Survey of load balancing techniques for Grid

Deepak Kumar Patel a,⁎, Devashree Tripathy b, C.R. Tripathy a

a Department of Computer Science & Engineering, Veer Surendra Sai University of Technology, Burla, Sambalpur 768018, Odisha, India
b CSIR-Central Electronics Engineering Research Institute, Pilani 333031, Rajasthan, India

A R T I C L E   I N F O

Article history:
Received 28 November 2014
Received in revised form 8 February 2016
Accepted 17 February 2016
Available online 24 February 2016

Keywords:
Grid computing
Distributed systems
Load balancing

A B S T R A C T

In recent days, due to the rapid technological advancements, the Grid computing has become an important area of research. Grid computing has emerged a new field, distinguished from conventional distributed computing. It focuses on large-scale resource sharing, innovative applications and in some cases, high-performance orientation. A Grid is a network of computational resources that may potentially span many continents. The Grid serves as a comprehensive and complete system for organizations by which the maximum utilization of resources is achieved. The load balancing is a process which involves the resource management and an effective load distribution among the resources. Therefore, it is considered to be very important in Grid systems. The proposed work presents an extensive survey of the existing load balancing techniques proposed so far. These techniques are applicable for various systems depending upon the needs of the computational Grid, the type of environment, resources, virtual organizations and job profile it is supposed to work with. Each of these models has its own merits and demerits which forms the subject matter of this survey. A detailed classification of various load balancing techniques based on different parameters has also been included in the survey.

© 2016 Elsevier Ltd. All rights reserved.

Contents

1. Introduction .................................................................................................................. 104
2. Background .................................................................................................................. 105
  2.1. Load balancing challenges in Grid computing ...................................................... 105
      2.1.1. Heterogeneity .................................................................................................. 105
      2.1.2. Autonomy ...................................................................................................... 105
      2.1.3. Scalability ...................................................................................................... 105
      2.1.4. Adaptability .................................................................................................... 105
      2.1.5. Dynamic behavior ......................................................................................... 105
      2.1.6. Application diversity ...................................................................................... 105
      2.1.7. Resource non-dedication .............................................................................. 105
      2.1.8. Resource selection and computation-data separation ................................ 106
  2.2. Methods of performing load balancing in Grid ................................................... 106
  2.3. Load balancing performance metrics .................................................................... 106
3. Load balancing survey ................................................................................................ 116
4. Load balancing applications ....................................................................................... 116
5. Adoption graph of load balancing techniques .............................................................. 116
6. Conclusions ................................................................................................................ 116
References ...................................................................................................................... 116

⁎ Correspondence to: Veer Surendra Sai University of Technology, Burla, Sambalpur 768018, Odisha, India.
E-mail addresses: patel.deepak42@gmail.com (D.K. Patel), devashree.tripathy@gmail.com (D. Tripathy), crt.vssut@yahoo.com (C.R. Tripathy).

http://dx.doi.org/10.1016/j.jnca.2016.02.012
1084-8045/© 2016 Elsevier Ltd. All rights reserved.
1. Introduction

A Grid is a computing and data management infrastructure that provides the electronic underpinning for a global society in business, government, research, science and entertainment (Berman et al., 2003). A computational Grid constitutes the software and hardware infrastructure that provides dependable, consistent, pervasive and inexpensive access to high end computational capabilities (Foster and Kesselman, 1999; Foster, 2002). The Grid integrates networking, communication, computation and information to provide a virtual platform for computation and data management in the same way that the Internet integrates resources to form a virtual platform for information (Berman et al., 2003). The Grid can also be considered as a collection of distributed computing resources over a local or wide area network that appear to an end user as one large virtual computing system (Myer, 2003). The speedy development in computing resources has enhanced the performance of computing systems with reduction in cost. The availability of low cost, high speed networks, powerful computers coupled with the advances and the popularity of the Internet has led the computing environment to be mapped from the traditional distributed systems to the Grid environments (Rathore and Channa, 2014).

A computational Grid enables the effective access to high performance computing resources. It supports the sharing and coordinated use of resources, independently from their physical type and location, in dynamic virtual organizations that share the same goal (Rathore and Channa, 2011). Grid infrastructure provides us with the ability to dynamically link together resources as an ensemble to support the execution of large-scale, resource-intensive, and distributed applications (Berman et al., 2003). With its multitude of heterogeneous resources, a proper scheduling and efficient load balancing across the Grid is required for improving the performance of the system (Shah et al., 2007).

Load balancing has been discussed in traditional distributed systems literature for more than three decades. 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 the 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 (Hao et al., 2012). The process of load balancing algorithms in Grids can be generalized into the following four basic steps as shown in Fig. 1 (Yagoubi et al., 2006; Rathore and Channa, 2014).

---

**Fig. 1.** Basic load balancing steps.

**Fig. 2.** Grid load balancing tree.
(i) Load Monitoring: Monitoring the resource load and state.
(ii) Synchronization: Exchanging load and state information between resources.
(iii) Decision Making: Calculating the new work distribution and making the work moment decision.
(iv) Job Migration: Actual data movement.

The load balancing can be defined by the implementation of these policies (Hao et al., 2012; Yagoubi and Slimani, 2006, 2007a).

(i) The Information Policy specifies what workload information is to be collected, when it is to be collected, and from where.
(ii) The Triggering Policy determines the appropriate time at which to start a load balancing operation.
(iii) The Resource Type Policy classifies a resource as a server or a receiver of tasks according to its status availability and capabilities.
(iv) The Location Policy uses the results of the Resource Type Policy to find a suitable partner for a resource provider or a resource receiver.
(v) The Selection Policy defines the tasks that should be migrated from overloaded resources (source) to the idlest resources (receiver).

The rest of the paper is organized as follows. In Section 2, we summarize the challenges of load balancing in heterogeneous Grid environments and the various methods of performing load balancing in Grid. A detailed survey of various load balancing techniques is presented in Section 3. Section 4 discusses various load balancing applications. In Section 5, the adoption of load balancing techniques is described. Finally, the concluding remarks are presented in Section 6.

2. Background

This section puts this work in perspective by briefly summarizing the challenges of load balancing in heterogeneous Grid environments. It then discusses the various methods of performing load balancing in Grid and the performance metrics of load balancing.

2.1. Load balancing challenges in Grid computing

A distributed system adopts various policies for the use of the resources and for the resources themselves. The policies include load balancing, scheduling, and fault tolerances. Although a Grid belongs to the class of distributed systems, the traditional policies of the distributed systems cannot be applied as such into a Grid directly. In addition, the load balancing methods used in conventional parallel and distributed systems are not applicable in Grid architectures. 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 to not provide accurate results in a Grid. The heterogeneity, autonomy, scalability, adaptability, dynamic behavior, application diversity, resource non-dedication, resource selection and computation-data separation of a Grid makes the load balancing more difficult and challenging (Yagoubi et al., 2006; Hao et al., 2012).

2.1.1. Heterogeneity

The heterogeneity exists in both the computational and networks resources. Firstly, the networks used in Grids may differ significantly in terms of their bandwidth and communication protocols. Secondly, computational resources are usually heterogeneous. Because resources may have different hardwares such as instruction set, processors, CPU speed, memory size and different softwares like operating systems, file systems and so on (Yagoubi and Slimani, 2006, 2007a).

2.1.2. Autonomy

Typically a Grid may comprise of multiple administrative domains. Each domain shares a common security and management policy. Each domain usually authorizes a group of users to use the resources in the domain. Thus, the application from non-authorized users should not eligible to run on the resources in some specific domains. Because, the multiple administrative domains share Grid resources, a site is viewed as an autonomous computational entity. It usually has its own scheduling policy, which complicates the task allocation problem. A single overall performance goal is not feasible for a Grid system since each site has its own performance goal and the scheduling decision is made independently of other sites according to its own performances (Yagoubi and Slimani, 2006, 2007a).

2.1.3. Scalability

A Grid may grow from a few resources to millions. This raises the problem of potential performance degradation as the size of a Grid increases (Yagoubi and Slimani, 2006, 2007a).

2.1.4. Adaptability

In a Grid, the resource failure may occur frequently. That means the probability that some resources may fail is naturally high. The resource managers must tailor their behavior dynamically so that they can extract the maximum performance from the available resources and services (Yagoubi and Slimani, 2006, 2007a).

2.1.5. Dynamic behavior

The pool of resources can be assumed to be fixed or stable in the traditional parallel and distributed computing environments. However, in Grid, both the networks and computational resources may work dynamically. First, a network shared by many execution domains cannot provide guaranteed bandwidth. This particularly true when the wide-area network like the internet is involved. Second, both the availability and capability of computational resources may exhibit dynamic behavior. On one hand new resources may join the Grid and on the other hand, some of the existing resources may become unavailable due to problems like network failure. The resource managers must tailor their behaviors dynamically so that they can extract the maximum performance from the available resources and services (Yagoubi et al., 2006).

2.1.6. Application diversity

The Grid applications involve a wide range of users, each having its own special requirements. For example, some applications may require sequential executions, some may consist of a set of independent jobs and other may consist of a set of dependent jobs. In this context, building a general purpose load balancing system seems extremely difficult. An adequate load balancing system should be able to handle a variety of applications (Yagoubi et al., 2006).

2.1.7. Resource non-dedication

The resource usage contention appears as a major issue due to the non-dedication of resources. This results in inconsistency of behavior and performance. For example, in wide area networks, the network characteristics such as latency and bandwidth may be varying over time. Under such an environment, designing an accurate load balancing model is extremely difficult (Yagoubi et al., 2006).
2.1.8. Resource selection and computation-data separation

In traditional systems, the executable codes of applications and input/output data are usually in the same site, otherwise, the input sources and output destinations are determined before the submission of an application. Thus, the cost for data staging can be neglected or the cost-constant is determined before execution. So, the load balancing algorithms need not consider it. But in a Grid, the computation sites of an application are usually selected by the Grid scheduler according to resource status and some performance criterion. Additionally, the communication bandwidth of the underlying network is limited and is shared by a host of background loads, so the communication cost cannot be neglected. This situation brings about the computation-data separation problem. The advantage of it is brought by selecting a computational resource that can provide the low computational cost by neutralizing its high access cost to the storage site (Yagoubi et al., 2006).

The above said challenges put significant obstacles to the problem of designing an efficient and effective load balancing system for the Grid environments. Some such problems resulting from the above have not yet been solved successfully and still remains as an open research issue. Thus, it is a challenging problem to design a load balancing system for the Grid environments that can integrate the above said factors (Hao et al., 2012).

2.2. Methods of performing load balancing in Grid

Fig. 2 depicts a diagrammatic picture of various methods of performing Grid load balancing (Yagoubi et al., 2006).

In general, the load-balancing algorithms are classified as static and dynamic (Yagoubi et al., 2006; Shah et al., 2007; Subrata et al., 2008). The static load-balancing algorithms assume that the information governing load-balancing decisions which include the characteristics of the jobs, the computing nodes, and the communication networks are known in advance. The load-balancing decisions are made deterministically or probabilistically at compile time and remain constant during runtime. However, this is considered to be the drawback of the static algorithm. In contrast, the dynamic load-balancing algorithms attempt to use the runtime state information to make more informative load-balancing decisions. Here, the responsibility for making global decisions may lie with one centralized location, or be shared by multiple distributed locations. Undoubtedly, the static approach is easier to implement and has minimal runtime overhead. However, the dynamic approaches result in better performance. The advantage of dynamic load balancing over static is that the system need not be aware of the runtime behavior of the application before execution.

The dynamic load-balancing algorithms are classified as adaptive and non-adaptive (Yagoubi et al., 2006; Shah et al., 2007). The adaptive algorithms are a special type of dynamic algorithms where the parameters of the algorithm and/or the scheduling policy itself is changed based on the global state of the system. Here, the scheduled decisions take into consideration the past and the current system performance and are affected by previous decisions or changes in the environment. A dynamic solution takes the environment inputs into account while making decisions. On the other hand, an adaptive solution takes the environment stimuli into account to modify the load balancing policy itself. In the non-adaptive scheme, the parameters used in the balancing remain the same regardless of the system’s past behavior.

The dynamic load-scheduling algorithms could also be classified as centralized or distributed algorithms (Yagoubi et al., 2006; Shah et al., 2007; Subrata et al., 2008). In the centralized approach, one node in the system acts as a scheduler and makes all the load-balancing decisions. The information is sent from the other nodes to the scheduler. In the distributed approach, all the nodes of the system remain involved in the load-balancing decisions. It therefore, becomes very costly for each node to obtain and maintain the dynamic state information of the whole system. Here, each node obtains and maintains only the partial information locally to make suboptimal decisions. In distributed load balancing, the state information is distributed among the nodes that are responsible in managing their own resources or allocating tasks residing in their queues to other nodes. However, the distributed algorithms suffer from the problem of communication overheads incurred by frequent information exchange between processors. The centralized strategy on the other hand has the advantage of ease of implementation, but it suffers from the lack of scalability, fault tolerance and the possibility of becoming a performance bottleneck. Therefore, the centralized algorithms are found to be less reliable than the decentralized algorithms.

In distributed load balancing, the assignment or reassignment of a task among the resources should also be considered (Yagoubi et al., 2006). The one-time assignment of a task may be dynamically done but, once it is scheduled to a given resource, it can never be migrated to another one. On the other hand, in the dynamic reassignment process, the jobs can migrate from one node to another even after the initial placement is made. A negative aspect of this scheme is that tasks may endlessly circulate about the system without making much progress.

The local and global load balancing fall under the distributed scheme since a centralized scheme should always act globally (Yagoubi et al., 2006). In local load balancing, each resource polls other resources in its neighborhood and uses this local information to decide up on a load transfer. The primary objective is to minimize remote communication and to efficiently balance the load on the resources. However, in global load balancing scheme, the global information of all or a part of system is used to initiate the load balancing. This scheme requires a considerable amount of information to be exchanged in the system which may affect its scalability.

If a distributed load balancing mode is adapted, the next issue that should be considered is whether the nodes involved in job balancing are working cooperatively or independently (non-cooperatively) (Yagoubi et al., 2006). In the non-cooperative case, the individual loaders act alone as autonomous entities and arrive at decisions regarding their own optimum objects independent of the effects of the decision on the rest of the system.

The techniques of balancing tasks in the distributed systems are divided mainly into three types. Those are sender-initiated, receiver-initiated and symmetrically-initiated (Yagoubi et al., 2006; Shah et al., 2007). In the sender-initiated models, the overloaded nodes transfer one or more of their tasks to more under-loaded nodes. In the receiver-initiated schemes, the under-loaded nodes request tasks to be sent to them from nodes with higher loads. In the symmetrically-initiated approach, both the under-loaded as well as the loaded nodes initiate the load transfers.

2.3. Load balancing performance metrics

The performance impact of any load balancing algorithm can be measured using the following performance metrics.

(1) Makespan or execution time: It is the total application execution time that is measured from the time the first job is sent to the Grid until the last job comes out of the Grid.

(2) Average response time: If \( n \) no. of jobs are processed by the system, then the average response time (ART) is given by

\[
\text{Average Response Time} = \frac{1}{n} \sum_{i=1}^{n} (\text{Finish}_i + \text{Arrival}_i)
\]

where the \( \text{Arrival}_i \) is the time at which the \( i \)th job arrives, and \( \text{Finish}_i \) is the time at which it leaves the system.
Table 1
Survey of load balancing techniques.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Proposed by: research focus/contribution/features</th>
<th>Compared algorithm</th>
<th>Performance metrics/ improvement</th>
<th>Gap/future work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tree based approach</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGDC</td>
<td>Hao et al. (2012): Pays attention towards deadline of tasks and presents a load balancing mechanism based on deadline control</td>
<td>WLB (Buyya and Murshed, 2002a, 2002b), LBEGS (Qureshi et al., 2010), PPUTF (Saha et al., 1995; Paranhos et al., 2003), Min–Min (Maheswaran et al., 1999), Max–Min (Maheswaran et al., 1999)</td>
<td>Finished jobs, unfinished jobs, makespan, resubmitted time</td>
<td>Considers bandwidth, resource processing ability, requirement of job</td>
</tr>
<tr>
<td>PLBA</td>
<td>Rathore and Channa (2014): Proposes a hierarchical load balancing technique based on variable threshold value</td>
<td>WLB (Buyya and Murshed, 2002a, 2002b), LBEGS (Qureshi et al., 2010), Min–Min (Rings et al., 2009), Max–Min (Suresh and Balasubramaniam, 2013; Chen et al., 2013)</td>
<td>Response time, resource allocation efficiency, communication overhead time, makespan</td>
<td>Extends by adjusting the balanced threshold function</td>
</tr>
<tr>
<td>LBEGS</td>
<td>Qureshi et al. (2010): Proposes that the machine entity should be active and should participate in load balancing at its level, this enhancement in GridSim known as the Enhanced GridSim</td>
<td>WLB (Buyya and Murshed, 2002a, 2002b), LBG (Yagoubi and Slimani, 2006, 2007a, 2007b)</td>
<td>Communication overhead, response time, percentage response time gain</td>
<td>Implements various other scheduling and fault tolerance techniques</td>
</tr>
<tr>
<td>LBG</td>
<td>Yagoubi and Slimani (2006), (2007a) and (2007b): Proposes a load balancing strategy based on a tree model, representation of a Grid architecture</td>
<td>Not compared</td>
<td>Average communication time</td>
<td>Not given</td>
</tr>
<tr>
<td>WLB</td>
<td>Buyya and Murshed (2002a) and (2002b): Discuss an object-oriented toolkit, called GridSim, for resource modeling and scheduling simulation</td>
<td>Not compared</td>
<td>Job completion rate, time utilization, budget utilization</td>
<td>Focuses on strengthening the network model by supporting various types of networks with different static and dynamic configurations and cost -based quality of services</td>
</tr>
<tr>
<td>HLBFT</td>
<td>Nanthiya and Keerthika (2013): Addresses the issues of resource failures and user deadline for distribution of the load</td>
<td>LBEGS (Qureshi et al., 2010)</td>
<td>Makespan, communication overhead, hit rate</td>
<td>Not given</td>
</tr>
<tr>
<td>NDFS</td>
<td>Goswami and Sarkar (2013): Proposes an algorithm to solve the prevailing problem of dynamic load balancing with respect to deadline of job submitted by the clients</td>
<td>WLB (Buyya and Murshed, 2002a, 2002b), LBG (Yagoubi and Slimani, 2006, 2007a, 2007b)</td>
<td>Finished jobs</td>
<td>Focuses in the direction of varying number of processing elements, and reduction of communication overheads</td>
</tr>
<tr>
<td><strong>Estimation based approach</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Proposed by: research focus/contribution/features</td>
<td>Compared algorithm</td>
<td>Performance metrics/improvement</td>
<td>Gap/future work</td>
</tr>
<tr>
<td>MELISA, LBA</td>
<td>Shah et al. (2007): Considers the job migration cost, resource heterogeneity, and network heterogeneity, performs load balancing by parameter estimation such as the expected finish time of a job, job arrival rate, CPU processing rate and load on the processor</td>
<td>PIA (Anand et al., 1999), ELISHA (Anand et al., 1999)</td>
<td>Performance metrics/improvement Total execution time, average response time</td>
<td>Gap/future work Extends by providing fault tolerance into the system</td>
</tr>
<tr>
<td>ELISHA</td>
<td>Anand et al. (1999): Uses estimated state information based upon periodic exchange of exact state information between neighboring nodes to perform load scheduling</td>
<td>PIA, NS, RS, NH (Ni and Hwang, 1985)</td>
<td>Mean response time, idle time/ elapsed time</td>
<td>Extends by studying the effect of limiting the buddy set to a fixed number of processors</td>
</tr>
<tr>
<td>HLB</td>
<td>Malavizhi and Uthariaraj (2009): Considers problems such as scalability, heterogeneity of computing resources and considerable job transfer delay/communication cost for computational intensive jobs</td>
<td>MCT (Ritchie and Levine, 2003), PIA (Anand et al., 1999)</td>
<td>Average response time, average processing time</td>
<td>Considers precedence constraint among different tasks of a job and some fault tolerant measures</td>
</tr>
<tr>
<td><strong>Optimization based approach</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ludwig and Moallem (2011), Li (2008), Chen (2008), Rahmeh and Johnson (2010), Nasir et al. (2010), Moradi et al. (2010) and Nikkhah et al. (2010).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Proposed by: research focus/contribution/features</td>
<td>Compared algorithm</td>
<td>Performance metrics/ improvement</td>
<td>Gap/future work</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ANTZ, PRAC-TICALZ</td>
<td>Ludwig and Moallem (2011): Proposes two new distributed swarm intelligence inspired load balancing algorithms</td>
<td>SBA (Zhu et al., 1996)</td>
<td>Makespan, number of communications</td>
<td>Addresses the problem of dynamic resource failure and security in the Grid</td>
</tr>
<tr>
<td>Not named</td>
<td>Li (2008): Addresses the optimal load distribution problem in a non-dedicated Grid computing system with heterogeneous servers processing both generic and dedicated applications</td>
<td>Not compared</td>
<td>Average response time</td>
<td>Formulates for other non-dedicated cluster or Grid computing systems such as clusters of clusters or multi-cluster systems where each server itself is a cluster</td>
</tr>
<tr>
<td>ACO, GJAP</td>
<td>Chen (2008): Considers the heterogeneity of Grid resources, the overhead of job transferring among computing nodes</td>
<td>FIFO, TABU (Armentano and Yamashita, 2000)</td>
<td>Makespan, machine usage</td>
<td>Focuses on deal with machine crash or failure by fault tolerance</td>
</tr>
<tr>
<td>BRS</td>
<td>Rahmeh and Johnson (2010): Introduces a latency reduction factor in the random sampling</td>
<td>Not compared</td>
<td>Communication latency, sampling length</td>
<td>Not given</td>
</tr>
<tr>
<td>EANT</td>
<td>Nasir et al. (2010): Focuses on pheromone trail update and trail limit, determine the best resource to be allocated to the jobs based on job characteristics and resource capacity, and at the same time to balance the entire resources</td>
<td>ANTZ (Moallem and Ludwig, 2009)</td>
<td>Average completion time</td>
<td>Not given</td>
</tr>
<tr>
<td>MCPLB</td>
<td>Moradi et al. (2010): Considers workclass, cost, deadline and herd behavior, suggestions on loading indexes and new resource conditions in accordance with synchronous neighborhood</td>
<td>RandLB, OLB, MCLB, Random (Zikos and Karatza, 2008), MCT (Ritchie and Levine, 2003)</td>
<td>Average response time, execution time, cost-percentage, task failure percentage</td>
<td>Not given</td>
</tr>
<tr>
<td>PLB, MEPLB, MCLPB, MCOSTPLB, MCCOSTPLB</td>
<td>Nikkhah et al. (2010): Considers workclass, cost, deadline and herd behavior, suggestions on loading indexes and new resource conditions in accordance with synchronous neighborhood</td>
<td>RandLB, ML, MET, MELB, MCLB, MCOST, MCOSTLB, MCCOST, MCCOSTLB, Random (Zikos and Karatza, 2008), MCT (Ritchie and Levine, 2003)</td>
<td>Average response time, execution time, cost-percentage, task failure percentage</td>
<td>Not given</td>
</tr>
</tbody>
</table>

**Agent based approach**

In this approach, a combination of intelligent agents and multi-agent approaches is applied to both the local Grid resource scheduling and the global Grid load balancing. Each agent is a representative of a local Grid resource and it utilizes predictive application performance data with iterative heuristic algorithms to engineer local load balancing across multiple hosts. At a higher level, the agents cooperate with each other to balance workload using a peer-to-peer service advertisement and discovery mechanism. The load balancing algorithms based on this approach are described in Cao et al. (2003), (2005), Ahmad et al. (2004), Chen et al. (2004), Cao (2004), Salehi et al. (2006), Wang et al. (2006) and Salehi and Deldari (2006).

| Not named                        | Cao et al. (2003): An agent-based Grid management infrastructure is coupled with a performance-driven task scheduler that has been developed for local Grid load balancing | Not compared       | Advance time of application execution completion, resource utilization, load balancing level       | Test the scalability of the system |
| Not named                        | Ahmad et al. (2004): Presents the design and implementation of distributed analysis and load balancing system for hand-held devices using multi-agents system, also proposes a system, in which mobile agents will transport, schedule, execute and return results for heavy computational jobs submitted by handheld devices | Not compared       | Time distribution                                                                                 | Not given                                        |
| Not named                        | Chen et al. (2004): Introduces into the practical protein molecules docking applications, which run at the DDG, a Grid computing system for drug discovery and design | Not compared       | Robustness                                                                                       | Concerns more elements in the algorithm other than be confined to only CPUs and network bandwidth |
| Not named                        | Cao (2004): Proposes to perform self-organizing load balancing of batch queuing jobs with no explicit QoS requirements across distributed Grid resources and also to evaluate quantitative performance using a modeling and simulation approach | Not compared       | Ants, ants wandering steps, ants wandering style                                                  | Focuses on the refinement of the system prototype and the ant algorithm, discussions on security and data management |
| Not named                        | Cao et al. (2005): Combination of intelligent agents and multi-agent approaches is applied to both local Grid resource scheduling and global Grid load balancing. Here agents cooperate with each other to balance workload using a peer-to-peer service advertisement and discovery mechanism | Not compared       | Total application execution time, average advance time of application execution completion, average load utilization rate, load balancing level | Extends the agent framework with new features such as automatic QoS negotiation, self-organizing coordination, semantic integration, knowledge-based reasoning and ontology based service brokering |
| MLBMM                            | Salehi et al. (2006): Here overloaded nodes get balances through layers. In the first layer, which is node-level, an efficient scheduler tries to use node's resources equally. The | Not compared       | Efficiency, convergence speed, communication count                                                | Plans to prove MLBMM mathematically and to promote ant's intelligence and adaptation |
second layer, which is called neighbor-level, periodically scatters the extra load of overloaded nodes to a limited domain. The third layer, which is Grid-level, is a colony of intelligent ants which spread the regional extra load throughout the Grid.

Artificial life techniques

The artificial life techniques have been used to solve a wide range of complex problems in recent times. The power of these techniques stems from their capability in searching large search spaces, which arise in many combinatorial optimization problems, very efficiently. Due to their popularity and robustness, a genetic algorithm (GA), Simulated Annealing (SA), Fuzzy operators and tabu search (TS) are used to solve the Grid load balancing problem. The load balancing algorithms based on this approach are found in [Akhtar (2007); Subrata et al. (2007); Ma (2010); Wu et al. (2011); Salimi et al. (2014, 2012); and Prakash and Vidyarthi (2011)].

Not named

Akhtar (2007): Predicts the execution time for each task with respect to the resource it is assigned to. The prediction time is based on the current attributes of task, current and historical parameters, like load, memory of resources and so on.

GA, TS

Subrata et al. (2007): Here adaptive memory is used to guide problem solving, also useful in situations where the solution space to be searched is huge, making sequential search computationally expensive and time consuming.

HGLBA

Ma (2010): Aims to assign proper tasks to processor according its performance, so as to minimize the time that execute the applied program, and to precisely estimate the load on the server, assigning new tasks to each server.

OSLS

Wu et al. (2011): This approach circumvents the scalability of job scheduling problem by using an ordinal distributed learning strategy, and realizes multi-agent coordination based on an information sharing mechanism with limited communication.

FLUZZY NSGA-II

Salimi et al. (2014): Improves the famous multi-objective genetic algorithm known as NSGA-II using fuzzy operators to improve quality and performance of task scheduling in the market-based Grid environment.

NSGA-II WITH MUTATION

Salimi et al. (2012): Addresses scheduling problem of independent tasks in the market-based Grid where resource providers can request payment from users based on the amount of computational resource that used by them.

HGA

Li et al. (2009): Proposes a novel load balancing strategy using a combination of static and dynamic load balancing strategies, combine a first-come-first-served algorithm.

Hybrid based approach

The hybrid load balancing method combines the principles of both the static and dynamic load balancing for addressing the problem of resource allocation. They use the metric of update interval for reducing the delay and deadlock. It reduces the waiting time of the jobs and assigns the priority. The load balancing algorithms based on this approach are found in [Yan et al. (2009); Li et al. (2009); and Yan et al. (2007)].

VF

Yan et al. (2009): Proposes a hybrid load balancing policy to integrate static and dynamic load balancing technologies. When a node reveals the possible inability to continue providing resources, the system will then obtain a new replacement node within a short time, to maintain system execution performance.

HGA

Li et al. (2009): Proposes a novel load balancing strategy using a combination of static and dynamic load balancing strategies, combine a first-come-first-served algorithm.
**Table 1 (continued)**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Proposed by: research focus/contribution/features</th>
<th>Compared algorithm</th>
<th>Performance metrics/ improvement</th>
<th>Gap/future work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VF</strong></td>
<td>Yan et al. (2007): Proposes a hybrid load balancing policy which integrated static and dynamic load balancing technologies to assist in the selection for effective nodes. If any selected node can no longer provide resources, it can be promptly identified and replaced with a substitutive node to maintain the execution performance and the load balancing of the system</td>
<td>FCFS (Ritchie and Levine, 2003; Cao et al., 2005), CPU-BASED (Yang et al., 2003)</td>
<td>Task redistribution time, task completion time</td>
<td>Not given</td>
</tr>
<tr>
<td><strong>Neighbor based approach</strong></td>
<td>The neighbor based approach is a dynamic load-balancing technique that allows the nodes to communicate and transfer tasks with their neighbors so that the whole system is balanced after a number of iterations. Since this technique does not require a global coordinator, it is inherently local, fault tolerant and scalable. The load balancing algorithms based on this approach are described in Balasangameshwara and Raju (2013), (2012) and (2010).</td>
<td>PD,NoMinR, DA (Lu et al., 2007), ASAP (Zhu et al., 2011)</td>
<td>Response time, load balancing level, back up response time, replication cost</td>
<td>Consider issues related to security</td>
</tr>
<tr>
<td><strong>PD_MinRC</strong></td>
<td>Balasangameshwara and Raju (2013): Integrate the proposed load-balancing approach with fault-tolerant scheduling namely MinRC and develop a performance-driven fault-tolerant load-balancing algorithm or PD_MinRC for independent jobs</td>
<td>MCT (Braun et al., 2001), MIN–MIN (Braun et al., 2001)</td>
<td>Job redistribution time, job completion time, average response time</td>
<td>Consider issues related to security</td>
</tr>
<tr>
<td><strong>AlgHybrid_LB</strong></td>
<td>Balasangameshwara and Raju (2012): Takes into account Grid architecture, computer heterogeneity, communication delay, network bandwidth, resource availability, resource unpredictability and job characteristics. AlgHybrid_LB juxtaposes the strong points of neighbor-based and cluster based load balancing algorithms</td>
<td>Nobel Fault tolerant technique</td>
<td>Mean response time</td>
<td>Study the impact of communication delay on the model under varying load conditions</td>
</tr>
<tr>
<td><strong>OP</strong></td>
<td>Balasangameshwara and Raju (2010): Proposes a dynamic, symmetric initiated model which takes a decentralized approach to load balancing, the computing nodes in a cluster interact with each other through a symmetrically initiated strategy</td>
<td>Not compared</td>
<td>Not named</td>
<td>Mathematically proof</td>
</tr>
</tbody>
</table>

**Partitioning based approach**

The partitioning of a distributed adaptive Grid for distribution over parallel processors is considered in the context of adaptive multilevel methods for solving partial differential equations. The efficient parallel execution of Grid-oriented scientific calculations requires the partitioning of the Grid that minimizes both the load imbalance and interprocessor communication. For unstructured static Grids, good partitions are obtained with the recursive spectral bisection heuristic, applied to the interdependency graph of the Grid. The load balancing algorithms based on this approach are available in Keyser and Roose (2015), Mitchell (2007), Driessche and Roose (1995) and Kejariwal and Nicolau (2005).

<table>
<thead>
<tr>
<th>Others</th>
<th>Viswanathan (2007): Specially designed to handle large volumes of computationally intensive arbitrarily divisible</th>
<th>Not compared</th>
<th>Not named</th>
<th>Mathematically proof</th>
<th>Multilevel implementations of the spectral bisection algorithm can easily be applied to our alternative spectral bisection heuristic that are an order of magnitude faster</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Load arrival rate</td>
<td></td>
<td>A fading memory could be plugged.</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Authors</td>
<td>Year</td>
<td>Description</td>
<td>Speedup curves, optimal finish time</td>
<td>Makespan</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>------</td>
<td>-------------</td>
<td>-----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>DLT</td>
<td>Bharadwaj et al. (2003)</td>
<td>Not compared</td>
<td>Divisible load theory is a methodology involving the linear and continuous modeling of partitionable computation and communication loads for parallel processing. It adequately represents an important class of problems with applications in parallel and distributed system scheduling.</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>A2DLT</td>
<td>Othman et al. (2008)</td>
<td>Not compared</td>
<td>Presents a new divisible load balancing model known as adaptive DLT (A2DLT) for scheduling the communication intensive Grid applications</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>Not named</td>
<td>Yang (1997)</td>
<td>Not compared</td>
<td>Adopts a decentralized solution for the centralized server for job statistics</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>GT</td>
<td>Subrata et al. (2008)</td>
<td>Not compared</td>
<td>Combines the inherent efficiency of the centralized approach and the fault-tolerant nature of the decentralized approach. The algorithm does not assume any particular distribution for service times of tasks, it only requires the first and second moments of the service times as input.</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>CCOOP, NCOOPC</td>
<td>Pennatsa and Chronopoulos (2011)</td>
<td>Not compared</td>
<td>Using cooperative game theory, CCOOP algorithm provides fairness to all the jobs in a single-class job distributed system and using non-cooperative game theory, NCOOPC algorithm provides fairness to all users in a multi-user job distributed system by taking the communication costs into account.</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>Not named</td>
<td>Arora et al. (2002)</td>
<td>Not compared</td>
<td>Consider the overheads of coordination and communication between the Grid nodes which were assumed to be N-resource servers that varied in their respective capacities across resources, introduces a new load balance Triggering Policy based on the endurance of a node reflected by its current queue length.</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>DLB</td>
<td>Lu et al. (2006)</td>
<td>Not compared</td>
<td>Operates on two job scheduling and load balancing policies. The first is Instantaneous Distribution Policy, which tries to control the job processing rate on each site in the system. The second is Load Adjustment Policy, which tries to continuously reduce load difference among a site and its neighbor sites. Considers the different</td>
<td>Not compared</td>
<td>Makespan</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Proposed by: research focus/contribution/features</td>
<td>Compared algorithm</td>
<td>Performance metrics/ improvement</td>
<td>Gap/future work</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>NORMAL LOAD BALANCER</td>
<td>Job waiting time</td>
<td>Gap/future work</td>
<td>Working towards load balancing and job migration between the meta-scheduler in the real Grid environment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>GAP (Rajavel, 2010): Provides the decentralized load balancing in both meta-scheduler and cluster or resource level. The Triggering Policy is used to initiate the load balancing algorithm, which determines the appropriate time period to start the load balancing operation using the boundary value and threshold value approach.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPAS_DEC</td>
<td>Azzoni and Down (2009): Uses an effective mechanism for state information exchange, which significantly reduces the communication overhead, while quickly updating the state information in a decentralized fashion.</td>
<td>MCT (Ritchie and Levine, 2003)</td>
<td>Average task completion time</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>AlgMinT, AlgMinD</td>
<td>Zheng et al. (2008): Study the effect of pricing on load distribution by considering a simple pricing function. Develop distributed algorithms to decide which group the load should be allocated to, taking into account the communication cost among groups. These algorithms use different information exchange methods and a resource estimation technique to improve the accuracy of load balancing.</td>
<td>NASH (Grosu and Chronopoulos, 2005), NASHP (Pennatsa and Chronopoulos, 2005)</td>
<td>Mean response time, mean cost</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>PLBP</td>
<td>Fathy and Zoghdy (2012): Proposes a fully decentralized two level load balancing policy for balancing the workload in a multi-cluster Grid environment where clusters are located at administrative domains, takes into account the heterogeneity of the Grid computational resources, and it resolves the single point of failure which many of the current policies suffer from.</td>
<td>No. LB, Random (Zikos and Karatza, 2008), Min (Balasangameshwara and Raju, 2010)</td>
<td>Mean response time</td>
<td>Study the effect of the length of information update periodical interval at the global scheduler and local scheduler, increase the reliability of the proposed policy by considering some fault tolerance measures</td>
<td></td>
</tr>
<tr>
<td>HLB</td>
<td>Lu and Zomaya (2007): Integrates static and dynamic approaches to make load distribution and redistribution driven by performance benefit jobs, achieves a balance between the inherent efficiency of centralized approach, and the autonomy, load balancing and fault tolerant features offered by distributed approach.</td>
<td>MCT (Maheswaran et al., 1999)</td>
<td>Average response time</td>
<td>Proposes job execution cost-estimation to reduce the possible impact</td>
<td></td>
</tr>
<tr>
<td>PAD, FZF-PAD</td>
<td>Zikos and Karatza (2009): Study the performance of three scheduling policies at Grid scheduler level i.e. Basic Hybrid, PAD, FZF-PAD which utilize dynamic site load information to route nonclairvoyant jobs to heterogeneous sites, in a 2-level Grid system.</td>
<td>H_LS, H_GS (Zikos and Karatza, 2008), PAD, FZF-PAD</td>
<td>Response time, load information traffic, resource utilization fairness</td>
<td>Apply optimizations on scheduling policies at Grid scheduler level, examine additional metrics such as throughput for feedback between sites and Grid scheduler, simulate the experiment in case of highly variable job service demands</td>
<td></td>
</tr>
<tr>
<td>AWLB</td>
<td>Korkhov et al. (2009): Proposes to enhance the quality of handling multi-task jobs in Grid environment by integrating the AWLB developed for parallel applications on heterogeneous resources.</td>
<td>FIFO</td>
<td>Iteration time, balancing speed up, processors capacity</td>
<td>Plans to enhance the resource selection and matchmaking mechanisms by further development of the automated application performance analysis</td>
<td></td>
</tr>
<tr>
<td>CPU_PM</td>
<td>Singh and Awasthi (2011): Focuses on dynamic load balancing on a network of workstations and to develop a distributed scheduling algorithm for load balancing which takes heterogeneity CPU, memory and disk resource into account.</td>
<td>CM_PM, IO CM_RE, IO CM_PM</td>
<td>Mean slowdown</td>
<td>Evaluate performance of the proposed scheme using feedback control technique</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Karthikumar et al. (2013): Design a fair scheduling approach with equal opportunity to all the jobs, follows the hybrid scheduling by calculating the residue value for each job for a number of iterations until the residue gets down to zero.</td>
<td>Not compared</td>
<td>Fair rates</td>
<td>Design an optimal fault tolerance approach based on check-pointing, classify the incoming job request into local and external site request to optimize the task completion by inducing priority to the jobs</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Lee and Huang (2002): Review the effects of the spatial and temporal heterogeneity on performance of a target task.</td>
<td>Not compared</td>
<td>Average parallel execution time</td>
<td>Develop an application to channel bandwidth allocation in mobile computing</td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>Lu et al. (2006): Consider heterogeneity of sites, makes more powerful sites carry more loads, as jobs executed at fast sites are more likely to execute at high speed, taking into account the different network communication delays between sites can reduce the cost of load movement, and enables quicker response to load imbalances.</td>
<td>NN (Sanders, 1999; Xu et al., 1995)</td>
<td>Average response time</td>
<td>Study better approaches for selection of partner sites.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Wang and Wang (2005); Enhances orbus1.1 software with load balancing service on request chain processing, which should be emphasized in Grid workflow.</td>
<td>Not compared</td>
<td>Fault tolerant service</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Lu et al. (2006): Consider heterogeneity of sites, fast sites are more likely to execute at high speed, taking into account the different network communication delays between sites can reduce the cost of load movement, and enable quick response to load imbalances.</td>
<td>LOCAL, BN, RANDOM (Zikos and Karatza, 2008),</td>
<td>Average response time</td>
<td>Model the impacts of accuracy of job execution time estimation, utilize migration threshold dynamically based on real-time observation of load behavior of system resources, consider network and hardware failures.</td>
<td></td>
</tr>
<tr>
<td>BILB</td>
<td>Rzadca and Trystram (2009): Proposes a simple mathematical model for such systems and a novel function for computing the cost of the execution of foreign jobs depends on the size of a job and on the local load.</td>
<td>Not compared</td>
<td>Mathematically proof</td>
<td>Enhance our algorithm in order to reduce the dispersion of the results observed in the experiments.</td>
<td></td>
</tr>
<tr>
<td>AWLB</td>
<td>Korkhov et al. (2009): Suggests a hybrid resource management approach operates on both application and system levels, combines user-level job scheduling with dynamic workload balancing algorithm.</td>
<td>FIFO</td>
<td>Balancing speed up, execution time</td>
<td>Test other connectivity schemes, such as the different Master-Worker modes, as well as Mesh, Ring and Hypercube topologies.</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Nasir et al. (2010): Based on the combination of local pheromone update and trail limits.</td>
<td>Not compared</td>
<td>Mathematically proof</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>mDELAY</td>
<td>Mehta et al. (2010): Presents a modified delay strategy to significantly enhance delay-based scheduling algorithm, for delaying the scheduling of new jobs instead of dispatching them to one of the overloaded workstations.</td>
<td>DELAY (Hui and Chanson, 1999), ROUND ROBIN</td>
<td>Average completion time</td>
<td>Proposes a two-level service based decentralized framework to implement the mDELAY scheduling strategy for improved performance over the centralized scheduler.</td>
<td></td>
</tr>
<tr>
<td>MACO</td>
<td>Bai et al. (2010): Here, multiple ant colonies work together and exchange information to collectively find solutions with a objective of minimizing the execution time of tasks and the degree of imbalance of computing nodes.</td>
<td>FCFS (Zomaya and Teh, 2001), ACS</td>
<td>Makespan</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>PLBA</td>
<td>Rathore and Chana (2013): Proposes technique based on variable threshold value which can be found out using load deviation is responsible for transfer the task and flow of workload information, introduces a sender initiated policy to reduce the communication overhead.</td>
<td>WLB (Buyya and Murshed, 2002; Buyya and Murshed, 2002), LBEGS (Qureshi et al., 2010)</td>
<td>Response time, resource allocation efficiency</td>
<td>Adjusts the function of the balance threshold and makes it more adaptive to differing environments.</td>
<td></td>
</tr>
<tr>
<td>PROPOSED</td>
<td>Nandagopal et al. (2010): Addresses the problem of load balancing using Min-Load and Min-Cost policies while scheduling jobs to the resources in multi-cluster environment, develops a heuristic taking both the resource load and the network cost into consideration to evaluate the benefits of scheduling jobs to resources in different clusters.</td>
<td>RANDOM (Zikos and Karatza, 2008)</td>
<td>Response time, slow down</td>
<td>Considers some fault tolerant measures to increase the reliability of our algorithm.</td>
<td></td>
</tr>
<tr>
<td>HJS</td>
<td>Reddy and Roy (2012): Addresses two common parameters, namely CPU utilization and heap memory are employed for load balancing and a computational intensive job is executed on a Grid test bed deployed using Gridgain.</td>
<td>FJS</td>
<td>Total execution time</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Erciyes and Payli (2005): The Grid consists of clusters and each cluster is represented by a coordinator. Each coordinator first attempts to balance the load in its cluster and if</td>
<td>Not compared</td>
<td>Mathematically proof</td>
<td>Implements the recovery procedures.</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Proposed by: research focus/contribution/features</td>
<td>Compared algorithm</td>
<td>Performance metrics/ improvement</td>
<td>Gap/future work</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------</td>
<td>--------------------</td>
<td>----------------------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>this fails, communicates with the other coordinators to perform transfer or reception of load</td>
<td>RANDOM (Zhou and Ferrari, 1987), LOWEST (Zhou and Ferrari, 1987), CENTRAL (Zhou and Ferrari, 1987), DPWP (Araujo et al., 1999; Araujo et al., 1999), TLBA (Mello et al., 2004), GAS (Senger et al., 2005)</td>
<td>Average response time</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>DLB</td>
<td>Liao et al. (2010): Presents a Grid-based dynamic load balancing approach for data-centric storage for wireless sensor networks. This scheme is based on two mechanisms, the cover-up and the multi-threshold. The cover-up mechanism can adjust to another storage node dynamically when a storage node is full, while the multi-threshold mechanism can spread the data into several storage for load balancing of the sensor nodes</td>
<td>GHT (Ratnasamy et al., 2002)</td>
<td>Total energy consumption, average of storage space, standard deviation of storage, dropped events</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Ma et al. (2011): Incorporates functional modules Buffer Management and Load Balancing Management over a Grid networking platform, to buffer the read data and share the middleware loading, thereby solving the overloading issues in RFID applications</td>
<td>TRADITIONAL RFID SYSTEM (Park et al., 2007; Cui and Chae, 2007; Pan et al., 2005), CONNECTION POOL MECHANISM (Park et al., 2007; Park et al., 2007)</td>
<td>Processing time, packet loss ratio</td>
<td>Adjusts the number of readers and middleware hosts to enable the system to reach the optimal efficiency, concerns about the security problem</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Khanli et al. (2012): Uses the subtraction of forward and backward ants as a competency rank to take the priority of the sites, and also uses a control word to search the suitable resource as well. The main purpose is to devote jobs to the existing resources based on their processing power.</td>
<td>B&amp;B (Mezmaz et al., 2007)</td>
<td>Makespan, tardiness, cost</td>
<td>Increases the number of existing resources and the jobs entered to the environment can be increased. Also devotes the jobs to the existing resources in the form of grouping</td>
<td></td>
</tr>
<tr>
<td>Not named</td>
<td>Erdil and Lewis (2012): Describes information dissemination protocols that can distribute load, without using load rebalancing through job migration, which is more difficult and costly in large-scale heterogeneous Grids.</td>
<td>Not compared</td>
<td>Query satisfaction, packet overhead, resource utilization, reservation requests</td>
<td>Not given</td>
<td></td>
</tr>
<tr>
<td>M²ON, M²ON*</td>
<td>Jiang et al. (2009): Presents Min-cost and Max-flow Channel based Overlay Network (M²ON), here the communication capability is denoted as M²C (Min-cost and Max-flow Channel) which is obtained using a Labeled Tree Probing (LTP) method.</td>
<td>BON (Bridgewater et al., 2007)</td>
<td>Mean executing time</td>
<td>Obtain accurate topology matching by a better and more flexible fusion function which in turn further optimize the load balancing process</td>
<td></td>
</tr>
</tbody>
</table>
(3) **Finished and unfinished jobs**: The finished rate of jobs or hit rate can be defined as the number of jobs that are successfully completed on the Grid system on the first schedule. Some of the jobs may not be executed before their deadline. The numbers of jobs that cannot be finished on time (unfinished jobs) are also selected as the standard performance criteria.

![Fig. 3. Grid load balancing approaches.](image)

### Table 2 Load balancing applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Proposed by: research focus/contribution/features</th>
<th>Gap/future work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Data Management</td>
<td>Lee (2004): Describes an intelligence balancing for communication data management. The intelligence balancing allows to execute a complex large-scale Grid computing system and share dispersed data assets collaboratively, focuses on the intelligence balancing to each Grid component and various degrees of intelligence</td>
<td>Not given</td>
</tr>
<tr>
<td>Successive Over Relaxation (SOR)</td>
<td>Dobber et al. (2004): Analyze the impact of the fluctuations in the processing speed on the performance of Grid applications as resources are shared among numerous applications, and therefore, the amount of resources available to any given application highly fluctuates over time</td>
<td>Improve the running times for more complex computation-intensive applications with more complex structures</td>
</tr>
<tr>
<td>Aligning Long DNA Sequences</td>
<td>Chen and Schmidt (2004): Apply the computational Grid concept to aligning long DNA sequences and study the new load balancing techniques for hierarchical Grids called “scheduler-worker” under disturbance and for different levels of application-level inter-cluster bandwidths</td>
<td>Identifies more biology applications that profit from hierarchical Grid systems and presents more efficient parallel models to map these applications onto hierarchical Grid systems</td>
</tr>
<tr>
<td>Scatter Operations</td>
<td>Genaud et al. (2004): Modifies the data distributions used in scatter operations, presents a general algorithm which finds an optimal distribution of data across processors, a quicker guaranteed heuristic relying on hypotheses on communications and computations and a policy on the ordering of the processors</td>
<td>Not given</td>
</tr>
<tr>
<td>Scatter Operations</td>
<td>Genaud et al. (2003): Study the replacement of scatter operations with parameterized scatters, allow custom distributions of data</td>
<td>Not given</td>
</tr>
<tr>
<td>Barnes-Hut Algorithm</td>
<td>Alt et al. (2005): Proposes a high-level approach to Grid application programming, based on generic components or skeletons with prepackaged parallel and distributed implementations and integrated load-balancing mechanisms, present an experimental java-based programming system with skeletons and use it on a non-trivial, dynamic application, the Barnes-Hut algorithm</td>
<td>Not given</td>
</tr>
<tr>
<td>Lattice Boltzmann Model</td>
<td>Farina et al. (2006): Modifies the original Lattice Boltzmann model to approximate a diffusive phenomenon that suitably solves the dynamic load balancing problem</td>
<td>Not given</td>
</tr>
<tr>
<td>Cosmology SAMR Simulations</td>
<td>Lan et al. (2006): Design to improve the performance of distributed cosmology simulations, focuses on reducing the redistribution cost through a hierarchical load balancing approach and a run-time decision making mechanism</td>
<td>Investigates multi-level approach and evaluate it against the proposed two-level approach</td>
</tr>
<tr>
<td>Distributed and Integrated Power Systems</td>
<td>Al-Khannak and Bitzer (2007): Develop an interface between the power systems and the Grid computing which interacts with other power systems connected to the Grid computing. Grid computing resources perform real time load forecasting where the results will be returned to each power system for decentralized load balancing operations</td>
<td>Not given</td>
</tr>
<tr>
<td>Grid-based Virtual Reactor</td>
<td>Korkhov et al. (2008): Introduce a generic technique for adaptive load balancing of parallel applications on heterogeneous resources and evaluate it using a case study application: a Virtual Reactor, contains a number of parallel solvers originally designed for homogeneous computer clusters that needed adaptation to the heterogeneity of the Grid</td>
<td>Integrates the adaptive load-balancing algorithm with the DIANE user-level scheduling system, which extends the testing ground to the multitude of real applications executed on the EGEE Grid</td>
</tr>
<tr>
<td>HLA-Based Simulations</td>
<td>Boukerche and Grande (2009): Supports the re-distribution of load for HLA-based simulations running on large-scale distributed systems</td>
<td>Consider the simulation intercommunication to minimizing the communication</td>
</tr>
<tr>
<td>High Level Architecture (HLA) Based Simulations</td>
<td>Grande and Boukerche (2011): Proposes to evenly distribute the load of large-scale HLA based simulations on non-dedicated, heterogeneous environments when computational and communication imbalances are present</td>
<td>Detects communicative federates, achieve better detection of and reactivity to load imbalances by different communications and computation balancing techniques</td>
</tr>
</tbody>
</table>
(4) Resubmitted time or task redistribution time: In a Grid, some Gridlets cannot be finished at the first resource scheduling, but can be scheduled again as its request. The sum of resubmitted time is another standard for our test.

(5) Communication overhead: The communication overheads are calculated by counting the number of messages over Internet, LAN, and Machine.

(6) Efficiency: This is the property of any load balancing algorithm which relate to the amount of resources used by the algorithm. An algorithm must be analyzed to determine its resource usage. For the maximum efficiency, the algorithm should minimize the resource usage.

(7) Throughput: Throughput is the amount of jobs that a system can execute in a given time period.

(8) Fairness: Access to any resources is formally rated by a fairness measure. The fairness measures or metrics determine whether users or applications are receiving a fair share of the system's resources.

(9) Robustness: It is the ability of a computer system to cope with errors during the execution. Robustness can also be defined as the ability of an algorithm to continue operating despite abnormalities in input, calculations, etc.

(10) Latency: It is the time interval between the stimulation and response, or, from a more general point of view, measure of the time delay or waiting that is experienced by some jobs on the system.

3. Load balancing survey

Table 1 summarizes various load balancing techniques that have been proposed over the years for usage in the Grid. The load balancing techniques have been appropriately classified under different approaches as shown in Fig. 3. Their research focus, contribution, features, compared model, performance metrics, improvement, gap and future work have been analyzed.

4. Load balancing applications

Various load balancing applications are discussed below in Table 2.

5. Adoption graph of load balancing techniques

On the basis of the survey, an analysis of trends in publication of load balancing techniques for Grid has been described in Fig. 4.

6. Conclusions

This paper presents an extensive survey of various load balancing techniques that have been proposed over the years for usage in the Grid. The load balancing techniques that are available in the literature have been appropriately classified under different headings. The algorithm, research focus, contribution, features, compared model, performance metrics, improvement, gap and future work of each load balancing technique have been analyzed and presented.

References


Braun T, Siegel HJ, Beck N, Boloni L, Maheswaran M, Reuther A. A comparison of eleven static heuristics for mapping a class of independent tasks onto


