ISSN: 2277-3061

Service Oriented Load Balancing Framework in Computational Grid Environment

Sukalyan Goswami[1], Ajanta De Sarkar[2]
[1]Assistant Professor, Department of Computer Science & Engineering,
Institute of Engineering & Management, Salt Lake, Kolkata, India.
sukalyan.goswami@gmail.com
[2]Associate Professor, Department of Computer Science & Engineering,
Birla Institute of Technology, Mesra, Deemed University, Kolkata Campus, India.
adsarkar@bitmesra.ac.in

ABSTRACT

Grid computing or computational grid has become a vast research field in academics. It is a promising platform that provides resource sharing through multi-institutional virtual organizations for dynamic problem solving. Such platforms are much more cost-effective than traditional high performance computing systems. Due to the provision of scalability of resources, these days grid computing has become popular in industry as well. However, computational grid has different constraints and requirements to those of traditional high performance computing systems. In order to fully exploit such grid systems, resource management and scheduling are key challenges, where issues of task allocation and load balancing represent a common problem for most grid systems as because the load scenarios of individual grid resources are dynamic in nature. The objective of this paper is to review different existing load balancing algorithms or techniques applicable in grid computing and propose a layered service oriented framework for computational grid to solve the prevailing problem of dynamic load balancing.

Indexing terms/Keywords

Grid computing, load balancing, deadline prioritized scheduling.

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 9, No 3
editor@cirworld.com
www.cirworld.com, member.cirworld.com

ISSN: 2277-3061

I. INTRODUCTION

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce. Recent research on these topics has led to the emergence of a new paradigm known as *Grid Computing* [[12]]. It aggregates dispersed heterogeneous resources for solving various kinds of large-scale applications in science, engineering and commerce. In large-scale grid environments, the underlying network connecting them is heterogeneous and bandwidth across resources varies from link to link. Not limited to grid, in many of today's distributed computing environments, the computers are linked by a delay and bandwidth limited communication medium that inherently inflicts tangible delays. Moreover, the impacts of trust and availability on performance and development difficulty can influence the choice of whether to deploy onto a dedicated computer cluster, to idle machines internal to the developing organization, or to an open external network of volunteers.

Due to uneven task arrival patterns and unequal computing capabilities, the computing node in one grid site may be overloaded while others in a different grid site may be under-utilized. As a result, to take full advantage of such grid systems, task scheduling and resource management are essential functions provided at the service level of the grid software infrastructure, where issues of task allocation and load balancing represent a common problem for most grid systems.

Hence, objective of this paper is to propose a layered service oriented framework to address the problem of load balancing in grid. Section II discusses the related work done in load balancing till date. A few aspect of load balancing in grid and the algorithms are discussed in Section III and Section IV. Section V briefly presents the open challenges of load balancing in computational grid. Functionalities of the proposed layered service oriented framework is explained in section VI and section VII concludes the paper.

II. RELATED WORKS

Load balancing has been discussed in traditional distributed systems literature for long period. Various strategies and algorithms have been proposed, implemented, and classified in a number of studies. In those studies, the load balancing algorithms attempt to improve the response time of a user's submitted applications by ensuring maximal utilization of available resources. The main goal of this type of algorithm is to prevent, if possible, the condition in which some processors are overloaded with a set of tasks while others are lightly loaded or even idle.

In [[6]-[10]], some researchers have proposed several load balancing strategies in grid environments. Cao [[6]] and his co-researchers [[11]] use an ant-like self-organizing mechanism to achieve system-wide grid load balancing through a collection of simple local interactions between grid nodes. In this model, multiple resource management agents cooperate to achieve automatic load balancing of distributed job queues. Each ant takes two sets of m steps in succession to determine the least and the most loaded nodes, respectively. The two nodes then redistribute the load between themselves. After a series of successive redistributions, system-wide uniform load balancing can be achieved.

Yagoubi and Slimani [[7]] propose an algorithm which achieves dynamic load balancing in grid computing. On the basis of a tree model, their algorithm presents the following main features: (i) it implements a layered framework; (ii) it supports heterogeneity and scalability; and (iii) it is totally independent of any physical architecture of a grid.

Erdil and Lewis [[9]] describe information dissemination protocols that can distribute the load in a way without using load rebalancing through job migration, which is more difficult and costly in large-scale heterogeneous grid. Essentially, in their model, nodes adjust their advertising rates and aggressiveness to influence where jobs get scheduled.

Ludwig and Maollem [[10]] propose two new distributed swarm intelligence inspired load balancing algorithms. One algorithm is based on ant colony optimization, and the other algorithm is based on particle swarm optimization.

The discussed approaches have seldom given importance to the deadline stringency of the submitted jobs. This research concentrates on the prioritization of deadline of submitted jobs. There are two aspects of the proposed approach – i) nearer the deadline of the submitted job, higher is its priority and ii) highest priority job gets allocated to least loaded processing nodes. Hence, the proposed 'service oriented framework' described in this paper deals with the provision of the resources is capable of satisfying the Service Quality Agreement (SQA) of the task submitted by client.

III. LOAD BALANCING IN GRID

The load balancing mechanism in grid aims to equally spread the load on each computing node, maximizing their utilization and minimizing the total task execution time. In order to achieve these goals, the load balancing mechanism needs to be 'fair' in distributing the load across the computing nodes implying that the difference between the heaviest-loaded node and the lightest-loaded node should be minimized.

To implement the above discussed policies, grid middleware plays a significant role for creating a computational grid environment. It enables sharing and manages grid components based on user requirements and resource attributes (e.g., type of processor, speed, performance). It is a software that connects other software components or applications in order to provide the facilities, like, execution of data intensive applications on suitable resources in secured manner and allocation of resources are done based on policy.

The functionalities of grid middleware can be majorly classified into three categories: resource management, data management and information services.

Resource management functionality is mainly responsible for (i) resource allocation, (ii) job submission and (iii) remote execution of jobs and receiving the results. It is also responsible for managing job status and progress.

The data management functionality provides support to transfer files among nodes in the grid and for the management of these transfers.

The information services provide support for collecting information in the grid and for querying this information.

All of these are supported by the security infrastructure of grid. This provides security functions, including single/ mutual authentication, confidential communication, authorization, and delegation.

One major aspect of resource management in computational grid is load balancing. Load balancing can be defined by the following policies:

(1) The information policy specifies what workload information and when is to be collected, and from where.
(2) The triggering policy determines the appropriate time at which to start a load balancing operation.
(3) The resource type policy classifies a resource as a server or a receiver of tasks according to its availability status.
(4) The location policy uses the results of the resource type policy to find a suitable partner for a resource provider or a resource receiver.
(5) The selection policy defines the tasks that should be migrated from overloaded resources (source) to the idlest resources (receiver).
In fact, a distributed system consists of policies for the use of the resources and the resources themselves. The policies include load balancing, scheduling, and fault tolerances. Although a grid belongs to the class of distributed systems, traditional policies of the distributed system cannot be applied into a grid directly [[1], [2]]. In addition, although load balancing methods have been intensively studied in conventional parallel and distributed systems, they cannot work in grid architectures because these two classes of environment are radically distinct [[3], [4]]. Indeed, the scheduling of tasks on multiprocessors or multiple computers supposes that the processors are homogeneous and linked with homogeneous and fast networks. The rationale behind this approach is as follows [[1], [2]]:

(1) The resources have the same capabilities.
(2) The interconnection bandwidth between processing elements is high.
(3) Input data is readily available at the processing site.
(4) The overall time spent transferring input and output data is negligible in comparison with the total application duration.

Because of the distribution of a large number of resources in a grid environment and the size of the data to be moved, the traditional distributed approaches are not accurate in a grid [[5]]. Heterogeneity, autonomy, scalability and adaptability, resource selection and computation-data separation of a grid make load balancing more difficult. These challenges bring significant obstacles to the problem of designing an efficient and effective load balancing system for grid environments. Some problems resulting from the above have not been solved successfully yet and still remain open research issues. Thus, it is a challenging problem to design a load balancing framework or system which can integrate all these factors.

IV. LOAD BALANCING ALGORITHM

Load balancing algorithms aim to equally spread the load on each computing node, maximizing their utilization and minimizing the total task execution time. The load scheduling algorithms can be divided into two groups: centralized and decentralized [[19], [25], [34]].

In the centralized [[19]] one, a central controller performs the load distribution among different sites. Since this controller has a general view of all the resources and sites, it can devote every job to its appropriate resource. Of course, increase in the number of resources and sites in this algorithm intensify the problem of bottleneck.

The decentralized approach [[25]] performs load balancing based on the dynamic information which is derived from the sites. Among the advantages of this method, we can refer to scalability and high fault tolerance. A scheduling strategy can be divided into two groups: clairvoyant and non-clairvoyant.

The clairvoyant algorithm [[28]] allocates the jobs to the suitable resources according to the characteristics of the jobs such as service time.

On the other hand, the non-clairvoyant algorithm [[28]] unsystematically performs the job allocation to the resources without considering the characteristics of the job.

Another method used to make load balancing is Branch & Bound algorithm [[30]]. This method makes a collection of response space (nodes) by searching a binary tree and then making an interval of numbers for each available node of the tree. This method prunes a number of intervals and its existing numbers via the use of elimination policy in order to reduce the process of obtaining the load balancing. The branch & bound method is based on the farmer–worker model, and it only follows a worker to devote a job but it does not pay attention to the power of workers.

Among other studies having done, one is the use of dynamic tree model to get the load balancing. This model consists of three levels. The first level that includes leaves acts as sites, the second level acts as clusters and the third one is the Grid level. By using this method, and three available levels, the load balancing is formed in three levels namely *intra-site*, *intra-cluster* and *intra-grid*.

The hybrid methods establish load balancing via the use of First-Come-First-Served (FCFS) [[27]] and genetic algorithms (GA) [[27]]. Whenever the number of jobs entered the queue is low, the FCFS algorithm will be appropriate and suitable. Whenever the number of jobs increases, in a way that they are not placed in the queue, the genetic algorithm is used and the load balancing is provided using sliding window. The sliding window makes only the jobs that are entered this window are scheduled by GA. This window makes the jobs to be allocated to the resources rapidly. The only fault of this method is that switching between two algorithms makes the overhead increase.

One of the proposed algorithms for load balancing is intelligent agents [[11]]. In this method, each agent acts as a local resource. These agents try to reduce the execution time by sharing and interchanging the information with each other. Having hierarchical structure is of the specification of these agents. By using this specification, as well as static and dynamic methods, the agents provide load balancing.

The other proposed algorithm for load balancing is the use of ant colonies. In this method, which is based on ant colony optimization (ACO) [[10]], the ants can move in the form of search-max or search-min. In the first case, an ant moves ahead at random to find a node with overload, then it switches in the form of search-min (underload), and it is the time that the ant makes balance between the heavy-load node and the light one. Using artificial life techniques is one of the methods which provide load balancing in Grid. This method utilizes two kinds of algorithms including genetic and Tabu search (TS) to solve the problem of Grid load balancing. Whenever the space of solution is broad the genetic algorithm can be used. This algorithm performs the job scheduling alternatively and continuously because the load balancing can be done periodically.

The Tabu algorithm [[27]] initiates with a neighborhood structure which includes the list of neighboring nodes. This list is searched. In searching process, the best move is selected. That is the same as optimum solution. Afterward the jobs are allocated to the optimum solution.

The Sufferage, Max–min and Min–min algorithms [[27]] can be statically or dynamically used so as to establish load balancing. Through these algorithms and among a set of entered jobs to the environment, one job is selected in order to be scheduled and it is eliminated from the set. The process continues up to the allocation of the existing jobs. The Min–min algorithm selects the job which contains the minimum completion time (MCT) to allocate, but the Max–min algorithm selects the job which contains maximum completion time, and finally the Sufferage algorithm calculates the Sufferage value for each job, and performs the allocation accordingly. Sufferage value for a task is called the difference between best MCT (minimum MCT) and second best MCT (is the first number which is larger than best MCT).

One of the proposed methods in the field of load balancing is using the load managers. In this method each site in the grid includes a unit named load manager which accepts the jobs entering the system. It has relation to other load managers in other sites. Whenever this unit accepts the job, it executes that job in its local site via local scheduler, but if the site is overloaded, the unit sends the job to the remote site via network. Monitoring and managing the load scheduler and load manager is done by storage manager. Ultimately, all of these units establish the load balancing together. The static load balancing algorithms which use scatter operation are among the existing algorithms used to establish load balancing in this manner in grid.

V. OPEN CHALLENGES

Task scheduling in parallel and distributed systems has been intensively studied, but new challenges in grid environments still make it an interesting topic and call for further exploration. Among them heterogeneity, dynamism, computation and data separation are of prime importance. These unique characteristics of grid computing, which make the design of scheduling algorithms more challenging, are explained in what follows. Although we can look for inspirations in previous research, traditional scheduling models generally produce poor grid schedules in practice.

• Heterogeneity and Autonomy

Although heterogeneity is not new to scheduling algorithms even before the emergence of grid computing, it is still far from fully addressed and a big challenge for scheduling algorithm design and analysis. In grid computing, because resources are distributed in multiple domains in the Internet, not only the computational and storage nodes but also the underlying networks connecting them are heterogeneous. The heterogeneity results in different capabilities for job processing and data access. In traditional parallel and distributed systems, the computational resources are usually managed by a single control point. The scheduler not only has full information about all running/pending tasks and resource utilization, but also manages the task queue and resource pool. Thus it can easily predict the behaviors of resources, and is able to assign tasks to resources according to certain performance requirements. In a grid, however, resources are usually autonomous and the grid scheduler does not have full control of the resources. It cannot violate local policies of resources, which makes it hard for the grid scheduler to estimate the exact cost of executing a task on different sites. The autonomy also results in the diversity in local resource management and access control policies, such as, for example, the priority settings for different applications and the resource reservation methods. Thus, a grid scheduler is required to be adaptive to different local policies. The heterogeneity and autonomy on the grid user side are represented by various parameters, including application types, resource requirements, performance models, and optimization objectives. In this situation, concepts such as application-level scheduling and grid economy are proposed and applied for grid scheduling.

• Performance Dynamism

Making a feasible scheduling usually depends on the estimate of the performance that candidate resources can provide, especially when the algorithms are static. In traditional parallel and distributed systems contention caused by incoming applications can be managed by the scheduler according to some policies, so that its impact on the performance that the site can provide to each application can be well predicted. Computations and their data reside in the same site or data staging is a highly predictable process, usually from a predetermined source to a predetermined destination, which can be viewed as a constant overhead. On the contrary grid schedulers work in a dynamic environment where the performance of available resources is constantly changing. The change comes from site autonomy and the competition by applications for resources. Because of resource autonomy, usually grid resources are not dedicated to a grid application. For example, a grid job submitted remotely to a computer cluster might be interrupted by a cluster's internal job which has a higher priority; new resources may join which can provide better services; or some other resources may become unavailable. The same problem happens to networks connecting grid resources: the available bandwidth can be heavily affected by Internet traffic flows which are non-relevant to grid jobs. For a grid application, this kind of contention results in performance fluctuation, which makes it a hard job to evaluate the grid scheduling performance under classic performance models. From the point view of job scheduling, performance fluctuation might be the most important characteristic of grid computing compared with traditional systems. A feasible scheduling algorithm should be able to be adaptive to such dynamic behaviors. Some other measures are also provided to mitigate the impact of this problem, such as SQA negotiation, resource reservation (provided by the underlying resource management system) and rescheduling.

- **Resource Selection and Computation-Data Separation**

In traditional systems, executable codes of applications and input/output data are usually in the same site, or the input sources and output destinations are determined before the application is submitted. Thus the cost for data staging can be neglected or the cost is a constant determined before execution, and scheduling algorithms need not consider it. But in a grid which consists of a large number of heterogeneous computing sites (from supercomputers to desktops) and storage sites connected via wide area networks, the computation sites of an application are usually selected by the grid scheduler according to resource status and certain performance models. Additionally, in a grid, the communication bandwidth of the underlying network is limited and shared by a host of background loads, so the inter-domain communication cost cannot be neglected. Further, many grid applications are data intensive, so the data staging cost is considerable. This situation brings about the computation-data separation problem: the advantage brought by selecting a computational resource that can provide low computational cost may be neutralized by its high access cost to the storage site. These challenges depict unique characteristics of grid computing, and put significant obstacles to design and implement efficient and effective grid scheduling systems.

VI. PROPOSED FRAMEWORK

The proposed 'service oriented framework' to solve the load balancing problem in computational grid is pictorially depicted in figure 1. The major focus of this framework is the deadline stringency of the submitted jobs.

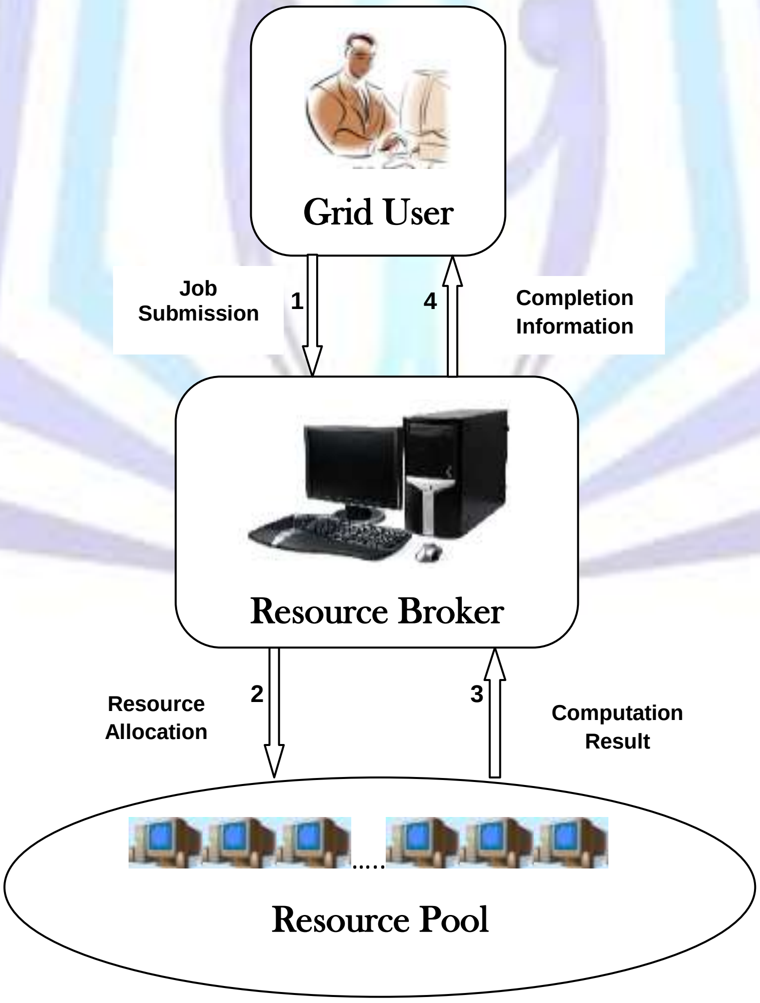

Figure 1 – Service Oriented Load Balancing Framework

The framework consists of three different layers:

a. Grid User – submits computation or data intensive application to grid for execution.

b. Resource Broker – solely responsible for distribution of the jobs in an application to the grid resources based on user's service quality requirements and details of available grid resources for further executions.

c. Resource Pool – pool of resources include cluster, PCs, supercomputer etc.

Initially, users submit the jobs or applications with details through the portal. Then resource broker of the grid collects runtime status or information from the resource pool. Thus, the grid information service collects the details of available grid resources and passes the information to the resource broker and broker keeps a record of those which will be required during task allocation.

On the other side, the tasks being submitted to the broker have their own 'service quality requirements', which, if only mutually agreed between broker and user or client (who submits jobs), then only the task gets submitted to the broker. In this SQA (Service Quality Agreement) stress is given on the deadline for completion of the task efficiently.

The framework works on the notion of priority scheduling, nearer the deadline of the submitted job higher is its priority. And the broker will allocate the highest priority job to the least loaded processing entity, provided that processing entity is capable of satisfying the SQA. If two tasks are submitted to broker with same deadline to completion values, then the one which is submitted first gets allocated to minimal utilized resource first. Other hand, submitted job can not be scheduled in the resources of grid if SQA is not satisfied.

VII. CONCLUSION

The computational grid is a promising platform that provides large resources for distributed algorithmic processing. Such platforms are much more cost-effective than traditional high performance computing systems. In order to fully exploit such grid systems, resource management and scheduling are key grid services, where issues of task allocation and load balancing represent a common problem for most grid systems. This paper presents a study of various load balancing approaches and algorithms in computational grid environment. This paper proposes a layered service oriented framework. This framework puts thrust on two basic parameters, deadline of the job and load status of the resources.

In future, this research will concentrate on implementation of the proposed framework and propose a novel load balancing algorithm in computational grid environment.

REFERENCES

[1] B. Yagoubi, Y. Slimani, Task Load Balancing Strategy in Grid Environment, Journal of Computer Science 3 (3) (2007) 186–194.

[2] B. Yagoubi, Y. Slimani, Load Balancing Strategy in Grid Environment, Journal of Information Technology Applications 4 (2007) 285–296.

[3] S. Hussain, K. Qureshi, H. Rashid, Local predestination with range index communication parallelization strategy for fractal image compression on a cluster of workstations, The International Arab Journal of Information Technology 6 (3) (2009) 293–296.

[4] M.A. Wani, H.R. Arabnia, Parallel edge-region-based segmentation algorithm targeted at reconfigurable multi-ring network, Journal of Super computer 25 (1) (2003) 43–63.

[5] L. Yun Han, L. Seiven, C. Ruay Shiung, Improving job scheduling algorithms in a grid environment, Future Generation Computer Systems 27 (2011) 991–998.

[6] J. Cao, Self-organizing agents for grid load balancing, in: Proceedings of the fifth IEEE/ACM International Workshop on Grid Computing, GRID'04, Pittsburgh, PA, November 2004.

[7] B. Yagoubi, Y. Slimani, Dynamic load balancing strategy for grid computing, Engineering and Technology (2006) 90–95.

[8] http://www.lenders.ch/publications/books/thesis.pdf [visit: 2011-04-01].

[9] D. Erdil, M. Lewis, Dynamic grid load sharing with adaptive dissemination protocols, The Journal of Supercomputing (2010) 1–28.

[10] S. Ludwig, A. Moallem, Swarm intelligence approaches for grid load balancing, Journal of Grid Computing (2011) 1–23.

[11] J. Cao, D. P. Spooner, S. A. Jarvis, G. R. Nudd, Grid load balancing using intelligent agents, Future Generation Computer Systems (ISSN: 0167-739X) 21 (1) (2005) 135–149.

[12] I. Foster, C. Kesselman, S. Tuccke, "The Anatomy of the Grid", International Journal of Supercomputer Applications, 2001.

[13] V. V. Korkhova, J. T. Moscicki, V. V. Krzhizhanovskaya, Dynamic workload balancing of parallel applications with user-level scheduling on the Grid, Future Generation Computer Systems 25 (2009) 28–34.

[14] Y. Li, Y. Yanga, M. Mab, L. Zhou, A hybrid load balancing strategy of sequential tasks for grid computing Environments, Future Generation Computer Systems 25 (2009) 819-828.

[15] L. M. Khanli, S. Razzaghzadeh, S. V. Zargari, A new step toward load balancing based on competency rank and transitional phases in Grid networks, Future Generation Computer Systems 28 (2012) 682–688.

[16] R. Buyya, "Market-Oriented Grid Computing and the Gridbus Middleware", 16th International Conference on Advanced Computing and Communications, 2008. ADCOM 2008.

[17] A. Rajan, A. Rawat, and R. K. Verma, "Virtual Computing Grid Using Resource Pooling", International Conference on Information Technology, 2008. ICIT '08.

[18] J.C. Patni, M.S. Aswal, O.P. Pal, and A. Gupta, "Load balancing strategies for Grid computing", 3rd International Conference on Electronics Computer Technology (ICECT), 2011.

[19] V. Kant Soni, R. Sharma and M. Kumar Mishra, "An analysis of various job scheduling strategies in grid computing", 2nd International Conference on Signal Processing Systems (ICSPS), 2010.

[20] S. Penmatsaa, A. T. Chronopoulos, Game-theoretic static load balancing for distributed systems, Journal of Parallel and Distributed Computing 71 (2011) 537–555.

[21] J.H. Abawajy, Adaptive hierarchical scheduling policy for enterprise grid computing systems, Journal of Network and Computer Applications 32 (2009) 770– 779.

[22] J. Balasangameshwara, N. Raju, A hybrid policy for fault tolerant load balancing in grid computing environments, Journal of Network and Computer Applications 35 (2012) 412–422.

[23] A. Lissy, New scheduling problems coming from grid Computing, Electronic Notes in Discrete Mathematics 36 (2010) 1033–1040.

[24] N. Malarvizhi, Dr. V. R. Uthariaraj, Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment, 6th IEEE International Conference on Autonomic Computing, 2009.

[25] R. Rajavel, De-Centralized Load Balancing for the Computational Grid Environment, International Conference on Communication and Computational Intelligence, Tamil Nadu, India, 2010.

[26] IBM Red Books, Introduction to Grid Computing with Globus, December, 2002.

[27] F. Dong and S. G. Akl, Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, Technical Report No. 2006-504, School of Computing, Queen's University, Kingston, Ontario, January 2006.

[28] S. Zikos, H.D. Karatza, Communication cost effective scheduling policies of non-clairvoyant jobs with load balancing in a Grid, Journal of Systems and Software 82, 2009, 2103–2116.

[29] K.Q. Yan, S.C. Wang, C.P. Chang, J.S. Lin b, A hybrid load balancing policy underlying Grid computing environment, Computer Standards & Interfaces 29, 2007, 161–173.

[30] M. Mezmaz, N. Melab, E.G. Talbi, An efficient load balancing strategy for Gridbased branch and bound algorithm, Parallel Computing 33, 2007, 302–313.

[31] Y. Fei, J. Changjun, D. Rong, Y. Jianjun, Grid resource management policies for load-balancing and energy-saving by vacation queuing theory, Computers and Electrical Engineering 35 , 2009, 966–979.

[32] M.A. Salehi, H. Deldari, B.M. Dorri, Balancing load in a computational Grid applying adaptive intelligent colonies of ants, Journal of Informatica 32, 2008, 327–335.

[33] R. Subrata, A.Y. Zomaya, B. Landfeldt, Artificial life techniques for load balancing in computational Grids, Future Generation Computer Systems 73, 2007, 1176–1190.

[34] J. Balasangameshwara, N. Raju, Performance-Driven Load Balancing with Primary-Backup Approach for Computational Grids with Low Communication Cost and Replication Cost, IEEE Transactions on Computers, Digital Object Identifier 10.1109/TC.2012.44, 2012.

Author' biography with Photo

Sukalyan Goswami has received his M.Tech in Information Technology from IIIT-Bangalore in 2006 and his B.E. in Electronics & Communication Engineering from VTU, Karnataka in 2004. He has six years of teaching experience and one and half years of industry experience. He has two international conference publications. His research interest is grid computing. He is currently engaged with Institute of Engineering & Management, Kolkata as an Assistant Professor in the Department of Computer Science & Engineering.

Dr. Ajanta De Sarkar is working as Associate Professor in the department of Computer Science & Engineering in Birla Institute of Technology, Mesra, Deemed University. She is having altogether seventeen years of experience including six years of industry experience. She had worked for reputed, famous, pioneer companies like, Tata Steel, Jamshedpur, India and Lexis Nexis Inc., Boston, USA. She has proven herself confident and capable of handling real-life projects according to the deadline. Later, she joined academic and started teaching and research simultaneously. She has been awarded PhD in Engineering from Jadavpur University in 2009. Her major interest of teaching includes database, software engineering and distributed computing. Her field of Specialization is Distributed Computing, specifically Grid Computing. Her focused research area includes Grid Computing, Cloud Computing and Wireless Sensor Network. Currently, four students are pursuing PhD under her supervision. She is life member of Computer Society of India.