IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 2, Issue 6 (July-Aug. 2012), PP 13-19
www.iosrjournals.org

Statistical Framework For Load Balancing In Grid Computing For Efficient Job Migration

[1]Ashish Chandra Srivastava, [2]A Kakoli Rao
[1,2]*Student: Dept of Computer Science &Engg. Galgotias College of Engg. & Technology Greter Noida, India*

Abstrac: *The proposed system has addressed issues that are imperative to Grid computing environments by introducing job migration algorithms. The proposed algorithms differ within the manner load balancing is disbursed and is shown to be cost effective in minimizing the response time on Grid environments. The algorithm is enhanced for large-scale systems to take into account the job migration value, resource heterogeneity, and network heterogeneity when load balancing is considered. The algorithm is applicable to small-scale systems, performs load balancing by estimating the expected end time of employment on individual processors on every job arrival to estimate system parameters like the job arrival rate, processing rate, and load on the processor and balance the load by migrating jobs to individual processors by considering job transfer value, resource heterogeneity, and network heterogeneity.*

Keywords*-component; Grid Computing, Load balancing, Inter arrival time, processing elements*

I. INTRODUCTION

Grid computing has emerging in recent years as a viable computing paradigm to solve data and compute intensive applications. Programming environments for building and executing applications must handle, among others, the complexity of the Grid's deployment, dynamicity, distribution, fault tolerance, heterogeneity, high performance, interoperability, and scalability. Given the complexity of these concerns, it is unreasonable to charge users for directly address them when programming their Grid applications. This has led to the development of Grid middleware delivering low-level functionalities handled in a domain independent way, related for example to security, low-level resource management, monitoring, data management, efficient communication, etc [1]. On top of such low-level middleware high-level programming environments can be built. Each programming environment provides a domain specific high-level programming model that simplifies and hides away the complexities of the Grid.

In general, any load-balancing algorithm consists of two basic policies—a transfer policy and a location policy. The transfer policy decides if there is a need to initiate load balancing across the system. By using workload information, it determines when a node becomes eligible to act as a sender (transfer a job to another node) or as a receiver (retrieve a job from another node). The location policy determines a suitably under loaded processor. In other words, it locates complementary nodes to/from which a node can send/receive workload to improve the overall system performance. In a centralized system, the load scheduling is done only by a single processor [2]. Such algorithms are bound to be less reliable than decentralized algorithms, where load scheduling is done by many, if not all, processors in the system. However, decentralized algorithms have the problem of communication overheads incurred by frequent information exchange between processors.

Load balancing involves assigning to each processor work proportional to its performance, thereby minimizing the response time of a job. However, there are a wide variety of issues that need to be considered for a heterogeneous Grid environment. For example, the capacities (in terms of processor speed) of the machines differ because of processor heterogeneity. Also, their usable capacities vary from moment to moment according to the load imposed upon them. Further, in Grid computing, as resources are distributed in multiple domains in the Internet, not only the computational nodes but also the underlying networks connecting them are heterogeneous [3]. The heterogeneity results in different capabilities for job processing and data access [4].

The project work design and propose a new algorithm that balances load by transferring a job on its arrival epoch rather than waiting for the next transfer instant. This is clearly a faster reaction to respond to higher arrival rates on smaller Grids. In the proposed algorithm, instead of estimating the expected finish time of a job at every estimation time period, it will be calculated on each arrival of a job to a processor. Here, estimating the finish time of a job is an aperiodic event, and job migration will now happen a periodically. Therefore, when the load is not distributed evenly across all processors, a job will be migrated to lightly loaded processors much faster in the proposed load balancing algorithm than in enhanced version of ELISA algorithm. In section 2 we give an overview of related work which identifies all the major research work being done in this area. Section 3 highlights about the assumptions considered for the proposed system. Proposed system is discussed in Section 4 followed by result and analysis in Section 5. Section 6 makes some concluding remarks.

II. RELATED WORK

Malarvizhi et.al [5] assumes a hierarchical structure of computational resources and clusters with different processing capacity and network bandwidth. Based on certain assumptions and neighborhood strategy they present a hierarchical load balancing algorithms at different levels of hierarchy.

P.Neelakantan [6] propose a diffusive load balancing algorithm which distributes a proportion of excessive workload of heavily loaded node to lightly loaded node by considering the nodes processing capabilities.

Hongzhang et.al [7] focus of this study is to explore the impact of data migration under a variety of demanding grid conditions. They evaluate our grid scheduling algorithms by simulating compute servers, various groupings of servers into sites, and inter-server networks, using real workloads obtained from leading supercomputing centers.

Satish and Anthony [8] describe two price-based dynamic job allocation schemes for computational grids are proposed hose objective is to minimize the execution cost for the grid users' jobs. Review two static job allocation schemes for conventional distributed systems and extend them to dynamic job allocation schemes for computational grids.

Jeremy K. Chen et.al [9] presents an efficient iterative load balancing algorithm for time and bandwidth allocation among access points (APs) and users subject to heterogeneous fairness and application requirements.

Giuseppe and Michael [10] present a Dynamic Load Balancing (DLB) policy for problems characterized by a highly irregular search tree, whereby no reliable workload prediction is available.

Lorenzo Muttoni et.al [11] presented the workload characterization of the applications running on a grid is fundamental to predict the expected performance of the system. They also pointed out that the statistical distributions used to model the computation and communication time heavily affect the response time.

Hans and Michael [12] consider the problem of mapping tasks to processor nodes at run-time in multi programmed multicomputer systems (i.e. message passing MIMD-systems).

Xiaobo et.al [13] presents some ongoing work and planned future work at the Cambridge eScience Centre. They describe two use-cases: a database of results in computational fluid dynamics (CFD) and a small computational Grid for aircraft engineering design.

Giuseppe et.al [14] build the system as an overlay network, in which the nodes hosting instances of each of many different types of services can self-organize as "virtual clusters", and efficiently load-balance incoming requests amongst themselves.

Constantino Lagoa et.al [15] addresses the problem of optimal decentralized traffic engineering when multiple paths are available for each call. They also propose a large family of decentralized sending rate control laws having the property that each of the members of this family "steers" the traffic allocation to an optimal operation point.

G.Kavitha and Sankaranarayanan [16] introduce a new method that provides a quantitative trust value, based on the past interactions and present environment characteristics. They propose a novel trust model to calculate the quantitative value of execution trust.

Sandeep et.al [17] present the performance analysis of various load balancing algorithms based on different parameters, considering two typical load balancing approaches static and dynamic. Their purpose of this paper is to help in design of new algorithms in future by studying the behavior of various existing algorithms.

Magdy and Cherine [18] present a decentralized algorithm for diffusion dynamic load balancing based on mobile agent paradigm. They introduce the architecture of three types of agents employed to meet the requirements of the proposed diffusion load-balancing algorithm

Malarvizhi et.al [19] a load distribution algorithm in the decentralized heterogeneous computing platform is proposed. A decentralized grid model, as a collection of clusters is also proposed. Don Abraham and VetriSelvi [16] have dealt in detail about formation of a grid overlay with dynamic VOs over a mobile ad hoc network using lightweight algorithms

S. Prakash and Vidyarthi [20] deals with the load balancing in computational grid with emphasis on the observation of load variation and load distribution among the nodes in the computational grid environment. They have been considered six cases to observe the load variation and load distribution.

Sajal K. Das [21] deals with decentralized load balancing in distributed-memory multicomputer in which processors are connected by a point-to-point network topology and communicate with one another via message passing. They propose three efficient dynamic load-balancing algorithms.

Han Zhao et.al [22] propose the DLBEM (Dynamic Load Balancing Based on EM Algorithm) approach, which uses EM algorithm to make accurate workload migration decisions with reduced number of communications.

Darin and Jon [23] present an analysis of the costs and benefits of load sharing of parallel jobs in the computational grid. They begin with a workload generation model that captures the essential properties of parallel jobs and use it as input to a grid simulation model. They also evaluate an effective scheduling heuristic for migrating a job within the grid.

Said Fathy [24] addresses the problem of load balancing and task migration in grid computing environments. We propose a fully decentralized two-level load balancing policy for computationally intensive tasks on a heterogeneous multi-cluster grid environment.

III. ASSUMPTIONS

The research work assumes that each processor has an infinite capacity buffer to store jobs waiting for execution. This assumption eliminates the possibility of dropping a job due to unavailability of buffer space. The jobs are assumed to arrive randomly at the processors, the inter-arrival time being exponentially distributed with average $1/\lambda_i$. The jobs are assumed to require service time that is exponentially distributed with mean $1/\mu_i$. Each processor is modeled as anM|M|1 Markov chain, with the number of jobs queued up for processing at each processor representing the state of the system. Job size is assumed to have a normal distribution with a given mean and variance. This job size includes both the program and data sizes. The major dependency of the project work is the availability of the java environment for building the project application.

IV. PROPOSED SYSTEM

The main aim of the project work is to design an architectural framework to propose decentralized, scalable, adaptive, and distributed algorithms for load balancing across resources for data-intensive computations on Grid environments using two job migration algorithms which is modified ELISA and Load Balancing Algorithm. In the proposed system, we present a complete formulation and results for large-scale, as well as for small-scale, Grid environments, including the contributions and addressing all the above mentioned issues. In all, through this study, we demonstrate the usefulness and effectiveness of the load estimation approach to devise adaptive and dynamic load-balancing strategies for small and large-scale computational Grid structures. The product is design in Java Swing

The current work exhibits two dynamic, adaptive, and decentralized load balancing algorithms for computational Grid environments that are shown to be applicable in balancing loads depending on the size of the underlying Grid infrastructure. Thus, for smaller size Grids, one of our algorithms, Load Balancing on Arrival (LBA), is shown to be effective, whereas for large-scale Grid systems, the Modified ELISA (MELISA) is shown to have better control in balancing the loads. One of the key strengths of our algorithm is in estimating the system parameters and in proactive job migration. For large-scale Grid environments, resources are geographically distributed, and the communication latency between them is also very large due to the WAN through which they are usually connected. Therefore, the job migration cost, based on the estimate of the traffic and loading conditions, becomes an imperative factor for load balancing. The proposed algorithms consider the job migration cost, which is primarily influenced by the available bandwidth between the sender and receiver nodes, when making a decision for load balancing. Further, Grid infrastructures are dynamic in nature in the sense of resource availability and, hence, a changing network topology. Resource heterogeneity and network heterogeneity also exists in the Grid environment. We have also considered these facts into account by generating a random topology with nodes of varying capacities and varying bandwidth across the links connecting them. The sensitive parameters such as the arrival and service rates, uneven load distribution scenarios, effects of status exchange information and estimation periods, capturing the migration limits, and utilization are not addressed. In this paper, we present a complete formulation and results for large-scale, as well as for small-scale, Grid environments, including the contributions of [14] and addressing all the above mentioned issues. In all, through this study, we demonstrate the usefulness and effectiveness of the load estimation approach to devise adaptive and dynamic load-balancing strategies for small and large-scale computational Grid structures. It was also seen that in grid-based supercomputing systems, the transfer delays are significantly high, and network heterogeneity also exists in terms of the varying available network bandwidth contributing to a large communication cost in case of ELISA. Hence the modified version algorithm is as below:

Algorithm: MELISA

Input: Inter-arrival Time, Jobs, status exchange period

Output: Job Migration

STEPS

1 Estimate Arrival rate (λ_i) and Service Rate (μ_i)

2 $\lambda_i(T_{n-1})=\alpha.\ \lambda(T_{n-2})+(1-\alpha).(A_i(T_s)/T_s)$

3 $\mu_i(Tn-1)=\beta.\ \mu_i(T_{n-2})+(1-\beta).(D_i(T_s)/T_s)$

//where α and β are arrival rate estimation factor, A_i is actual number of jobs arrival, Di is actual number of jobs departure, and T_s is status exchange period.

4 Estimate load on processor (L_i)

5 $=Q_i(T_{n-1})/\ \mu_i(T_{n-1})$

//Q_i is number of jobs waiting in queue.

6 Communicate the status as<estimate_load, arrival_rate, service_rate> to all processor

7 Initiate transmissions

8 Estimate load for each processor
9 Go line Initiate transmission
10 Estimate mean normalized load.
11 If (Proc_Load>Mean_Load)
12 If(Proc_Load<Mean_Norm_Load)
13 Include processor
14 Calculate load to be transferred
15 If $EFT^j_k < EFT^j_i$
//ERT: Estimated run time of Job
//EFT: Estimated finish time of Job
16 Initiate load migration

END

Algorithm: Load balancing on Arrival

Input: Inter-arrival Time, Jobs, status exchange period

Output: Job Migration

STEPS

1 Estimate Arrival rate (λ_i) and Service Rate (μ_i)
2 $\lambda_i(T_{n-1})=\alpha.\ \lambda(T_{n-2})+(1-\alpha).(A_i(T_s)/T_s)$
3 $\mu_i(Tn-1)=\beta.\ \mu_i(T_{n-2})+(1-\beta).(D_i(T_s)/T_s)$
//where α and β are arrival rate estimation factor, A_i is actual number of jobs arrival, Di is actual number of jobs departure, and T_s is status exchange period.
4 Exchange information with all processor
5 Count Arrival of new jobs j in processor i
6 If (processor=idle_state)
7 Start processing job j
8 Else
9 If ($T^j_{migration}$<Migration_Limit)
10 Estimate EFTij
11 $EFT^j_k=Q_i(CST)/\ \mu_i(T_{n-1})+ERT^j_i$
12 For all processor,
13 Estimate EFT^j_k
$\max(L_{k,i}(CST)+EA(t^j_c)-ED_k(t^j_c))/\ \mu_k(T_{n-1})+ERT^j_k$.
//Q_i is number of jobs waiting in queue
//CST: Current System Time
14 Substitute $t^j_c=t=CST+t^j_c-T_{n-1}$ for first term
15 If($EFT^j_k<EFT^j_i$)
16 Initiation job migration on processor k
17 Else
18 Put job j on waiting queue of processor i.
19 Else
20 Put job j on waiting queue for processor i.

END.

We design and propose a new algorithm, referred to as LBA, which balances load by transferring a job on its arrival epoch rather than waiting for the next transfer instant. This is clearly a faster reaction to respond to higher arrival rates on smaller Grids. In the LBA algorithm, instead of estimating the expected finish time of a job at every estimation time period Te, it will be calculated on each arrival of a job to a processor. Here, estimating the finish time of a job is an aperiodic event, and job migration will now happen a periodically. Therefore, when the load is not distributed evenly across all processors, a job will be migrated to lightly loaded processors much faster in the LBA approach than in (M) ELISA.

V. RESULT AND PERFORMANCE ANALYSIS

The proposed system is design on 32-bit Windows OS with 1.84 GHz Processor and 2 GB RAM. The framework is design on Java platform. The system design has considered datagrid storage design considering hard-drive storage and tape storage. In this work, we have considered three performance metrics of relevance at three different levels. At the job level, we consider the ART of the jobs processed in the system as the performance metric. If N jobs are processed by the system, then

$$\text{Average Response Time} = \frac{1}{N}\sum_{i=1}^{N}(Finish_i - Arrival_i)$$

where Arrivali is the time at which the ith job arrives, and Finishi is the time at which it leaves the system. The delay due to the job transfer, waiting time in the queue, and processing time together constitute the response time. At the processor level, we consider resource utilization as the performance metric. It is the ratio between the processor's busy time to the sum of the processor's busy and idle time:

$$U_i = \frac{Busy_i}{(Busy_i + idle_i)}$$

where $Busy_i$ indicates the amount of time P_i remains busy, and $Idle_i$ indicates the amount of time P_i remains idle during the total execution time of N jobs. The simulation design is performed individually for implementing ELISA, MELISA, LBA, and PIA (Perfect Information Algorithm) algorithm.

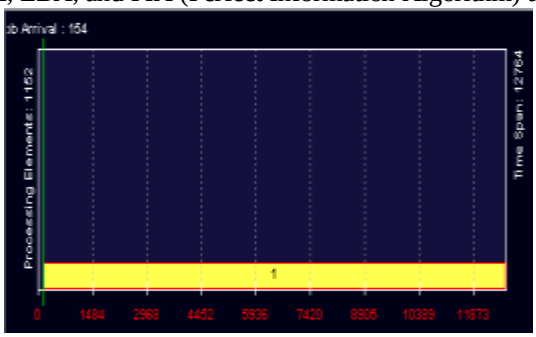

Figure 1 Simulation Result-I

Gridlet ID	User ID	#Jobs	file size	Exe.Time
0	0	3500	300	300
1	0	5000	500	500
2	0	9000	900	900
3	1	4502	140	237
4	1	23447	117	284
5	1	9062	147	234
6	2	13272	141	265
7	2	6108	147	306

Figure 2 Simulation Result-II

Figure 1 shows the simulation design considering time span, job arrival rate, and processing elements. Whereas Figure 2 shows the second set of simulation that highlights Gridlet ID, User ID, Number of Jobs, File Size, and Execution Time in flow of simulation. Here, we present the results of our simulation study and compare the performance of our proposed algorithms with the other algorithms. The amount of

information that is made available for use at the instant of decision making for the transfer of jobs is expected to have a significant effect on the relative performance of the algorithms. In our simulation model, we have considered heterogeneous processors connected by communication channels assuming an arbitrary topology generated by a graph generator tool. The network bandwidth connecting two nodes is also arbitrary and varies from 0:5 Mbps to 10 Mbps. All time units are in seconds, so performance metrics (which are ART and the total execution time) are also measured in seconds. These parameter values are used for all cases unless otherwise stated explicitly. The simulation is performed considering Gridlet IDs, User ID, Exchange Status, Start and Finish Time, Duration, Length, and Number of Processing Element in sliding window. The framework visualization is done mainly for waiting jobs, Jobs completed, jobs in execution, and number of jobs migrated.

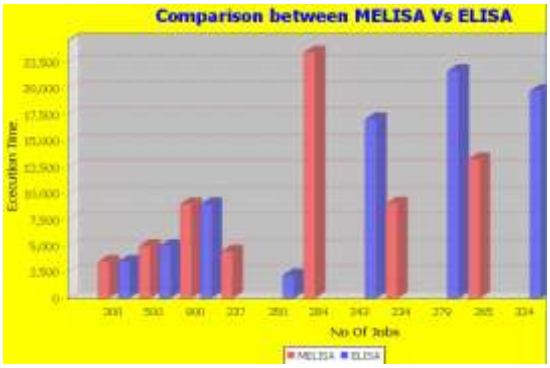

Figure 3 Comparison between MELISA and ELISA

The main difference between ELISA and MELISA is that MELISA takes into account the job migration cost and resource heterogeneity when balancing the load across individual processors. MELISA also takes into account the job migration cost, and as job size largely affects the job migration cost, it will be very interesting to measure the performance of the algorithms by varying the job size. As can be seen from Figure 3, the performance of MELISA is far better than that of ELISA for a heterogeneous Grid environment. This result shows that MELISA is largely suitable for a heterogeneous Grid environment. n Figs., as we increase the job size, the performance of our algorithm becomes far better than ELISA in terms of the decrease in ART and the total execution time. This result indicates that the performance of MELISA outperforms that of ELISA when the job migration cost is very large, which is the case for a large-scale Grid environment.

Figure 4 Comparison between LBA with Existing system (ELISA & PIA)

In this section, we will evaluate the performance of our proposed LBA algorithm with ELISA and PIA. In this set of experiments, we have quantified the performance of our LBA algorithm for real-life situations wherein arrival rates and service rates are completely random. There is not much difference in ART for LBA and that for ELISA, that is, both algorithms exhibit an increasing tendency as we increase the arrival and service rates. ELISA is highly sensitive to the magnitude of the status exchange period Ts. For LBA, increasing the value of Ts also increases ART, but its performance is much better than that of ELISA. However in ELISA, a job has to wait for the next transfer instant before migrating to a lightly loaded processor. In LBA, the variation in processor utilization is less than that for ELISA and for PIA; there is very little or no variation in processor utilization. for a larger job size, the performance of LBA is better than that of ELISA. This is due to the fact that as the job size increases, the migration cost is expected to increase, which prevents migration in LBA. For the LBA algorithm, the job migration cost is also one of the factors for load balancing across its individual processors. Indeed, we can expect that it should give better performance when we increase the job size.

VI. CONCLUSION

In this paper, we presented decentralized, scalable, adaptive, and distributed algorithms for load balancing across resources for data-intensive computations on Grid environments. The objective is to minimize ART and the total execution time for jobs that arrive at a Grid system for processing. Several constraints such as communication delays due to the underlying network, processing delays at the processors, and an arbitrary topology for a Grid system are explicitly considered in the problem formulation. Our algorithms are adaptive in the sense that they estimate different types of strongly influencing system parameters such as the job arrival rate, processing rate, and load on the processor and use this information for estimating the finish time of job on a individual processor. Through this study, we demonstrate the usefulness and effectiveness of the load estimation approach to devise adaptive and dynamic load-balancing strategies for data hungry computational Grid structures.

REFERENCES

[1] Nadia Ranaldo, Giancarlo Tretola, and Eugenio Zimeo "A Scheduler for a Multi-paradigm Grid Environment" INRIA Sophia-Antipolis - Universit´ de Nice - CNRS/I3Se 2004, Route des Lucioles, BP 93 FR-06902 Sophia Antipolis, France

[2] Mrs. Sharada Patil, Prof. Dr. Arpita Gopal "Comparison of Cluster Scheduling Mechanism using Workload and System Parameters" International Journal of Computer Science and Application ISSN: 0974-0767

[3] Malarvizhi Nandagopal Rhymend V Uthariaraj and "Hierarchical Status Information Exchange Scheduling and Load Balancing For Computational Grid Environments Balancing For Computational Grid Environments" IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010

[4] Fangpeng Dong and Selim G. Ak "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems" International Conference on e-Science and Grid Computing

[4] Malarvizhi Nandagopal, Rhymend V. Uthariaraj "Hierarchical Load Balancing Approach in Computational Grid Environment" International J. of Recent Trends in Engineering and Technology, Vol. 3, No. 1, May 2010

[5] P.Neelakantan "DECENTRALIZED LOAD BALANCING IN HETEROGENEOUS SYSTEMS USING DIFFUSION APPROACH" International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.1, January 2012

[6] Hongzhang Shan, Leonid Oliker, Warren Smith, Rupak Biswas "Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration" International Conference on Advanced Computing and Communication, Ahmedabad

[7] Satish Penmatsa, Anthony T. Chronopoulos "Comparison of Price-based Static and Dynamic Job Allocation Schemes for Grid Computing Systems" 2009 Eighth IEEE International Symposium on Network Computing and Applications

[8] Jeremy K. Chen Theodore S. Rappaport Gustavo de Veciana "Iterative Water-filling for Load-balancing in Wireless LAN or Microcellular Networks" Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd, **Volume:** 1 **Page(s):** 117 - 121

[9] Giuseppe Di Fatta , Michael R. Berthold "Decentralized Load Balancing for Highly Irregular Search Problems" Microprocessors & Microsystems,Volume31Issue4,June,2007 Pages 273-281

[10] Lorenzo Muttoni, Giuliano Casale, Federico Granata, Stefano Zanero "Optimal number of nodes for computation in grid environments" Dipt. di Elettronica ed Informazione, Politecnico di Milano, Italy **Page(s):** 282 – 289, 11-13 Feb. 2004

[11] Hans-Ulrich Heiss and Michael Schmitz "Decentralized Dynamic Load Balancing: The Particles Approach" Information Sciences, Vol. 84, Issue 1-2 (May 1995) S. 115 - 128.

[12] Xiaobo Yang, Mark Hayes, Andrew Usher "Developing Web Services in a Computational Grid Environment" Informatics and Computer Science: An International Journal, Volume 84 Issue 1-2, May 1995 Pages 115 - 128

[13] Giuseppe Valetto, Paul Snyder, Daniel J. Dubois, Elisabetta Di Nitto, and Nicol`o M. Calcavecchia "A Self-organized Load-balancing Algorithm for Overlay-based Decentralized Service Networks" Proceedings of the 2011 IEEE Fifth International Conference on Self-Adaptive and Self-Organizing Systems Pages 168-177, ISBN: 978-0-7695-4542-4

[14] Constantino Lagoa, Hao Che and Bernardo A. Movsichoff "Adaptive Control Algorithms for Decentralized Optimal Traffic Engineering in the Internet" To appear in IEEE/ACM TRANSACTIONS ON NETWORKING.

[15] G.Kavitha, V.Sankaranarayanan "Secure Resource Selection in Computational Grid Based on Quantitative Execution Trust" International Journal of Computer and Information Engineering 4:3 2010

[16] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma "Performance Analysis of Load Balancing Algorithms" World Academy of Science, Engineering and Technology 38 2008

[17] MAGDY SAEB, CHERINE FATHY "Modified Diffusion Dynamic Load BalancingEmploying Mobile Agents"

[18] Malarvizhi. N , Gokulnath. K, Rhymend Uthariaraj. V "Load Distribution through Optimal Neighbour Selection in Decentralized Grid Environment" European Journal of Scientific Research ISSN 1450-216X Vol.50 No.4 (2011), pp.575-585, 011

[19] Don Abraham1 and VetriSelvi Vetrian2 "Decentralized Dynamic Load Balancing and Intersection Trust in Mobile Ad Hoc Grids" IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011 ISSN (Online): 1694-0814

[20] S. Prakash, D. P. Vidyarthi "Load Balancing in Computational Grid Using Genetic Algorithm" Advances in Computing: 2011; 1(1): 8-17 DOI: 10.5923/j.ac.20110101.02

[21] Sajal K. Das , Daniel J. Harvey and Rupak Biswas "Adaptive Load-Balancing Algorithms using Symmetric Broadcast Networks"

[22] Han Zhao, Xinxin Liu, Xiaolin Li "DLBEM: Dynamic Load Balancing Using Expectation-Maximization" Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on 14-18 April 2008

[23] Darin England and Jon B. Weissman "Costs and Benefits of Load Sharing in the Computational Grid" Proceedings of the 10th international conference on Job Scheduling Strategies for Parallel Processing Pages 160-175, ISBN:3-540-25330-0 978-3-540-25330-3

[24] Said Fathy El-Zoghdy "A Hierarchical Load Balancing Policy for Grid Computing Environment" I. J. Computer Network and Information Security, 2012, 5, 1-12 Published Online June 2012 in MECS