Distributed Parallel Resource Co-Allocation with Load Balancing in Grid Computing

Neeraj Nehra 1[†] , R.B.Patel 2[††], V.K.Bhat 3[†††]

[†]School of computer Science and Engineering, Shri Mata Vaishno Devi University,Katra(J&K),India.
[††]Department of computer science and Engineering, M.M.Engg.College, Mullana(Ambala),Haryana,India
[†††]School of Applied Physics and Mathematcs, Shri Mata Vaishno Devi University,Katra(J&K),India.

Summary
Resource Co-allocation is one of the crucial problems affecting the performance of the grid. In addition to this if the system load in each of nodes is nearly equal; it indicates good resource allocation and utilization. It is well known that load balancing is a key factor in developing parallel and distributed applications. Instead of balancing the load in grid by process migration, or by moving an entire process to a less loaded node, we make an attempt to balance load by splitting up processes into separated jobs and then balance them to nodes. To address the problem of load balancing, many centralized approaches have been proposed in the literature but centralization has proved to raise scalability tribulations. So in order to get the target, we use mobile agents (MAs) to distribute load among nodes in the grid.

Because a quick response time is necessary for need of users in real grid environment so a real time resource co-allocation is needed for such type of applications. So a parallel resource co-allocation using MA is proposed in this paper which not only balance the load on grid using proposed architecture but also allocate the resources. It is concluded with the results of the experiments that parallel method not only reduces total execution time but also reduces overall response time small.

Key words:
Load balancing, distributed systems, mobile agent and parallel resource co-allocation

1. INTRODUCTION

Grid computing has emerged as an important new field distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and high-performance orientation [1, 2]. Computational grid which is the most common grid [1] consists of large sets of diverse, geographically distributed resources that are grouped for executing specific applications. As the number of grid system components increases, the probability of a failure in the grid computing becomes higher than in a traditional parallel computing [14,15,16,17]. The basic component of grid is available resources, so resource managements can encompass not only a commitment to perform a task but also commitments to level of performance or quality of service [18].

The main components of a grid infrastructure are a security component, resource management services, information services and data management services [3]. The real and specific problem that underlies the Grid concept is to coordinate the shared resources and to solve problems through distributed programs [1]. The sharing that the grid computing is concerned with is not primarily file exchange but rather direct access to nodes, software data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies. The Open Grid Services Architecture (OGSA) enables the integration of services and resources across distributed, heterogeneous, dynamic, virtual organizations—whether within a single enterprise or extended to external resource-sharing and service-provider relationships [2]. Various grid services can be offered under the grid environment, which is defined as a web service that provides a set of well-defined interfaces and that follows specific conventions [4, 5]. Many services offered by the Grid need to access data from a certain source (database), such as the BioMap service using the grid system to identify the genes from open databases [6]. In [5] author presented an example for a web-based grid service in which the grid resources need to access visualization data from another remote server running on the grid.

The relational database management system (RDMS), the computational resources and the data source constitute a general model that can cover most grid services. The partition of a service task into subtasks and their

distribution among available resources are of great concern because they significantly affect the grid service reliability, cost and profits [7].

We will use the concept of mobile agent (MA) for resource management in grid because MA technology offers a new computing paradigm in which an autonomous program can migrate under its own or host control from one node to another in a heterogeneous network. In other words, the program running at a host can suspend its execution at an arbitrary point, transfer itself to another host (or request the host to transfer it to its next destination) and resume execution from the point of suspension is called MA [21]. MA technology provides a new solution to support load balancing with resource management [22]. This approach consists of a number of different types of MAs in a cooperative way to fulfill the task of load balancing instead of single centralized component managing all load-balancing activities. Each type of agent implements one of the predefined policies of load balancing. Moreover, the MA paradigm supports the disruptive nature of wireless links and alleviates its associated bandwidth limitations. The migration of MA is associated with different movement costs viz, transmission time, round trip time, number of hops, etc. MA research evolved over the past few years from the creation of many monolithic MA systems (MASs), often with similar characteristics and built by research groups spread all over the world for optimization and better understanding of specific agent issues [22, 23]. To improve the performance of MAs means to optimize their paths on the network. Furthermore, the agent uses a path through a network based upon known infrastructure characteristics. An agent optimizes its transmission between Agent hosts (AHs) [23] with the help of several migration strategies described in [24].

In this paper, we propose a load balancing mechanism with resource management using MA. Each MA executes predefined policy and has a task to be performed. Rest of the paper is organized as follows: Section 2 presents an overview of PMADE. Section 3 gives system architecture for load balancing along with policy selection and agent selection. Section 4 describes resource co-allocation using MA. Section 5 presents Implementation and performance study. Section 6 gives related work and finally article is concluded in Section 7.

2. OVERVIEW OF PMADE

Figure 1 shows the basic block diagram of PMADE (Platform for Mobile Agent Distribution and Execution). Each node of the network has an Agent Host (AH), which is responsible for accepting and executing incoming autonomous Java agents and an Agent Submitter (AS) [25], which submits the MA on behalf of the user to the AH. A user, who wants to perform a task, submits the MA designed to perform that task, to the AS on the user system. The AS then tries to establish a connection with the specified AH, where the user already holds an account. If the connection is established, the AS submits the MA to it and then goes offline. The AH examines the nature of the received agent and executes it. The execution of the agent depends on its nature and state. The agent can be transferred from one AH to another whenever required. On completion of execution, the agent submits its results to the AH, which in turn stores the results until the remote AS retrieves them for the user.

The AH is the key component of PMADE. It consists of the manager modules and the Host Driver. The Host Driver lies at the base of the PMADE architecture and the manager modules reside above it. It is the basic utility module responsible for driving the AH by ensuring proper co-ordination between various managers and making them work in tandem. Details of the managers and their functions are provided in [25]. PMADE provides weak mobility to its agents and allows one-hop, two-hop and multi-hop agents [25]. PMADE has focused on Flexibility, Persistence, Security, Collaboration, and Reliability [26].

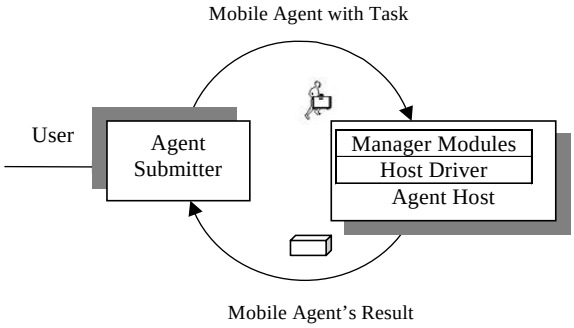


Fig. 1 Block architecture of PMADE

3. SYSTEM ARCHITECTURE FOR LOAD BALANCING

System architecture for load balancing is shown in Figure 2 which contains agents along with associated policies. Each agent executes a predefined policy. Each agent also cooperates with each other for valuable information sharing for update information. Each component of the architecture is as follows:

3.1 POLICY SELECTION

- *Information gathering policy (IGP):* It maintains the information about the workload at the servers. The

policy is made up of two components: frequency of information exchange and the method for dissemination of the information. There is a tradeoff between having accurate information and minimizing the overhead. It also includes the estimation and specification of workload, e.g., processor load, length of queue, storage utility, etc.

- *Initiation policy (PI):* It determines who initiates the process of load sharing. The initiator can be the source server, the destination server, or both (symmetric initiations). The process can be initiated by an overloaded server (called sender-initiated) or by an under-loaded server (called receiver-initiated). Sender initiated policies are those where heavily loaded nodes search for lightly loaded nodes while receiver initiated policies are those where lightly loaded nodes search for suitable senders. These policies can be executed either centralized manner or distributed fashion. We presents the comparative performance of both the policies in coming section.
- *Job transfer policy (JTP):* It decides when the initiator should consider re-allocate some requests to other servers. The decision can be made based on only local state or by exchanging global processor load information. Job reallocation is activated by a threshold-based strategy. In a sender-initiated method, the job transfer is invoked when the workload on a node exceeds a threshold. In a receiver-initiated method, a node starts the process to fetch jobs from other nodes when its workload is below a threshold. The threshold can be a pre-defined static value or a dynamic value that is assessed at runtime based on the load distribution among the nodes. When job reallocation is required, the appropriate job(s) will be selected from the job queue and transferred to another node.
- *Location policy (LP):* It determines to which servers the jobs should be re-allocated. The simplest location policy is to choose a server at random. More complicated policies use negotiation, where the initiator negotiates with each member in a subset of servers.

3.2 AGENT POOL

Agent pool consists of various agents each having its own role. These agents are:

- *Local Scheduler Agent (LSA)*: Its main function is to schedule the incoming jobs. Whenever a request for load comes from any node it will schedule the corresponding request to appropriate node in the network. The request may be of information about system resources, load information, number of processes currently running etc.
- *Grid Dispatcher Agent (GDA):* It is a mandatory component of architecture. The GDA provides services to other agents. Agents may register their services with the GDA or query the GDA to find out what services are offered by other agents. GDA is a mobile agent that implements the location policy. When the server is overloaded, a clone of the GDA will be created. When GDA is activated, using the global loading snapshot stored at the local server, it will carry the jobs in the job reallocation list to the appropriate remote servers for execution. Initially, the GDA will use the global load information stored at the local server site to decide which node to go for load distribution. The GDA presents itself at the receiver site and negotiate with the receiver server locally. There is no need for the sender site to wait for acknowledgement from the receiver as the GDA gets the acknowledgement on behalf of it. In the case where the GDA carries the job to a destination server and finds that the server became overloaded, the GDA can make decision on the fly to find another suitable server by using the current system state information collected while it travels through the destination servers.
- *Load Information Agent (LIA):* This is a mobile agent that implements the information gathering policy component of the load-sharing mechanism. It continuously travels through the servers to collect the global information about the workload and resource utilization at the participated servers. Each server site stores its own copy of the global workload information which will be used for the GDA to make decisions on job reallocation. The global load information will be maintained by the LIA shared by all the servers in the system. An LIA presents itself at a server site and obtains in real time, the accurate, current load information at that site. A single LIA traveling from one site to another continuously, carrying updated load information of previously visited server sites. At each site it visited, an LIA synchronizes itself with the global load information stored at the server. When it arrives at a server site, it will first update its knowledge about the local workload at the site. Then it will retrieve the local copy of the global load information stored at the site, which is in the format of a table containing one entry for each server in the group, and synchronize it with the information collected from the sites it previously visited. Using only one LIA to move among all server sites to collect and maintain global workload

information may result in delay in updating the load information tables at individual server sites. This delay will have impact on the effectiveness of load sharing because the timeless and information coverage of the global workload snapshot are the most critical factors in determining the system performance. Without up to date global load information, it is unlikely that an accurate decision can be made for sharing the heavy workload of an overloaded server. To overcome this difficulty, we can deploy multiple LIAs, each propagates through a portion of the network.

- *Resource Management Agent (RMA):* The resources are managed by RMA the resource management is responsible for gathering information concerning the process nodes on which tasks may execute and pass this information to GDA. There is a proper coordination among the mobile agent for information exchange using mobile group approach [14]. This information includes availability, load average and idle time. Resource management is also responsible for organizing the GDA scheduling and *Task execution.*

3.3 INTER- AGENT COMMUNICATIONS

The framework for load balancing consisting of multi-agents with each agent has a specific role to play and have facility for inter agent communication as shown in Figure 2. Each agent is implemented for managing hosts processors of a Cluster resource and scheduling incoming tasks to achieve load balancing. The function of various layers is as follows:

- *Communication and Coordination Layers:* Agents in the system communicate with each other or with users using mobile group approach for coordination of MAs. The request an agent receives from the communication layer should be explained and submitted to the coordination layer, which decides how the agent should act on the request according to its own knowledge. We assume a distributed system as a collection of agents, locations, and communication channels. A location represents a logical place in the distributed environment where agents execute. When a mobile agent migrates, it moves from a location to another. Agents communicate by exchanging messages through reliable communications channels, i.e., transmitted messages are received uncorrupted and in the sequential sent order, as long as the message sender does not crash until the message is received (reliable channels can be implemented over unreliable channels by tagging transmitted messages with sequential numbers, delivering such messages according to the sequential order and asking for retransmission in case of missing messages). As implied by reliable channel assumption, we assume that network partitions do not occur or, when they occur, they are repaired within a finite amount of time and communication reestablished.

  No bounds on message transmission or relative agent execution times are assumed. Agents and locations are assumed to fail only by crashing (without producing any further action), and the agents of a faulty location are assumed to have crashed. The failure of a given location is not directly handled. Instead, it is only detected when the associated agents are detected faulty. An agent that never crashes is named correct [27]. Let L denote the set of all possible locations. Let P be the set of all possible agents. A mobile group is denoted by the set of agents $g = \{p1, p2, \ldots pn\}$, $g \subset P$. On a mobile group, five operations are defined:

- J*oin (g)*: issued by an agent, when it wants to join group *g*.
- L*eave (g)*: issued by an agent, when it wants to leave group *g.*
- M*ove (g, l)*: issued when an agent wants to move from its current location-to-location *l.*
- S*end (g, m)*: issued by an agent when it wants to multicast a message *m* to the members of group *g.*
- R*eceive (g, m)*: issued by an agent to receive a message *m* multicast from the group *g*.

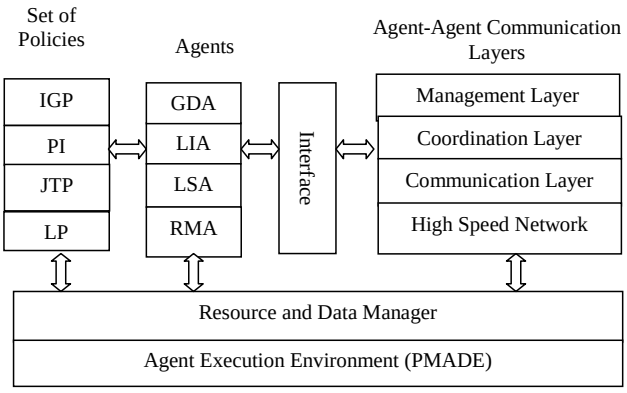


Fig .2 System Architecture for load balancing

- *Management Layer:* This layer is responsible for submitting local service information to the coordination layer for agent decision-making. This layer is responsible for management of resources done by resource and data manger module. RMA is

responsible for conducting various resource management activities. It is the overall manager of all the resources at this layer.

4. RESOURCE CO-ALLOCATION

Figure 3 shows a typical scenario of load balancing with resource management with different modules each has a role to play.

- Agent Management System (AMS) is a mandatory component of architecture. The AMS exerts supervisory control over access to, and use of agents. Only one AMS exists in one local grid. The AMS maintains a directory of agent identifiers (AID) which contains transport addresses for agents registered in local grid. The AMS offers services to other agents. Each agent must register with an AMS in order to get a valid AID.
- The resource management is also major component in the operation of a grid and is controlled by RMA. The basic function of RMA is to accept requests for resources from nodes within the grid and assign specific node resources to a request from the overall pool of grid resources for which the user has access permission. A RMA matches requests to resources, schedules the matched resources, and executes the requests using the scheduled resources. In architecture resources are provided by agents as services and LSA co-operates with RMA for allocation of local high performance resource in a grid environment. The high performance computing capability that a local resource can provide is modeled as a service. Each LSA is a service provider of high performance computing and can register its service information with a GDA in same grid. In a Local grid, AMS is the manager of the local agent system and offers services to other agents. LSA may register their services with the GDA or query the GDA to find out what services are offered by other agents.

In this resource management architecture, Local Grid is organized in a hierarchical manner. A local grid can register its service to another local grid. Here the services of a local grid means all the services provided by the LSA are registered in it. And its registration with other local grid is achieved through interaction between their respective GDA. For examples, if a local grid x will register its service with another local grid y, and then GDA in x will send registration information to GDA in y.

The scheduling on a local grid resource is a ‘‘multiple applications on multiple processors’’ problem. Applications arrive at the resource at different times with different requirements. Resource scheduling in a local resource manager is responsible for deciding when to start running an application, and how many processes should be dispatched to an application. Scheduling method can be divided into centralized and decentralized. The centralized scheduling means- the system has a single scheduler to schedule the entire resources. We adopt the decentralized scheduling. In this architecture agents co-operate with each other to schedule applications that need to utilize the available resources. LSA in the same local grid can interact with each other to schedule the local resources in this local grid; local grid also can interact with each other to scheduling resources in different local grid. In grid computing, there are different types of resources to be shared. Basically resources can be shared in two categories: computing resources, such as network of workstations or personal nodes, and non-computing resources such as data repositories or input/output devices. Non-computing resources are further divided into sharable or non-sharable.

Let there are n computing resources and m computing non-sharable resources and h non-sharable, non computing resources. Each resource has two attributes: type and capability. Type attribute represent type of the attribute and capability specifies available capability of the resource. Communication cost between the resources is handled by communication among the agents. When an application is processed on grid, it is decomposed into tasks such that every task is executed on single computing resource. So each task need a single computing resource and may need some non-computing resource. Each application requires set of tasks, resource sharing constraints among the application’s tasks. Resource sharing constraints arise from the fact that tasks, which may belong to same or different applications, may require use of same non sharable non computing resources.

In this case G=(V,E) is to represent the applications, where nodes represents both tasks and their requirements, directed edges represent the communication requirement and weight of the edge represent the amount of data communications required. For allocating the resources MA executes a parallel resource co-allocation algorithm [28].

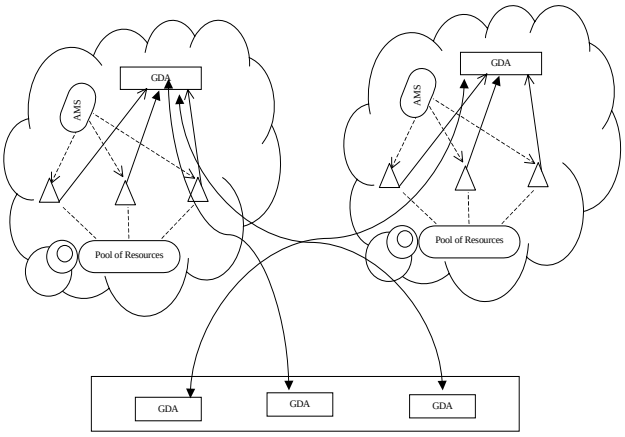


Fig. 3 Typical load balancing with resource management scenario in Grid Computing

The purpose of the algorithm is to minimize the overall scheduling length for a given set of applications. First all the applications are united into single node. Then this node is further partitioned into levels in terms of order of execution determined by precedence. Tasks within a single level are not executed simultaneously due to resource sharing constraints. Finally for each level the maximal independent sets are selected and tasks in each independent set are allocated their required resources. Resource allocation for each level is independent and same method for each level is taken at each level, so that resource allocation processes of all the levels can be executed concurrently. More on parallel resource co-allocation algorithm can be found in [28].

5. PERFORMANCE STUDY AND RESULTS OBTAINED

We have carried out a preliminary study to evaluate the performance of the proposed architecture for load balancing. Performance measures such as the average queue length at each server and the average throughput are used for the evaluation. Random number generators are used to generate the job inter-arrival time and the job service time, both follow an exponential distribution. In the preliminary simulation, for simplicity, workload at a server is defined as the length of the job queue, which represents the number of jobs in the queue. The threshold for dispatching a job by GDA was pre-defined between each experiment.

Figure 4 and Figure 5 show the effect of load sharing on queue length and throughput with and without load sharing, which reflects variance of load. Experiment is conducted with different inter arrival time and performance is compared using load balancing and without load balancing. As shown in Figure 4 for same inter arrival time queue length decreases with load balancing compared to without load balancing. This is due to the job dispatching by MA to corresponding node and consequently queue length decreases. Similarly in Figure 5 for different values of inter arrival time throughput is measured with respect to load balancing and without load balancing. As shown in Figure 5 throughput increases with load balancing compared to without using load balancing.

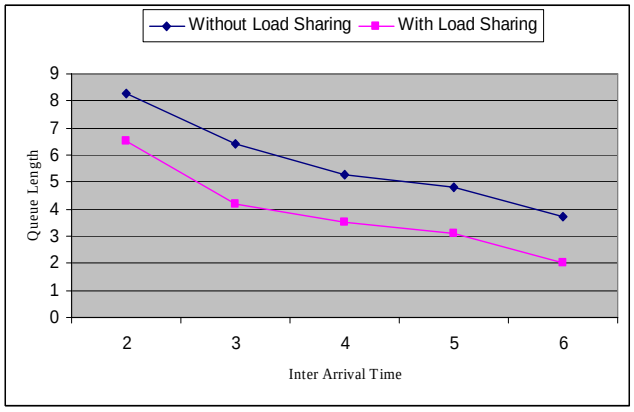


Fig. 4 Effect of load sharing on queue length

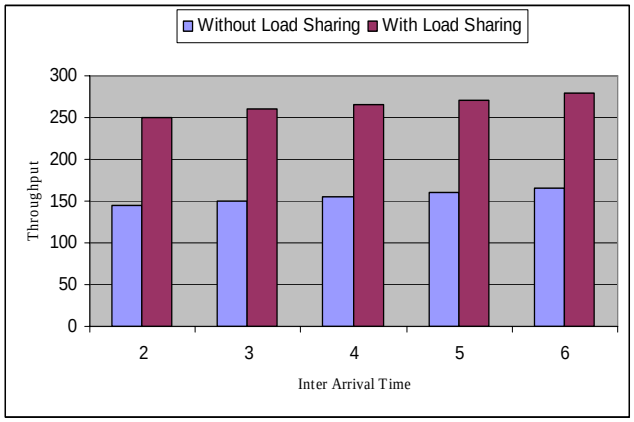


Fig. 5 Effect of load sharing on throughput

Figure 6 compares the relative comparison of policies Sender Initiated (S-I), Receiver initiated (R-I), Symmetrical Initiated (Sy-I) and Central. As shown in Figure 6 S-I policy improves the average response time significantly under varying system load. Under light load all policies are almost identical in terms of response time, but the key difference is under heavy load where S-I policy outperforms R-I and central.

Figure 7 shows the comparison of execution time for parallel and serial allocation. As shown in figure execution time reduces when the resources are allocated parallel compared to serial allocation. For parallel allocation

resources are allocated using MA which executes the predefined parallel resource allocation algorithm defined earlier. As shown in Figure 7 as the number of tasks increases execution time decreases using parallel algorithm compared to serial one.

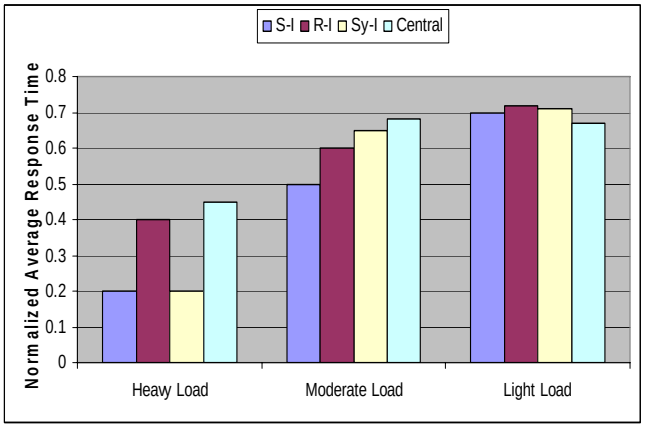


Fig. 6 Normalized average response time under varying load

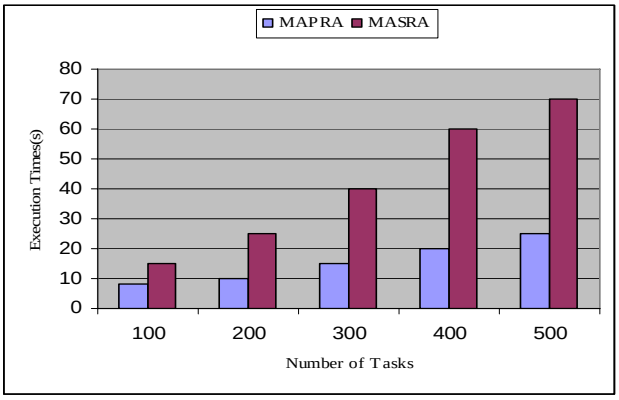


Fig. 7 Comparison of execution time for parallel and serial resource allocation using MA

6. RELATED WORK

Load balancing is indispensable for a grid system to assure even distribution of workload on each node in grid. But one of the most difficult problems that arise on grid system is the selection of an efficient load balancing policy. The load balancing policy should be for evenly utilized grid and a minimum response time for the processed requests. In recent times grid computing has emerged as the attractive computing paradigm for solving a lot of computation intensive applications. But best solution to any computing problem is the execution of job with optimal resource usage.

The most significant attempts can be found in meta-schedulers such as Nimrod-G [29, 30], software execution environments such as GRADS [31] and task brokers such as Condor-G [32]. The latter is a product of a much more complicated entity that consolidates scheduling policies which comprised specialized workload management systems. Additionally, AppLeS [33] is a scheduling system which primarily focuses on developing scheduling agents for individual applications on production. Other interesting works on scheduling and meta-scheduling are presented in [34] and [35] where, in the former, the authors present a heuristic scheduling of bag-of-tasks with QoS constraints, while the latter handles the problem of distributed job scheduling in Grids using multiple simultaneous requests. However, in coherent, integrated Grid environments (such as Globus [36] and Unicore [37]) there are also scheduling and resource management techniques applicable in a more standard manner. Finally, other studies have also addressed resource management in Grids, such as the knapsack formulation problem [38]. In this work the resource allocation in a Grid environment is formulated as a knapsack problem and techniques are developed and deployed so as to maintain the QoS properties of a schedule and at the same time, to maximize the utilization of the grid resources.

In [8] authors described a common grid service model that allowed agents representing various grid resources, which were owned by different real world enterprises. The grid task agents buy resources to complete tasks. Grid resource agents charge the task agents for the amount of resource capacity allocated. In the meantime, the grid task agents charge users who requested the service. In [9] author presented the economical opportunities and realizations through Grid services. They identified the challenges and requirements of economy-based Grid systems, and discussed various representative systems. In [8, 10] authors introduced the optimal task/resource scheduling problems and showed the significant improvement by a good schedule strategy. Some other optimization schemes, proposed for grid are described in [11, 12, 13].

7 Conclusions and Future work

Parallel computing with load balancing and efficient resource co-allocation is necessary for high performance computing. Architecture for load balancing with parallel resource allocation is presented in this paper. The experimental results show that execution time is reduced in parallel algorithm compared to serial one. Throughput is also measured with and without load balancing. Load is balanced using MA approach which has a number of advantages over the existing technology discussed earlier.

Each agent executes a predefined policy and has facility for inter-agent communication.

In the future work we will also consider various others QoS factors to improve the execution time and throughput.

References

[1] Foster, C. Kesselman, The Grid. Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, Los Altos, CA, 1998.

[2] Foster, C. Kesselman, S. Tuecke, The anatomy of the grid enabling scalable virtual organizations, International J. Supercomputer Applications, 15(3), 2001.

[3] B. Jacob, L. Ferreira, N. Bieberstein, C. Gilzean, J. Girard, R. Strachowski, S. Yu, Enabling Applications for Grid Computing with Globus, IBM, 2003.

[4] Foster, C. Kesselman, J.M. Nick, S. Tuecke, Grid services for distributed system integration, Computer, 35(6): 37–46, 2002.

[5] B. Marovi´c, Z. Jovanovi´c, Web-based grid-enabled interaction with 3D medical data, Future Generation Computer Systems, 22(4): 385–392, 2006.

[6] Y.S. Dai, M. Palakal, S. Hartanto, X.Wang, Y. Guo, A grid-based pseudo cache solution for MISD biomedical problems with high confidentiality and efficiency, International Journal of Bioinformatics Research and Applications, 2006.

[7] L. B¨ol¨oni, D. Turgut, D.C. Marinescu, Task distribution with a random overlay network, Future Generation Computer Systems 22(6): 676–687, 2006.

[8] C. Li, L. Li, Competitive proportional resource allocation policy for computational grid, Future Generation Computer Systems, 20(6): 1041–1054, 2004.

[9] R. Buyya, D. Abramson, S. Venugopal, The grid economy, in Proceedings of the IEEE, 93(3): 698–714, 2005.

[10] R. Buyya, M. Murshed, D. Abramson, S.Venugopal, Scheduling parameter sweep applications on global grids. A deadline and budget constrained cost-time optimization algorithm, Software. Practice and Experience Journal, 35(5): 491–512, 2005.

[11] J. Schneider, Searching for Backbones—a high-performance parallel algorithm for solving combinatorial optimization problems, Future Generation Computer Systems, 19(1): 121–131, 2003.

[12] V.P. Gergel, R.G. Strongin, Parallel computing for globally optimal decision making on cluster systems, Future Generation Computer Systems, 21(5): 673–678, 2005.

[13] M. Parashar, H. Klie, U. Catalyurek, et al., Application of Grid-enabled technologies for solving optimization problems in data-driven reservoir studies, Future Generation Computer Systems, 21(1): 19–26, 2005.

[14] N.T. Anh, Integrating fault-tolerance techniques in grid applications, Ph.D. Dissertation, August 2000.

[15] P. Stelling, I. Foster, C. Kesselman, C. Lee, G. von Laszewski, A fault detection service for wide area distributed computations, in Proceedings of 7th IEEE Symposium on High Performance Distributed Computing, 1998.

[16] S. Vadhiyar, J. Dongarra, A performance oriented migration framework for the grid, in Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003.

[17] Foster, C. Kesselman, The Grid 2 Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, Los Altos, CA, 2004.

[18] Foster, A. Roy, V. Sander, A quality of service architecture that combines resource reservation and application adaptation, in Proceedings of 8th International Workshop on Quality of Service, 2000.

[19] F. Vraalsen, R. Aydt, C. Mendes, D. Reed, Performance contracts Predicting and monitoring grid application behavior, in Proceedings of the 2nd International Workshop on Grid Computing, 2001.

[20] Waheed, W. Smith, J. George, J. Yan, An infrastructure for monitoring and management in computational grids, in Proceedings of the 5th Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers, March 2000.

[21] Patel, R.B., Design and implementation of a secure mobile agent platform for distributed computing. Ph.D. Thesis Department of Electronics and Computer Engineering, IIT Roorkee, India, 2004.

[22] Chess, D., B. Grosof, C. Harrison, D. Levine, C. Parris and G. Tsudik, Itinerant agents or mobile computing. IEEE Personal Commun. Mag., 2, 34-49, , 1995.

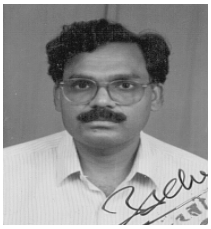
[23] Imielinsky, T. and B.R. Badrinath, Wireless computing. Challenges in Data management. Commun. ACM, 37, 18-28, 1994.

[24] Al-Jaroodi, J., N. Mohamed, J. Hong and D. Swanson, A middleware infrastructure for parallel and distributed programming models on heterogeneous systems, IEEE Trans. Parallel and Distributed Systems, Special Issue on Middleware, 14, 1100-1111, 2003.

[25] Patel, R.B. and K. Garg,, PMADE - A Platform for mobile agent Distribution & Execution, in Proceedings of 5th World Multi Conference on Systemics, Cybernetics and Informatics (SCI2001) and 7th International Conference on Information

System Analysis and Synthesis (ISAS 2001), Orlando, Florida, USA, July 22-25, 2001, Vol. IV, pp. 287-293.
[26] Patel, R.B. and K. Garg, A new paradigm for mobile agent computing, WSEAS Transaction on Computers, 3. 57-64, 2004.
[27] Raimundo, J., A. Macêdo, F.M. Assis Silva, The mobile groups approach for the coordination of mobile agents. J. Parallel Distributed Computing, 65, 275-288, 2005.
[28] Hui-Xian Li, Chun-Tian Cheng, Parallel resource co-allocation for computation grid, Computer languages, systems and structures 33, 1-10, 2007.
[29] D. Abramson, R. Sosic, J. Giddy, B. Hall, Nimrod. A tool for performing parametised simulations using distributed workstations, in Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing, Virginia, August 1995.
[30] D. Abramson, R. Buyya, J. Giddy, A computational economy for grid computing and its implementation in the Nimrod-G resource broker, Future Generation Computing System, 18(8): 1061–1074, 2002.
[31] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, R. Wolski, The GRADS Project. Software support for high-level grid application development, Int. J. High Perform. Computing Application, 15 (4): 327–344, 2001.
[32] J. Frey, T. Tannenbaum, I. Foster, M. Livny, S. Tuecke, Condor-G. A computation management agent for multi-institutional grids, Cluster Computing, 5, 237–246, 2002.
[33] M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, D. Zagorodnov, Adaptive computing on the grid using AppLeS, IEEE Trans. Parallel Distrib. Syst. 14 (4): 369–382, 2003.
[34] C. Weng, X. Lu, Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid, Future Generation Computing System, 21, 271–280, 2005.
[35] V. Subramani, R. Kettimuthu, S. Srinivasan, P. Sadayappan, Distributed job scheduling on computational grids using multiple simultaneous requests, in Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, Edinburgh, Scotland, July 2002, pp. 359–367.
[36] The Globus project, http.//www-fp.globus.org/hbm/.
[37] The Unicore project,http.//www.unicore.org/forum.htm.
[38] R. Parra-Hernandez, D. Vanderster, N.J. Dimopoulos, Resource management and Knapsack formulations on the grid, in Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID'04, 2004.

Neeraj Nehra is pursuing his Ph.D. (Computer Science and Engineering) from SMVDU, Katra (J&K), India. He has received his M.Tech. (Computer Science and Engineering) from K.U.Kurukshetra (Haryana). He is in teaching and research since 2000. He has published 7 research papers in International/National Journals and International /national Conferences. He is in the School of Computer Science and Engineering), Shri Mata Vaishno Devi University, Katra (J&K), India. His research is focused on use of agents, mobile computing, parallel/distributed computing, multi-agent system and fault tolerance, resource management. He is a member of various societies such as International Association of Engineers, Indian Academy of Mathematics. Prior to joining SMVDU, Katra he has worked with HEC Jagadhri and MMEC Mullana, Ambala, Haryana, India.



Dr. R. B. Patel received PhD from IIT Roorkee in Computer Science & Engineering, PDF from Highest Institute of Education, Science & Technology (HIEST), Athens, Greece, MS (Software Systems) from BITS Pilani and B. E. in Computer Engineering from M. M. M. Engineering College, Gorakhpur, UP. Dr. Patel is in teaching and Research & Development since 1991. He has published about 50 research papers in International/National Journals and Refereed International Conferences. He has been awarded for Best Research paper by Technology Transfer, Colorado, Springs, USA, for his security concept provided for mobile agents on open network in 2003. He has written 6 books for engineering courses. He is member of various International Technical Societies such as IEEE-USA, Elsevier-USA, Technology, Knowledge & Society-Australia, WSEAS, Athens, etc for reviewing the research paper. His current research interests are in Mobile & Distributed Computing, Mobile Agent Security and Fault Tolerance, development infrastructure for mobile & peer-to-peer computing, Device and Computation Management, Cluster Computing, etc.

.

Dr.VK Bhat received his Ph.D. (Mathematics) from Jammu University, Jammu (J&K), India. He is a Assistant Professor, SMVDU Katra, India. His research is focused on Ring and modules, Graph theory, and discrete structure. He has 12 years of teaching and research experience. He has published 25 papers in international/national journals and 22 papers in international/national conference proceedings. He is a recipient of UGC (SRF and JRF) fellowship. He is a member of various international societies such as American Mathematical Society, Indian Academy of Mathematics for reviewing the research papers. He is in the editorial board of various International Journal.