A Dynamic Task Scheduling for Divisible Load Applications using Simindex Strategy in Grid Environment

Sowntharya C[#1], Dr.V.Venkatesakumar[*2],

[#]*Department of Computer Science and Engineering, Anna University Regional Campus Coimbatore, TamilNadu, India*

[*]*Department of Computer Science, Anna University Regional Campus Coimbatore, TamilNadu, India*

Abstract**—** *Computational grid has the potentiality to solve large scale complex scientific applications. Improving the performance of computational grid in heterogeneous grid environment is a challenging task. The system certainly requires a fair distribution of workload among the available resources with optimum utilization of grid resources to schedule independent tasks. To address these requirements, a thorough knowledge on the system design and traditional scheduling algorithms is essential. The proposed simindex algorithm analyses the performance of the system in the simulation environment by distributing the sequential tasks through a middleware typically a global scheduler which selects best suitable computing resources from the clusters with lesser load. This is done by referring resource index entries of each cluster at global scheduler. The behaviour of numerous metrics such as load, distribution etc. for configuring the workload are studied. The efficiency of simindex with varying workload is compared with recent classical schedulers in most of the production grids today***.**

Keywords**—** *Simindex, Global scheduler, Resource index entries, Workload, Local Scheduler, Computational grid*

I. INTRODUCTION

Grid computing coordinates and interconnects all the heterogeneous and distributed grid resources namely CPU, disk, network, etc. This system has a guaranteed and secured infrastructure where very large scale and complex applications can be executed where the entire computational power of grid can be utilized. Since the grid systems [1] are highly complex in nature, in order to hide the system complexity from the users, the grid computing environments requires a middleware or a broker or a global scheduler to be designed and developed to distribute the user jobs to the corresponding computing resources.

The ultimate aim of the middleware or the global scheduler is to provide a programming Application Programming Interface (API) and an execution model for each of the entire applications. The scheduling algorithm executed on the global scheduler takes the entire responsibility for allocating the user submitted jobs to the distributed computing resources within the clusters. Scheduling distributed applications enables one of the major key services for proving efficient grids. The work focuses on one type of grid scheduler termed as middleware or a resource broker [2] or a Meta scheduler. The global scheduler distributes all the client jobs on to the grid computing resources. For instance, the clusters comprises of a pool of computing resources such as processors to form a grid [3]. All the clusters with pool of computing resources consist of Local Scheduler (LS) or a batch scheduler which adopts its own policy for executing all the compartment of tasks which are assigned to it. The implementation of a LS commonly uses a First In First Out scheduling policy by storing the dispatched tasks into a queue and are then assigned to the matching computing resources available within the clusters. Few other scheduling strategies [4] are also used depending on the requirement.

To be highly competent, a GS requires knowing the duration of the user jobs submitted explicitly on all computing nodes of the computational resources. Realistically it is highly complicated to identify the exact duration of the submitted tasks before carrying out its execution and mostly the duration of the tasks submitted are highly dependent on the parameter considered and the data. This requires every user to execute their jobs once and to submit the duration or deadline obtained to the GS.

In one of the scheduling algorithms where the duration of the job submitted by the user is not known to the GS, the duration is represented by a random variable assigning a fixed mean. Resource brokering [5] is an imperative issue where it involves various middleware's, for instance EDG/EGEE, which allocates jobs using a meta-scheduler.

In this paper, an attempt is made to evaluate the performance of computational grid system by using Task Distribution Strategy [6], Random Task Distribution Strategy [7] and Simindex Strategy for allocating workload under grid environment [8] alike to EDG/EGEE. Computational grids became rigorous for demanding computational tasks. They are potential challenges for the grid systems in becoming highly

efficient and recursively usable systems but still improving the grid system efficiency appears to be theoretically and practically raised open questions. Improving the efficiency largely deals with a promising scheduling strategy with the presence of a global scheduler to schedule the tasks to all its computing nodes by identifying the lesser load computing resources. The purpose of this dispatch is to achieve optimal distribution of load among all the computing nodes and to compute the sojourn time or response time which is the represented as the elapsed time between the arrival time into the gird and their completion time by taking waiting time and service time of jobs into account. The quality of dispatching the tasks is measured by computing the response time or sojourn time of all the tasks assigned.

The objective is to propose a proficient and a forthright strategy for scheduling tasks in computational grids having a centralized global scheduler based model such as EGEE (Enabling Grids for E-sciencE) [9], Grid 5000 [10] to distribute the tasks to computational resources.

The proposed system focuses much on computational grid and not on data grid or service grid so that the transfer times and data localization is ignored thus taking only the processing time of tasks into account. The performances of the proposed system are proven in terms of simple and complex modes of computational grid by considering the robustness of the system. The robustness has a tendency to show that the RIE would also perform well in real-life grids. The performance and the comparison show that approximation of 20% gain over most of the classical scheduling strategies.

II. Related Works

A detailed study on the conventional scheduling algorithms helped to evaluate and identify the performance and drawbacks of those algorithms. This helped to incorporate and address few of the drawbacks identified which made the aspects of the proposed system more efficient. Extensive investigations are made on the existing grid scheduling algorithms and workload balancing schemes [3]. Quite a few number of algorithms for instance First Come First Serve (FCFS) [11], Backfilling [12], Decentralized Scheduling [13], Rank Based Scheduling [14], Fine Grained Job scheduling [15], Prioritized Round Robin Scheduling [16], Adaptive Task Scheduling [17], Multi-round Algorithm [18] etc. which helped to analyse the static workload balancing and dynamic workload balancing [19] policies. Few works have also addressed the issue of faulty resource identification and tolerance [20] [21] to achieve effective management of grid resources and workflow management [3].

It is clearly identified from the analysis that though the grid resources are well managed the problem of overloading the grid resources are identified only when a fault occurs. Since the user submitted jobs are immediately scheduled the status of the resources are not tracked before assigning to it. Hence as an outcome of the above study, a novel approach is proposed which addresses the above issue. Also the efficiency of the proposed strategy is compared with the performance of existing algorithms which are in use today.

A. Serial Task Distribution Strategy

The Divisible load applications [6] are characterized by the input. The load consists of large numbers of independent units. Those independent units and be further divided into numerous multiple smaller tasks which consists of unknown number of units. It is clearly evident that the time taken for processing one unit is very less and the time taken for processing the entire input is more. It is assumed that the load is continuously divisible and most of the applications fall under the divisible load category.

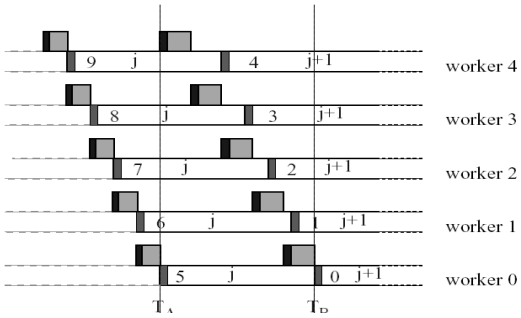

Fig. 1 Serial Task Distribution Model

The Serial Task Distribution Strategy helps the system to divide the total submitted workloads into numerous numbers of tasks and distributes those tasks to the workers over several rounds. The model above presented in fig.1 imposes the restriction that rounds are “uniform”, which means that the task sizes with in individual round are fixed. The system achieved better performance in terms of its execution time.

B. Random Task Scheduling Strategy

In Random Task Distribution Strategy, to be more efficient, introduces a Global Scheduler (GS) which divides the entire load into several numbers of smaller tasks and distributes the tasks to the cluster randomly [5] as shown in fig. 2. The system uses Poisson distribution P(λt) for distributing the tasks. Since the arrival rate of tasks ‘λt’ at time ‘t’ in not known and ‘μ’ represents the service time of task which is fixed. This system expects to know the duration of the submitted tasks on each node of the computing resources.

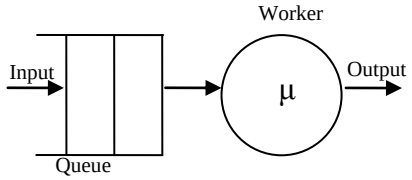

Fig. 1 Random Task Distribution Model

Indeed, this duration may depend on its parameters and data. It also requires the user to have run its job at least once and to have given the obtained duration to the GS. Since the task sizes are equally divided the duration of each job is with a random variable with a fixed mean. Hence this algorithm is therefore a randomized one.

III. Proposed Simindex Strategy

The proposed Simindex strategy functions in such a way that the Global Scheduler (GS) has a complete control over the clusters comprising large set of computing resources. The client submits the job to the GS, the proposed algorithm that is being executed in the Local Scheduler (LS) splits the entire job into numerous tasks and dispatches those tasks to the appropriate computing resource which evolves with lesser load. The idea is to deliberately not to make the job to wait in the GS except the time required for the GS to choose the computing resources which is approximately equal to zero. The tasks are ultimately be queued into lesser loaded cluster where the LS further implements the local scheduling strategy. The proposed system by evaluates the local performance. It also predicts the systems future behaviour. The execution of the system is based on the Resource Index Entries (RIE) computed using Markov decision processes. Since the RIE are based on local performances, their computations of RIE has less communication overhead and are done very fast and even computed online.

RIE is to be computed in each of the clusters based on which the Global Scheduler need to schedule the tasks. The RIE of each cluster should be updated frequently with a corresponding threshold whenever the jobs are submitted. The scheduling decision for a given job tries to minimize the time that this job will spend in the system, based on the current state of the system as well as predictions on its future. This is done using the mathematical framework of Markov Decision Processes. The proposed Simindex is to present a strategy, where the system acquires the current state of the system and as well as the statistical information of the system to gather predictions about the average future behaviour of grid using Resource Index Entries (RIE).

The overall goal of the system is to compare the computational time of our system with varying workloads and to show the efficiency of the work through simulation. Several tracks for improvements will be investigated in the future. One potential improvement is to re-compute RIE according to the current load. The simulations presented in this system use RIE with average load assumptions which is not true for many cases. Another potential improvement can be obtained by estimating the size of the jobs before scheduling them to the clusters.

A. Predictions

The focus of the work concentrates on the implementation of middleware strategies based on unpredictable models for allocation of jobs into grids and use predictions about the future behaviour of the grid to make smarter decisions. Systems may have high dynamicity which involves frequent failures, breakdowns, upgrades etc. and input traffic may tend to sudden bursts of big jobs. Also, each cluster within the grid has a complex behaviour which may change its states very fast and hence an efficient strategy which uses predictions to schedule tasks in computational grid environments is proposed.

B. Resource Index Entries (RIE)

A simple model of a computational grid has to be considered where clusters are seen as simple queues and traffic is a Markovian point process.

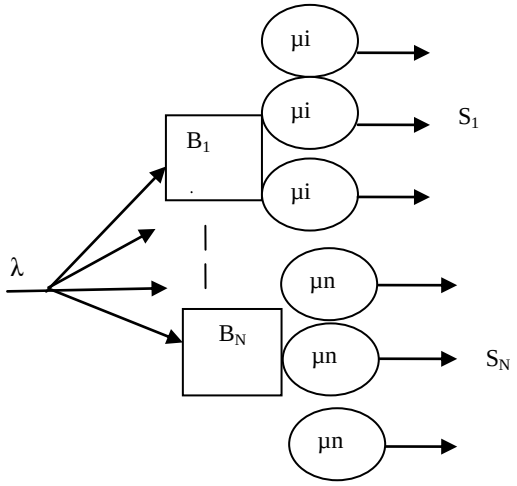

Fig. 2 A Queuing Model for Simindex Load Distribution

Fig. 3 shows simple model allows for a mathematical sound derivation of a scheduling policy based on Markov decision processes [22] where 'λ' represents the arrival rate of incoming jobs, 'B' represents the buffer size, 'μ' represents the service rate which is constant defined by a random variable, 'S' represents the execution time of tasks. Several simulations of the simple model show that the proposed strategy is very efficient and is also very robust to model variations.

C. Several Inputs

The computational grid is consists of five clusters. The first cluster is made of c1 = 97 identical CPUs. Each of them can execute 2.3 units of work per time-

unit on average (we denote this parameter by μ = 2.3). The second cluster is made of c2 = 79 CPUs with rate μ2 = 1.82, the other clusters have respective parameters c3 = 104, μ3 = 1.9, c4 = 105, μ4 = 1.5 and c5 = 113, μ5 = 2.1. The buffers in the clusters are all of size B = 1000. When 'r' is big (light loads) most policies lose around 25% with respect to RIE and have a similar behaviour. However, Simindex performs better for both light loads and heavy loads. When r is 1, the load is very high and it is more risky to make pertinent comments on them.

IV. Experimental Results and Comparison

The proposed Simindex algorithm is tested with varying workloads in the dynamic grid environment using SimGrid tool [23] and the performance of the system is compared with the existing systems performance. The system tested with varying loads in shown in the table 1 with various execution time of STDS, RTDS and Simindex scheduling below.

TABLE I
COMPARISON OF STDS, RTDS AND SIMINDEX FOR VARYING LOADS

No. of Tasks	Execution Time(Milliseconds)		
	STDS	**RTDS**	**Simindex**
200000	1166.62	990.47	740.471
500000	3045.21	2419.27	1850.57
800000	4878.59	3796.46	2961.21
1000000	6097.66	4578.07	3701

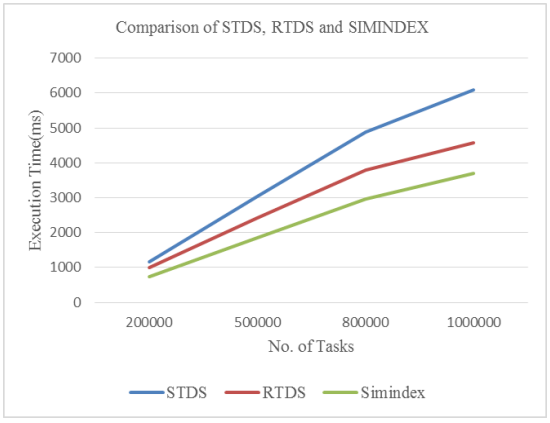

Fig. 4 Comparison of STDS, RTDS and SIMINDEX with varying loads

The comparisons of the proposed system with two other classical scheduling algorithms are shown in fig. 4. All the algorithms are tested with varying workloads and the proposed Simindex algorithm outperforms consistently with varying workloads and hence found to be more efficient.

V. Conclusions and Future Scope

The proposed Simindex algorithm for scheduling the tasks refers the RIE each time before scheduling the tasks to the computing resources. The RIE helps the system to find the lesser loaded clusters and thus helps to achieve better load balancing. The proposed system is compared with two other existing systems which are in use today and the proposed system achieves better results with varying loads and proves to be consistent.

In future, the system setup can include additional resource brokers to check the performance of the system for heavy loads. Resources fault identification and fault tolerance can also be incorporated to enhance the performance of overall performance of computational grid system.

References

[1] Rajkumar Buyya, Sri Kumar Venugopal, "A Gentle Introduction to Grid Computing and Technologies", Computer Society of India, CSI Communications, 2005.

[2] Emmanuel Medernach, "Workload analysis of a cluster in a grid environment", Proceedings of the 11th International conference on Job Scheduling Strategies for Parallel Processing, ACM Digital Library, Springer-Verlag Berlin, Heidelberg, Pages 36-61, 2005.

[3] Enis Afgan, "Role of the Resource Broker in the Grid", Proceeding ACM-SE, 42nd Annual Southeast Regional Conference, 2004, Pages 299-300

[4] Fangpeng Dongand Selim G. Akil, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", Queen's University, Kingston, Ontario, January 2006.

[5] V Berten, B Gaujal, "Brokering strategies in computational grids using stochastic prediction models", Parallel Computing 33 (4), Feb 2007, 238-249.

[6] Li. Y, Yang. Y and Zhu. R, "A Hybrid Load balancing Strategy of Sequential Tasks for Computational Grids,"IEEE. International Conference on Networking and Digital Society, DOI 10.1109/ICNDS.2009.

[7] Vandy Berten, Joël Goossens, Emmanuel Jeannot," On the distribution of sequential jobs in random brokering for heterogeneous computational grids" IEEE Transaction on Parallel and Distributed Systems, 17(2): 113-124, 2006.

[8] L. He, S.A. Jarvis, D.P. Spooner, X. Chen, and G.R. Nudd, "Hybrid Performance-Based Workload Management for Multi-clusters and Grids," IEEE Proc. Software, special issue on performance eng., vol. 151, no. 5, pp. 224-231, Oct. 2004.

[9] EGEE: Grids for E-sciencE. [Online]. Available: http://www.eu-egee.org.

[10] GridPP. [Online]. Available: http://www.gridpp.ac.uk. [10] Grid 5000, a large scale nationwide infrastructure for grid research. [Online]. Available: https://www.grid5000.fr/.

[11] Uwe Schwiegelshohm, RaminYahyapour, "Analysis of First-Come-First-Serve Parallel Job Scheduling", Proceedings of the 9th SIAM Symposium on Discrete Algorithms, 1998.

[12] Sofia K, Dimitriadou, Helen D. Karatza, "Job Scheduling in a Distributed System Using Backfilling with Inaccurate Runtime Computations", International Conference on Complex, Intelligent and Software Intensive Systems, 2010.

[13] R. Shah, B. Veeravalli, and M. Misra, "On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments," IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 12, pp. 1675–1686, 2007.

[14] Rajesh Kumar Bawa, Gaurav Sharma, "Ranking Based Resource Selection in Grid Environment", International Journal of Computing Science and Communication Technologies, Vol. 4, No. 1, July 2011. (ISSN 0974-3375).

[15] Yeqing Liao and Quan Liu, "Research on Fine-grained Job scheduling in Grid Computing" International Journal of Information Engineering and Electronic Business, 2009, 1, 9-16, October 2009.

[16] Sunita Bansal, Bhavik Kothari, Chittaranjan Hota, "Dynamic Task-Scheduling in Grid Computing using Prioritized Round

Robin Algorithm", International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.

[17] Babar Nazir, Mohd Fadzil Hassan, Halabi Hasbullah "Adaptive Task Scheduling Strategy for Economy Based Grid",International Symposiumon Computing, Communication, and Control (ISCCC 2009) Proc .of CSIT vol.1 (2011).

[18] Yang Yang, Henri Casanova "UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads", IEEE Transaction on Parallel and Distributed Systems, Vol. 16, No. 11, November 2005.

[19] SandipKumar Goyal, Manpreet Singh, "Adaptive and Dynamic Load Balancing Grid Using Ant Colony Optimization", International Journal of Engineering and Technology, Vol. 4, No. 4, 2012.

[20] Latchoumy.P, Sheik Abdul Khader. P "Survey on Fault tolerance in Grid Computing," International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.4, November 2011.

[21] Ritu Gargand Awadhesh KumarSingh "Fault Tolerance in Grid Computing: State of the Art and open issues," International Journal of Computer Science & Engineering Survey, Vol. 2, No 1, February 2011.

[22] Randolp Nelson, "Probability, Stochastic Process and Queuing Theory: The Mathematics of Computer Performance Modelling" Springer Science & Business Media, June 1995.

[23] Legrand. A, L. Marchal, H. Casanova, Scheduling Distributed Applications: The SimGrid Simulation Framework, in: 3rd International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Los Alamitos, CA, USA, 2003, p. 138.