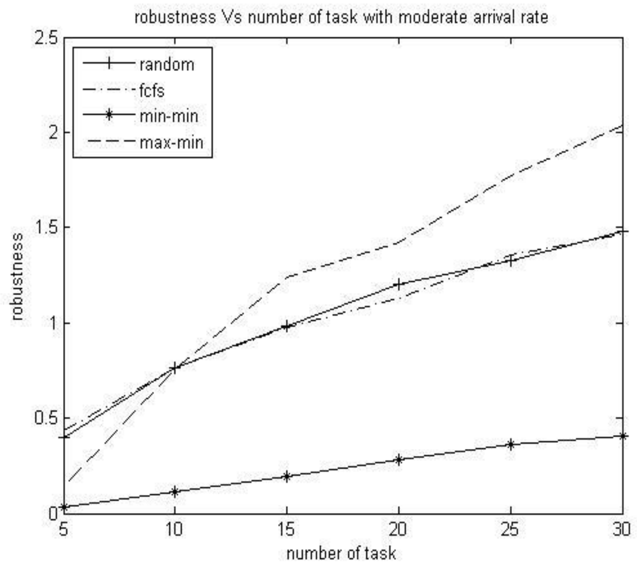

Fig 4: robustness V/s number of tasks with moderate arrival rate

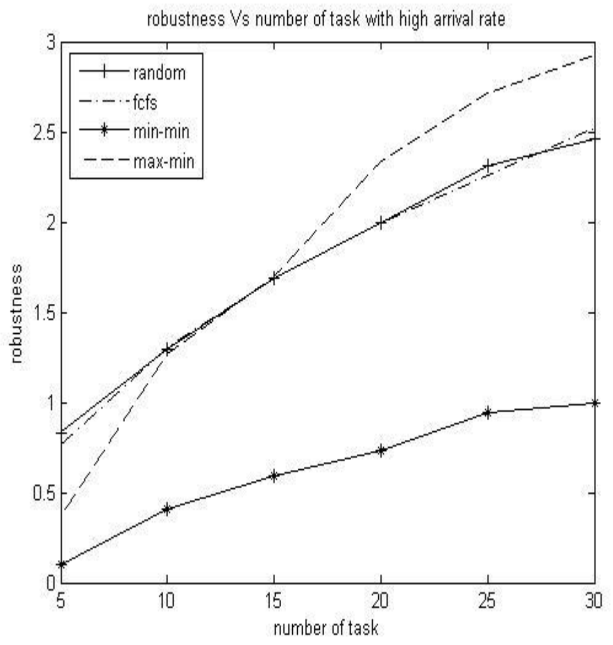

Fig 5: robustness V/s number of tasks with high arrival rate

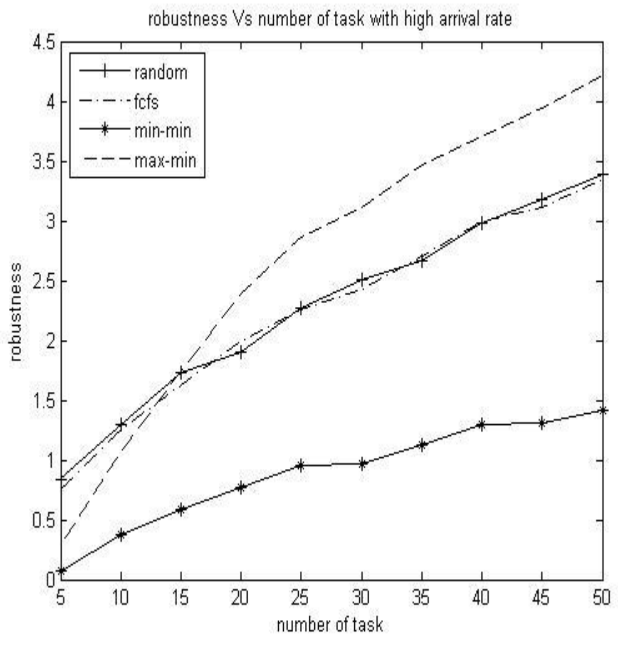

Fig 6: robustness V/s number of tasks with high arrival rate

5. RELATED WORK

Robustness is defined differently by different authors. Sotkov and Tanaev [7] have developed a metric for the robustness considering makespan against uncertainties in the estimated execution times of the application. It describes the effect of the uncertainties on the value of makespan and how the robustness metric could be used to find more robust resource allocation. Burns et al. [7, 14] use probabilistic guarantees for fault-tolerant real-time systems. In first step the authors have determined the maximum frequency of software or hardware faults that the system can tolerate without violating any hard real-time constraint. In second step the authors give value of the probability. The frequency determined in the first step is the maximum limit for a system failure. Davenport et al. [7, 10] uses slack-based techniques for producing robust resource allocations in a job-shop environment. Hence each task is provided with extra time defined as slack. Slack absorbs some level of uncertainty without reallocation.

In [3, 9] a single machine scheduling environment is considered where the processing times of individual jobs are uncertain. Using probabilistic information about processing time for each job a normal distribution is determined. The risk value is calculated by using the approx. distribution of flow time. One minus the risk of achieving substandard flow time performance gives the robustness of a given schedule. Slack is used as measure of robustness [3, 10] where slack is the extra time given to each job for completion so that some level of uncertainty can be tolerated without having to reallocate. In [3, 11] the authors use rescheduling policy in the event of breakdowns.

The researchers in [7, 13] examine the use of probability distribution of an applications execution time. They have described a scheduling policy, which tries to assign data to each processor so that all processors finish nearly about same time. Since execution times are different for different machines there may be machines, which are heavily loaded. Jobs are assigned to machines with smaller variability in performance. The research in [3, 15, 16] considers a scenario-based approach to represent the input data uncertainty to their robustness decision model.

There are various scheduling algorithms used to minimize the overall completion time of the tasks. These algorithms find the most suitable resources to be allocated to the tasks in a heterogeneous system. In [12, 17, 18, 19, 20, 21] max-min and min-min algorithms estimate the execution and completion times of each task on all the heterogeneous resources. The min-min algorithm selects the task with minimum completion time and assigns it to the, resource on which minimum execution time is achieved. One of the problems with this algorithm is that it assigns the smaller tasks to the resources with relatively higher computational power. Max-min is one of the variations of min-min algorithm where task with minimum completion time is assigned to resource on which maximum execution time is achieved. Max-min shows better performance than min-min algorithm if the number of shorter tasks is much more than longer ones. In case of max-min algorithm the small tasks may wait for larger ones to be executed. The researchers in [3, 12] used max-max, a variation of min-min algorithm. They argued that it performs better for static and dynamic mapping problem. QoS (Quality of services) guided min-min technique; a variation of conventional min-min is used in [17, 18].

QoS priority grouping scheduling [17, 20] considers deadline and acceptation rate of the tasks and makespan of the whole system as major factors for task scheduling. The authors have discussed that this algorithm performs better than min-min. In [22] FCFS is considered to be inefficient for many workloads as large number of jobs wait for execution. This situation may cause unnecessary idle time of some resources. The research in [22] showed that random policy is non-deterministic as jobs submitted earlier have a higher probability to be started before a given time.E.Elmroth et al. have proposed a user-oriented algorithm using advanced reservation and resource selection

[17, 23] for heterogeneous environment. This algorithm minimizes the total execution time of all the submitted tasks.

6. CONCLUSION

This paper presents a comparison of robustness of different simple heuristic such as FCFS, random, max-min and min-min. The results show that max-min performs better than FCFS, random and min-min i.e. max-min has highest capability to perform against the uncertainties. It also performs better than other algorithms even if we increase the number of task. The Max-min heuristic First, select a "best" (with minimum completion time) machine for each task. Second, from all tasks, send the one with maximum completion time for execution. Here larger tasks are executed first in best available machine which reduces the waiting time of these tasks. This leads to better load-balancing and efficient use of system. In FCFS execution of jobs are done according to their order of submission. Task execution will not be started until required resources are present which results stalling of others tasks in the submission queue So this method is inefficient as wide jobs waiting for execution can result in unnecessary idle time for some resources. Random heuristic randomly selects job for execution so it is highly non-deterministic. Min-min heuristic selects task with small completion time to schedule first on the best available machines. In this case task with longer completion time will have to wait for indefinite time which leads to load imbalance. So from the above discussion and experiment results we can conclude that max-min is better heuristic.

7. REFERENCES

[1] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal Supercomputer Applications, 2001

[2] Satish Penmatsa & Anthony T. Chronopoulos"Job allocation schemes in computational Grids Based on cost optimization". In Proc. 19th IEEE International parallel & distributed processing symposium (IPDPS'05), 2005.

[3] Prasanna Sugavanam,H.J.Seigel,Anthony A. Maciejewski,Mohana Oltikar,Ashish Mehta,Ron Pichel,Aaron Horiuchi,Vladimir Shestak,Mohammad AL-Qtaibi,Yogish Krishnamurthy,Syed Ali,Junxing Zhang,Mahir Aydin,Panho Lee,Kumara Guru,Michael Raskey,Alan Pippin "Robust static allocation of resources for independent tasks under makespan and dollar cost constraints "the journal of parallel and distributed computing (JPDC),2005.

[4] K.Q Yan,S.C. Wang,C.P. Chang and J.S. Lin "A hybrid load balancing policy underlying grid computing environment" in computer standards and interfaces,2006.

[5] Ashish M. Mehta, Jay Smith, H.J. Seigel, Anthony A.Maciejewski, Arun Jayaseelan, Bin Ye "Dynamic Resource Allocation heuristics that manage tradeoff between makespan and robust"in Springer Science-Business Media LLC 2007.

[6] Xiaoshan He, Xian-He Sun and Gregor Von Laszewski "A QoS Guided Scheduling Algorithm for Grid Computing" international workshop on grid & cooperative computing (GCC02) Pages 442-450, 2002.

[7] Zhimin Tian,Yang Yang,Zhengli Zhai "Modelling Robust Resource allocation for Grid computing" in proc.15th International Conference on Grid and Cooperation Computing(GCC'06),2006.

[8] Shoukat Ali,Anthony A. Maciejewski,Howard Jay Seigel, and Jong-Kook Kim "Definition of a Robustness Metric for Resource Allocation" in proc.17th International Parrallel and distributed processing symposium(IPDPS'03),2003.

[9] R.L. Daniels, J.E Carrilo "β-robust scheduling for single-machine systems with uncertain processing times ", IIE Trans.29 (11)(November-1997), 977-985.

[10] A.J. Davenport, C. Gefflot, J.C Beck "slack-based techniques for robust schedules", in: Sixth European conference on planning, September 1001,pp.7-18.

[11] V.J.Leon, S.D. Wu, R.h. Storer,"Robustness measures and robust scheduling for job shops, IIE Trans. 26(5) (September 1994) 32-34.

[12] T. D. Braun, H. J. Seigel, N. Beck, L.Boloni, R.F. Freund,D. Hensgen,M.Maheswaran,A.I. Reuther,J.P Robertson,M.D. Theys,B.Yao "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems" Journal of Parallel Distributed computing.61 (6) (june 2001)810-837.

[13] J.M. Schopf and F. Berman. "Stochastic scheduling "in proc of the 1999 ACM/IEEE conference on supercomputing, 1999.

[14] A. Burns, S. Punnekkat, B. Littlewood,and D. Wright,"Probabilistic Guarrantees for fault-Tolerant Real-time Systems" Technical Report,Design for validation (De Va) TR No. 44,Esprit Long Term Research Project no. 20072,Dept. of Computer Science,Univ. of Newcastle upon Tyne,U.K.,1997.

[15] P.Kouvelis, R.Daniels, G. Vairaktarakis "Robust scheduling of a two-machine flow shop with uncertain processing times" Iie Trans. 38 (5)(May 2000) 421-432.

[16] P. Kouvelis,G. Yu "Robust Discrete Optimization and its Applications" Kluwer Academic Publisher,Dordrecht,1997.

[17] Saeed Parsa and Reza Entezari-Maleki "RASA: A new task scheduling algorithm in grid environment" in World Applied sciences journal 7,152-160, 2009.

[18] He, X., X-He sun and G.V. Laszewski, 2003 "QoS guided Min-min heuristic for grid task scheduling" Journal of computer science and technology, 18:442-451.

[19] Maheswaran, M. Sh. Ali, H. Jay Siegel, D. Hensgen and R.F. Freund, 1999 "Dynamic Mapping of a class of independent Tasks onto Heterogeneous computing systems " journal of Parallel and Distributed Computing, 59:107-131.

[20] Dong, F., J. Luo,L. Gao and L. Ge,2006 "A grid task scheduling algorithm based on QoS priority grouping " in proc. Of the fifth international conference on grid and cooperative computing (GCC'06), IEEE.

[21] Etminani,K. and M. Naghibzadeh,2007 "A min-min Max-min Selective algorithm for grid task scheduling "The third IEEE/IFIP international conference on internet. Uzbekistan.

[22] Volker Hamscher, Uwe Schwiegelshohn, Achim streit, and Ramin Yashyapour "evaluation of job scheduling strategies for grid computing" in Grid-2000, volume: 1971, issue: 1, publisher: springer, pages: 191-202.

[23] Elmroth, E. and J. Tordsson, 2008 “grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions” journal of future generation computer systems, 24:585-593.

[24] Mohana Oltikar, Jeff Brateman , Joe White, Jon Martin, Keith Knapp, Anthony A. Maciejewski, H.J. Seigel “Robust resource allocation in weather data processing system” in proc international conference on parallel processing workshops(ICPPW’06),2006.

[25] Ashish Chandak, Bibhudatta Sahoo, and Ashok Kumar Turuk, "An Observation on Performance Analysis of Grid Scheduler", International Journal of Computer Science and Technology, Volume: 02, Issue: 04, Pages: 516-520, 2011, ISSN 2229-4333.

[26] Ashish Chandak, Bibhudatta Sahoo, and Ashok Kumar Turuk, "Performance Analysis of Adaptive Resource Clustering in Grid", International Journal of Computer Application, Volume: 29, Issue 09, Pages: 41-47, 2011, ISSN 0975 - 8887.

[27] Pratibha Zunjare, and Bibhudatta Sahoo, “Evaluating Robustness of Resource Allocation in Uniprocessor Real Time System", International Journal of Computer Application, Volume: 40, Issue 03, Pages: 13-18, 2011, ISSN 0975 - 8887.

[28] Bibhudatta Sahoo, S. Mohapatra, and S.K. Jena, “A Genetic Algorithm Based Dynamic Load Balancing Scheme for Heterogeneous Distributed Systems", in Proc. PDPTA, pp.499-505, 2008.