Survey on Grid Resource Allocation Mechanisms

J Grid Computing (2014) 12:399–441
DOI 10.1007/s10723-014-9292-9

Survey on Grid Resource Allocation Mechanisms

Muhammad Bilal Qureshi · Maryam Mehri Dehnavi · Nasro Min-Allah · Muhammad Shuaib Qureshi · Hameed Hussain · Ilias Rentifis · Nikos Tziritas · Thanasis Loukopoulos · Samee U. Khan · Cheng-Zhong Xu · Albert Y. Zomaya

Received: 25 March 2013 / Accepted: 10 February 2014 / Published online: 2 April 2014

Abstract Grid is a distributed high performance computing paradigm that offers various types of resources (like computing, storage, communication) to resource-intensive user tasks. These tasks are scheduled to allocate available Grid resources efficiently to achieve high system throughput and to satisfy user requirements. The task scheduling problem has become more complex with the ever increasing size of Grid systems. Even though selecting an efficient resource allocation strategy for a particular task helps in obtaining a desired level of service, researchers still face difficulties in choosing a suitable technique from a plethora of existing methods in literature. In this paper, we explore and discuss existing resource allocation mechanisms for resource allocation problems employed in Grid systems. The work comprehensively surveys Gird resource allocation mechanisms for different architectures (*centralized, distributed, static or dynamic*). The paper also compares these resource allocation mechanisms based on their common features such as time complexity, searching mechanism, allocation strategy, optimality, operational environment and objective function they adopt for solving computing- and data-intensive applications. The comprehensive analysis of cutting-edge research in the Grid domain presented in this work provides readers with an understanding of essential concepts of resource allocation mechanisms in Grid systems and helps them identify important and outstanding issues for further investigation. It also helps readers to choose the most appropriate mechanism for a given system/application.

Keywords High performance computing · Grid computing · Resource allocation

M. B. Qureshi · H. Hussain
COMSATS Institute of Information Technology, Islamabad, Pakistan

M. M. Dehnavi · N. Min-Allah
Massachusetts Institute of Technology, Cambridge, MA USA

M. S. Qureshi
Department of Computer Science, King Abdulaziz University, Jeddah, Saudi Arabia

I. Rentifis
University of Thessaly, Volos, Greece

N. Tziritas · C.-Z. Xu
Chinese Academy of Sciences, Beijing, China

T. Loukopoulos
Technological Educational Institute of Lamia, Lamia, Greece

S. U. Khan (✉)
North Dakota State University, Fargo, ND, USA
e-mail: samee.khan@ndsu.edu

A. Y. Zomaya
University of Sydney, Sydney, Australia

1 Introduction

The need for reliable, pervasive, and high computing power [32, 63, 64, 93, 94] forces researchers to focus on low-cost intelligent methodologies for sharing data and resources such as computers, software applications, sensors, storage space, and network bandwidth. As a result, *Grid computing* was introduced in 1990s which produced a comprehensive platform for virtual organizations and computing environments to share their owned services [136]. The development of Grid technologies was initially driven by the need of the scientific community to collaborate over the network. Gradually, the demand of high computing power and simultaneous access to multiple distributed resources (software and hardware) inspired the development of various types Grid systems.

A Grid resource can be defined as an entity that needs to carry out an operation by an application. With the ever growing size of Grid technology, scheduling and allocation of Grid resources has become a challenging and complex research area that has gained more popularity amongst researchers in recent years.

The Grid resource allocation (RA) process is comprised of four main functions [6]:

(a) scheduling,
(b) code transfer,
(c) data transmission, and
(d) monitoring.

Resource scheduling is an application-to-resource mapping process consisting of three main phases [33, 49]:

(a) resource discovery, where available resources are searched and a list is generated
(b) resource selection, which selects the best matching resource based on QoS criteria from a list of available resources generated in the resource discovery phase, and
(c) job execution, that involves submitting jobs to the selected resource(s) and monitoring their execution. *Code transfer* involves transferring the code of individual tasks to the allocated resource for execution. *Data transmission* concerns with data transfer needed by a task for its execution. The execution process takes place after the completion of all transfers. Finally, the *monitoring* step is responsible for continuous checking of resource availability, capability, usage and their future reservations [48]. Zanikolas et al. [145] defines monitoring as the process of collecting information about the status and characteristics of resources of interest. Future reservation is the advanced reservation of capable resources based on QoS requirements, reserved prior to task execution, for use at a specific time in the future [141]. The benefit of such reservations is in time availability of resources to applications in the future [34]. We use application, task or job interchangeably in this work.

For executing applications, users interact with the Grid resource broker. The broker performs resource discovery, scheduling, and processing application jobs on the distributed Grid resources [17]. When a job is submitted to the broker, it accesses the Grid information system (GIS) which keeps the status information of all the resources, and obtains information about appropriate resources based on the job requirements [45, 122].

The broker splits the jobs into parts and distributes them to the selected resources. However, if resource performance degradation is detected or a better resource is discovered by the broker that can efficiently execute the job, then the user can restart the job on a different resource. When a job is processed, the result is then returned to the broker and the user is notified [122]. The basic job execution procedure in a Grid is shown in Fig. 1.

On the basis of offered services such as computation, data, interaction, utility, knowledge, and application service, Grids can be classified into four main types [17, 55–57, 67, 69, 106, 141]:

(a) computational Grid (e.g., TeraGrid, ChinaGrid, and APACGrid [107]) that combines the computational power of distributed resources including clusters, desktops, and supercomputers that provide services for high performance computing [55–57],
(b) access Grid [106, 125] that offers limited specific resources for a short period of time,
(c) data Grid (e.g., LHCGrid, GriPhyN [107]) is used by data-intensive tasks that consists of distributed data repositories providing the facilities to store enormous amount of data that can be accessed, moved, and processed as if they were small files [65, 138], and

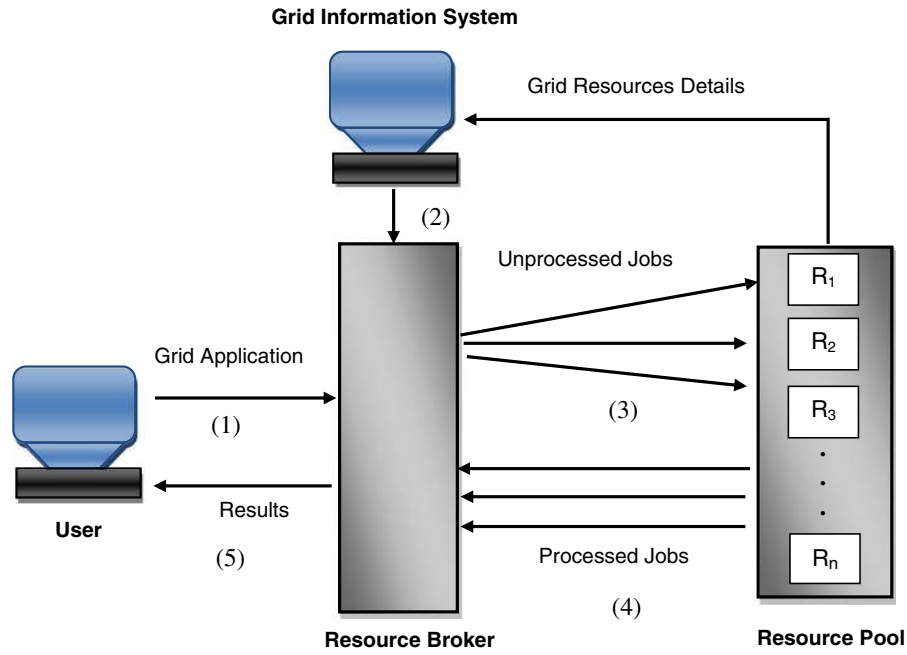

Fig. 1 Basic Grid model [122]

(d) data-centric Grid that facilitates the access and computation of large distributed data repositories [6, 125]. The difference between data and data-centric Grid is that in the latter, heavy computations are moved to data rather than data to computation. Distributed data mining is a typical example application using data-centric Grid.

Ranjan et al. [106, 107] and Yeo et al. [141] further classify Grids into four more types:

(e) *application service Grid* that provides access to remote applications, libraries hosted on various data centers and computational Grids (e.g., NetSolve, GridSolve [17, 69, 107]),
(f) *interaction Grid* that interacts and collaborates to provide visualization between participants,
(g) *knowledge Grid* [106, 113] that focuses on knowledge discovery, acquisition, management, and processing with the aim of providing analytical business services, and
(h) *utility Grid* that provides all the typical Grid services such as data, computing power, discovery and allocation of resources, QoS and contract management. Figure 2 shows the classification of basic Grid types.

Grids can be generally classified into homogeneous or heterogeneous Grids based on factors like operating system, amount of memory, CPU speed, number of resources, architecture types and so on [5]. Each application in Grid environment competes for various resources according to application needs. Figure 3 shows the basic classification of Grid resources.

Because of the growing need for computing resources and the popularity of Grid technologies, efficient, robust and scalable resource allocation (RA) mechanisms are of paramount importance [40]. Such mechanisms should aim at judiciously distributing the limited Grid resources in order to satisfy application needs and achieve common performance objectives.

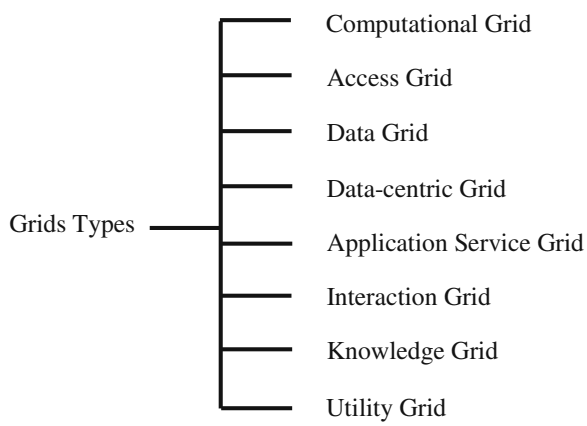

Fig. 2 Taxonomy of Grid types

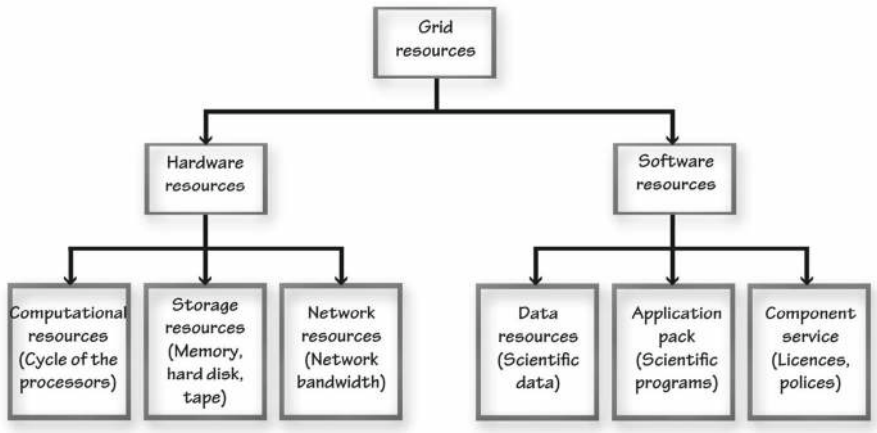

Fig. 3 Basic types of Grid resources

Koopman [68] was the first who investigated RA problems in distributed systems.

Since then, RA problems has been investigated in areas varying from parallel and distributed systems [116, 140], to operations research [140].

A resource allocation mechanism provides the facility of mapping different user tasks to available and suitable resources in order to meet the required specifications, under the allowed boundaries of *virtual organization* (VO) policy environment.

RA mechanisms are usually categorized into static and dynamic [48]. In the static approach, RA is controlled and administered on the basis of past statistics, obtained for instance periodically; while in the dynamic approach, RA meets task's requirements for the resources with the states being changed dynamically. Dynamic allocation is needed whenever resources are discovered and allocated at run time to the tasks. Performance wise, any RA mechanism must not lead to under-utilization of resources. Another issue is the high energy consumption that needs to be alleviated by designing efficient RA mechanisms. In this paper, along with Grid RA mechanisms we also discuss the energy consumption issues so that proper energy-aware features may be included in RA mechanisms to save power in all types of Grid systems. Because RA mechanism plays a vital role in Grid systems, this survey can help in choosing the best-suited RA mechanism for a particular domain.

The rest of the paper is structured as follows: Section 2 discusses brief comparison with existing surveys on resource allocation mechanisms. Section 3 explores resource management. Section 4 describes RA mechanisms. Section 5 gives details of resource management process in Cloud computing environment that is the immediate extension of Grid computing, and finally Section 6 presents some challenges in designing resource allocation mechanisms and concludes the paper.

2 Brief Comparison with the Existing Surveys on Grid Resource Allocation Mechanisms

Quite a few surveys on Grid resource allocation mechanisms exist. Ibaraki et al. [47] and Katoh et al. [51] provided detailed surveys on centralized methods and algorithms for RA problems. However, they restrict their discussion to centralized solutions only. Distributed methods for RA such as market mechanism, compensation and coalition formation, were summarized in the survey by Wu et al. [140]. Distributed solutions were also the focus of Laure et al. [75, 76] with a particular interest on data or data-centric Grids. Krauter et al. [69] surveyed and categorized existing *resource management systems* (RMS) and identified open challenges. Their work placed more emphasis on engineering rather than algorithmic issues. Aside from the fact that some existing reviews on the topic of RA mechanisms are getting old, most of them restricted their scope on particular types of mechanisms e.g., dynamic, centralized or distributed. The motivation for our work is to consolidate all types of mechanisms in a single place. Our goal is to both aid the uninitiated reader by pictorially describing RA mechanisms whenever possible and the experienced researcher by identifying open issues and research avenues to follow.

The following points illustrate the novelty of our work compared to existing surveys.

- Krauter et al. surveyed *resource management systems* in [69] by classifying them on the basis of certain parameters. On the contrary, the focus of this paper is on *resource allocation mechanisms*, thus both the context and the parameters used for taxonomy reasons differ from the aforementioned work.
- The survey conducted by the authors in [107] focuses on resource discovery process in global Grids. They have proposed a taxonomy and studied existing algorithms in scheduling systems. Our scope is broader since the discovery mechanism is only one of the RA components.
- Yeo et al. [116] classified and compared the taxonomy of market-based RMS in *Cluster computing*, while the authors in this paper focus on RA mechanisms over the Grid.
- The authors in [99] studied the problem of resource co-allocation in Grid computing environments. They have pointed out the issues and challenges arising during the process of resource co-allocation and presented possible solutions for such issues. They have also presented brief descriptions of various systems (like GARA, GridARS, OAR, KOALA etc) that support resource co-allocation and compare them on the basis of their features and limitations. However, strategies for allocating Grid resources to tasks were not studied.
- The authors in [142] have discussed the resource selection process with a limited view. They have given an introductory level overview of the three phases of RA process:

  (a) *resource discovery*,
  (b) *resource selection* and
  (c) *resource usage*.

The paper lacks complete and detailed discussion on various methods and mechanisms used for Grid resource allocation. In this paper, we discuss in more detail the resource management system taxonomy, the scheduling process of various Grid resources, the parameters on the basis of which these resources are selected, and the mechanisms developed till now for the allocation of Grid resources. We also compare the Grid RA process with other paradigms based on Grid technology like *Cloud computing* that share many features in resource management process.

- The survey given by Kertesz et al. in [54] briefly discussed resource broker taxonomies in the Grid. These brokers were grouped by only three parameters that are middleware support, job handling, and scheduling. The survey conducted by Kertesz et al. lacks complete working details of these brokers. Instead, in our work, we discuss the resource management system and its working mechanism in a more detailed way because broker is one part of resource management system. In this paper, we are mainly focusing on resource allocation mechanisms in detail used by the Grid for allocating homogeneous or heterogeneous resources to the users.
- The Grid workflow applications are consisting of divisible dependent tasks that are executed in a specific order on distributed Grid resources. Yu et al. [143] have presented a taxonomy for building and executing workflow applications on the Grid. For proving the preciseness of their developed taxonomy, they have surveyed and compared different Grid workflow management systems. Their presented taxonomy gives the general internal architecture of these systems and based on this taxonomy the similarities and differences in these systems architectures are pointed out. In the above mentioned reference, the authors have only explained and surveyed systems for workflow applications. They do not deal with other type of Grid tasks such as divisible independent and atomic tasks. On the other hand, we survey and categorize Grid RA mechanisms for all of the three types of tasks; workflow, divisible independent, and atomic tasks.
- Leal et al. [77] have discussed scheduling in federated Grids. They have discussed three scheduling techniques; static objective, dynamic objective, and static objective advanced scheduling that schedule independent tasks first to the internal Grid resources (resources in the same Grid) and then to the external Grid resources (resources in another Grid). The authors have analyzed the performance of various Grids forming a federated Grid. In our paper, we discuss the resource allocation problem in general that is not restricted by the number of Grids. We describe resource allocation

mechanisms where the task may be independent, dependent, or a workflow application.

3 Resource Management

Every Grid may have three basic features identified by Lamnitchi et al. in [74]:

- The coordinated resources are not subject to centralized control.
- The standards, protocols, and interfaces used should be standardized.
- The QoS delivered is non-trivial.

A *resource management* (RM) is a concrete process of managing Grid resource discovery, scheduling, allocation and system workloads [117] as shown in Fig. 4. The whole process of resource allocation, assignment, authorization, assurance, accounting, authentication and fault tolerance is also the responsibility of RM [53, 117]. *Resource Management System* (RMS) is a Grid service that controls RM processes. The abstract level model of an RMS is shown in Fig. 5. In Grid environment, RM is more complex and challenging than traditional *distributed computing environments* (DCEs) due to the geographic distribution, site autonomy, different usage policies, varying loads, extensibility, co-allocation, adaptability, resource heterogeneity and distributed ownership of the resources [69]. To solve the aforementioned problems, many different approaches and models are developed to manage Grid resources efficiently. Depending on the component organization, RM can be classified into three different types [17, 106]:

(a) *centralized- organization* (e.g., Condor [17, 69]),
(b) *hierarchical organization* (e.g., 2K, Apples, Darwin, Legion [17, 69]), and
(c) *decentralized organization* (e.g., MOL, Bond, Ninf, Javelin, Globus, GOPI [17, 69]).

In a *centralized organization*, the whole process of resources scheduling and allocation is controlled by a single centralized node [69, 109]. This type of organization can be easily managed and deployed but lacks scalability and fault tolerance.

The *hierarchical organization* [109] is different in a way that the resource controllers are managed in a hierarchy that is usually a tree like structure. There is one central manager and multiple low level schedulers. The central manager is responsible for the complete execution of the entire application. The central manager assigns various tasks of the application to the low level schedulers which in turn map the tasks on Grid resources. The resources on the same level can directly communicate with other resources above them or below them without involving any intermediate node. This organization is scalable and fault tolerable but lacks site autonomy. Also, if the central manager fails, then the entire system fails.

In a *decentralized organization*, the control is distributed among multiple nodes with each one making its independent decisions [116] and there is no central authority that has full knowledge of the system. The decentralized organization is more robust,

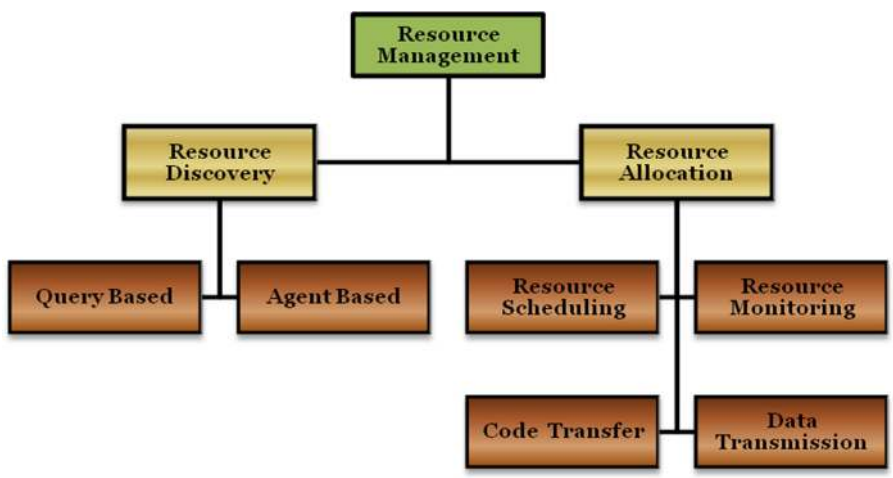

Fig. 4 Grid resource management taxonomy

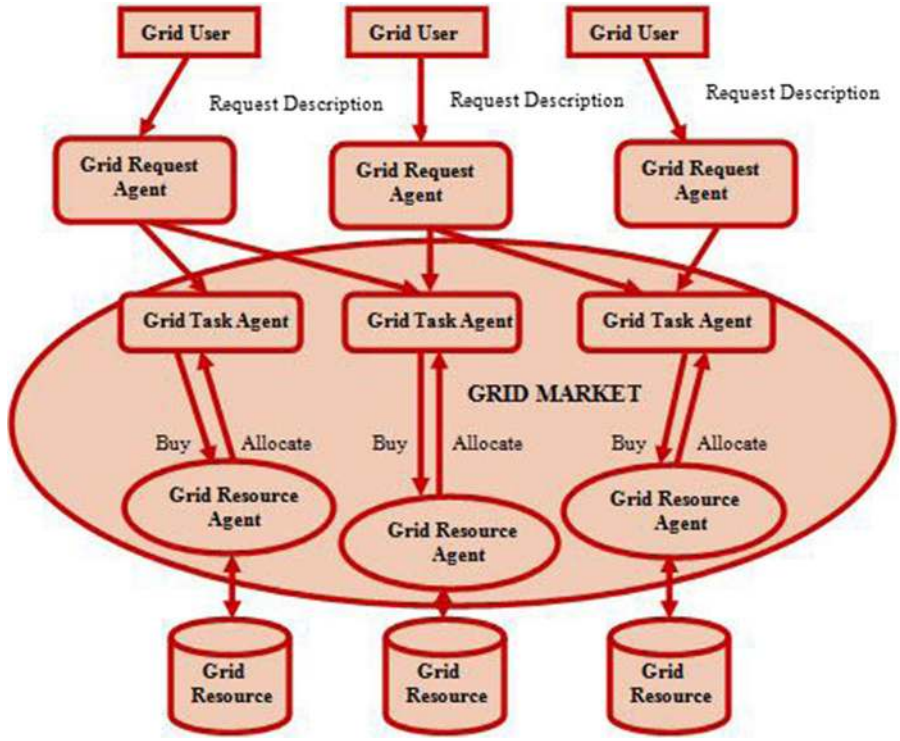

Fig. 5 Grid resource management system model

scalable, and fault tolerant than centralized and hierarchical organizations. Table 1 gives a comparison of the above mentioned organizations based on their common features.

To overcome the resource manageability, availability, and scalability issues of the aforementioned RM organizations, the *peer-to-peer* (P2P)-based Grid paradigm of resource management is used by the research community [90, 132]. The P2P-based Grid RMS follows a decentralized organization where peers (nodes) are arranged using an overlay network. Each peer in a P2P system works at the same time both as a server and a client. In a P2P-based Grid management system, if a peer knows the routing information to locate the other peer, then it communicates with it directly [72, 90]. Consequently, the information propagation strategy that the peers adopt is of primary importance. The P2P-based Grid RMS discovers resources by using the name lookup approach. The resources are located using different strategies like flooding, and indexing. Based on peers organization, the P2P-based Grid systems are classified as structured and unstructured [90, 132]. In *unstructured* P2P-based systems (e.g., Gnutella, KaZaA [89]), the flooding approach is used for locating resources where each peer propagates information about its local resources. When a user requests some resource, the peer matches the user query with the local resources. If a match is found then it returns this information to the peer who initiated the request following the same trajectory; otherwise the query is forwarded to the neighbor peers who adopt the same strategy. This

Table 1 Comparison of RM organizations

RM Organization	Control	Site autonomy	Fault tolerance	Scalability	Availability
Centralized	Centralized	Low	Low	Low	Low
Decentralized	Distributed	High	High	High	High
Hierarchical	Centralized and Hierarchical	Low	Medium	High	Medium

process continues the network flooding until the *time-to-live* (TTL) of the request expires. In *structured* P2P-based Grid systems (e.g., CAN, Chord, Pastry [72, 90], Tapestry, Cyclone, HiPeer [132]), the *distributed hash table* (DHT) approach is used for determining the exact location of a resource. DHT uses indexing service that is based on hash function. Structured P2P-based systems are more scalable than unstructured systems in terms of network traffic flooding. In unstructured systems the search query is received by peers that do not have the required resource, which results in flooding the entire network.

In a broad sense, resource manager is also responsible for resource monitoring, resource inventories, resource provisioning, autonomic capabilities, fault tolerance, and other service level activities [117]. The complex and dynamic nature of Grid technologies make sharing and discovery a challenging issue for RMS [117].

Some of the Grid features include different global administrative control over resources, existence of multiple Grid schedulers together with local schedulers, different performance characteristics of resources with respect to CPU especially, and storage system access [75, 76]. Current research on novel dynamic RM and job scheduling [81] add more scalability, robustness and fault tolerance to the system by balancing high workload due to the dynamic real-time availability of high computational power of various Grid resources. Apart from the aforementioned responsibilities, resource naming is also a complex process in distributed environments managed by Grid RMS because it affects the above discussed RM functions [53, 69, 96].

Three main approaches to name space organization identified by the author in [53, 69] are:

(a) *flat*,
(b) *hierarchical*, and
(c) *graph-based*.

The *flat* approach lacks scalability in the Grid system. The *hierarchical* approach follows down a traversing technique where a name is constructed in a top-down fashion. The *graph-based* approach follows the nodes connecting technique where resources are linked and a name is constructed from the links connecting the nodes. The three name space approaches are shown in Fig. 6.

3.1 Resource Discovery

Resource discovery is a basic service the Grid provides [7, 73]. Based on required characteristics, the resource discovery mechanism searches and returns the addresses of the resources that match with the provided descriptions. In a sense its functionality resembles a web search engine with search queries being the set of criteria for resource selection and links to web-pages being the analogous of matched resource addresses [73].

The *resource discovery* process selects a resource from a pool of resources by evaluating its capability based on the following trade-offs [117]:

1. Minimum execution cost
2. Shortest possible completion time of a Job
3. Minimum wastage of resource power

Discovering a resource by taking in to account the aforementioned parameters is a challenging task for maintaining Grid efficiency due to the resources dynamicity and heterogeneity [117]. Collecting and assessing resources information in heterogeneous environments is a complex task because resources are geographically dispersed, owned by different organizations, and restricted by different management policies and access rights [14]. Each resource discovery service makes use of a resource status database to carry out clients requests. This database is maintained and administered by network-wide information services.

Fig. 6 Grid resource naming taxonomy

The two main resources discovery techniques are *query-based* resource discovery and *agent-based* resource discovery [53]. In *query-based* resource discovery approach, for getting resource availability information, a query is made towards a database [53]. Alternatively, instead of querying a database, active code fragments can be sent to the various nodes for local process. This option is adopted by the *agent-based* resource discovery approach. Agents are autonomous, intelligent software entities that are executed on user's behalf and interact with their surroundings to carry on user requests [117]. This approach is distributed in nature and uses an underlying mobile code environment like Java.

The main difference between the aforementioned two approaches is that *agent-based* systems allow the agent to control the query process. The *agent-based* systems take discovery decisions based on internal logic rather than residing upon a fixed function query engine [69, 117].

Traditional resource discovery systems (e.g., Globus MDS [106, 126, 128], Condor Matchmaker [128], Portable Batch System [128]) use the exact lookup approach called matchmaking for locating resources. In such systems the resource matching is performed using a simple query language which compares strings or integers known as symmetric flat attributes based on exact matching [126, 128]. In this matching approach, the resource attribute-value pairs are matched with the request attributes specified as primitive constraint statements. The resource and request should use same attribute names and agree upon the attributes values. This resource discovery approach becomes less precise and inflexible as the number of attributes increases. This inflexibility denies incorporation of new characteristics. By precision we mean that how and how much the discovered resources are relevant to the request query. It may not necessary that the resources and requests should always use the same attributes specification which limits the system search space. Sometimes this approach of resource discovery misses capable resources. The resource miss penalty is explained by Amarnath et al. in [3] with the help of example. If a user requires a Linux operating system for its task completion then in case of non-availability of this resource, the Grid scheduler either puts the task in a wait queue or it may run on the systems that have Fedora or Unix-based operating systems because the scheduler has not the capability to establish a relationship between the two different operating systems. The flexibility and precision can be improved by adding semantic information to the resources description which locate only semantically related results [78]. Such an approach of resource discovery and selection is called asymmetric ontology-based semantic matching approach. This approach is capable of making semantic relationships between the user request and the resources' advertized information using an inference engine. It supports a context-based matching technique instead of a keywords-based approach that returns resources that are closely related to the request in case the exact match fails. The set of inference rules and vocabularies that help in matching user request to resource are defined by ontology [128] and the relationships in ontology are defined by semantics. In general the ontologies are easy to understand than flat attributes and they facilitate the integration and interchange of different type information. The Grid system that employs semantic techniques is called *semantic Grid* that is the addition of intelligent behavior to the conventional Grid paradigm.

The ontology-based resource discovery module consists of mainly three basic components defined in [128]:

(a) ontology,
(b) domain knowledge, and
(c) matchmaking rules.

The vocabulary captured from the ontology is used by the domain knowledge to obtain background information. This background knowledge and ontology is in turn used by the matchmaking rules for matching user requests to Grid resources. These components are capable of communicating and incorporating with different Grid middleware.

3.2 Information Propagation Strategies

In order to keep the resource description database updated resource availability information must be propagated [69]. A simple categorization of information propagation strategies is as follows:

(a) total awareness,
(b) neighborhood awareness, and
(c) distinctive awareness [88].

In *total awareness* the overall status information contained by each resource is broadcasted to all other

resources. *Neighborhood awareness* distributes status information of the nearest resources, whereas in *distinctive awareness* status information about distinct resources is propagated. The comparison of the aforementioned three information propagation strategies is given in Table 2.

3.3 Resource Allocation Problem

Resource allocation is one of the core services of Grid environment that achieves the basic Grid objective [48]. It is the process of distributing limited available resources among tasks based on some predefined rules. Wu et al. [140] represent the RA problem as a quadruple (R, A, X, O), where:

- R is the set of m available resources: $R = \{R_1, R_2, \ldots, R_i, \ldots, R_m\}$, $1 \leq i \leq m$. The resource can be any Grid resource summarized in Fig. 3.
- A is a set of n tasks (dependent, independent, single, or parallel) competing for the resources in R: $A = \{A_1, \ldots, A_j, \ldots A_n\}$, $1 \leq j \leq n$.
- $X_{m\times n} = \begin{bmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,n} \\ \vdots & & & \\ x_{m,1} & x_{m,2} & \ldots & x_{m,n} \end{bmatrix}_{m\times n}$

Each resource may accommodate more than one task. So Xi,j represents the portion of resource i allocated to task j.

- O is the objective function: $O = u(X_{m\times n}) - c(X_{m\times n})$ where $u(X_{m\times n})$ represents profit and $c(X_{m\times n})$ is the cost associated with a given allocation.

The generic RA problem can be posed as: find the values in matrix X, i.e., allocate portions of resources to tasks, so that the objective function $u(X_{m\times n}) - c(X_{m\times n})$ is maximized, subject to various validity and availability constraints on the assigned values in matrix X..

Depending on the problem setting, the allocated portion of resource to task, $X_{i,j}$ can be termed as continuous or discrete variable. For example, $X_{i,j}$ might be a continuous variable that take continuous and non-negative value if the resource under question is divisible like network bandwidth, processor time; and a discrete that take non-negative integer value if the resource is non-divisible like a person, an airplane or a car.

3.4 Energy - Aware Resource Allocation

The global Grid provides massive power for executing data- and compute-intensive scientific applications, but the issue of high energy consumption makes Grid platforms impractical for implementation [84, 85]. Significant research exists on efficiency enhancement of clusters at processor level [29, 112], virtualization based resource managers level [134, 135], and cluster resource managers level [61, 137]; however only a small portion of it deals with improving energy efficiency of HPC platforms. In the area of global Grids, many meta-schedulers are in use. GridWay [45] is a service oriented Grid system capable of adjusting itself to the changing environment to meet the needs of Grid applications for heterogeneous resources. It uses the *first come-first serve* (FCFS) scheduling policy. Moab uses FCFS batch scheduler with backfilling policy [11, 31]. In Moab, the priority of each task is calculated as a weighted sum of several factors (e.g., queues time, task arrival time etc) specified by the system administrator. Condor-G [37] is a user-level middleware scheduler working as a component of Condor RMS [143] providing an interface for heterogeneous batch systems. Condor system is a collection of heterogeneous resources providing a high throughput computing environment for computation-intensive tasks. Condor-G is built on Globus [45] toolkit that is a Grid middleware providing services to the Grid applications for accessing distributed resources securely

Table 2 Comparison of resource information propagation strategies

Strategy	Communication overhead	Scalability	Efficiency in homogenous Grid	Efficiency in heterogeneous Grid
Total Awareness	Very High	Low	Medium	High
Neighborhood Awareness	Low	Very High	High	Medium
Distinctive Awareness	Low	High	High	High

across multiple administrative domains. Condor-G uses either FCFS or matchmaking with priority sort [40] as scheduling policies. These meta-schedulers use parameters like job completion time and load balancing as performance measures, ignoring the issue of huge energy consumption. Significant research efforts on power-aware RA have been made in the context of clusters and datacenters.

Two primary methods are commonly used; switching off un-utilized cluster parts [13, 75, 76, 86, 101, 137]; or using *voltage and frequency scaling* (VFS) technique to compensate CPU processing speed [21, 41–43, 61, 86, 95, 129, 134, 135, 137]. According to Meisner et al. [95] it is hard to adopt a power on/off strategy in case of overloading but it is ideal and simple to switch off un-utilized systems. Therefore, VFS-enabled systems perform better in energy saving in such environments. Using VFS at CPU level can save a lot of energy by scaling down the CPU frequency either manually [36, 86] or dynamically (DVFS) [20, 30, 41–43, 55–57, 62, 80] by means of processor technologies like Intel's SpeedStep [58]. Thus the Grid meta-scheduler, along with other factors, should decide to assign jobs to resource sites at the time in which the jobs can be executed at minimum CPU frequency level to save power consumption. If the job deadline cannot be met by adopting such a scenario, then the meta-scheduler should scale up CPU frequency to the next level and then again search for a free time slot to execute the job. The basic Grid energy efficiency techniques are categorized in Fig. 7.

A prediction algorithm was proposed by Orgerie et al. in [101] which aggregates the workload and switches off the un-utilized CPUs thereby reducing the processing power consumption in large Grids such as Grid5000 [101]. In [38] the authors pointed out that just focusing on processor power consumption is not always sufficient. In order to minimize the power consumption of the entire Grid due to heterogeneity and a more holistic approach involving energy-aware changes to RA mechanisms is necessary [86]. In fact, although significant research exists on energy optimization within a cluster's or a datacenter's boundaries, the issue of power-aware RA in a distributed environment with multiple geographically dispersed resource sites remains largely open.

3.5 Resource Monitoring

Resource monitoring is the process of collecting information about Grid resources' status. This information is updated periodically by resource providers through resource update queries. The Grid schedulers retrieve resources' information by querying *Grid information service* (GIS) for matching the available resources with the user's requirements. The GIS is responsible for indexing the resources. Below we summarize some of the known Grid information systems.

3.5.1 WebMDS

WebMDS is a type of decentralized Grid information service infrastructure called *monitoring and discovery service* (MDS) that assists the users in monitoring and discovering resources and services on the Grid [2]. Every MDS provides two basic services to the Grid users; *Grid resource information service* (GRIS) and *Grid index information service* (GIIS) [78, 147]. The GRIS resides on each resource that keeps all the status, configuration and capability information of that resource and forwards it to the user who queries such information about that resource. The GIIS is a directory service that assists GRIS in caching the resources' information by using indexing and searching mecha-

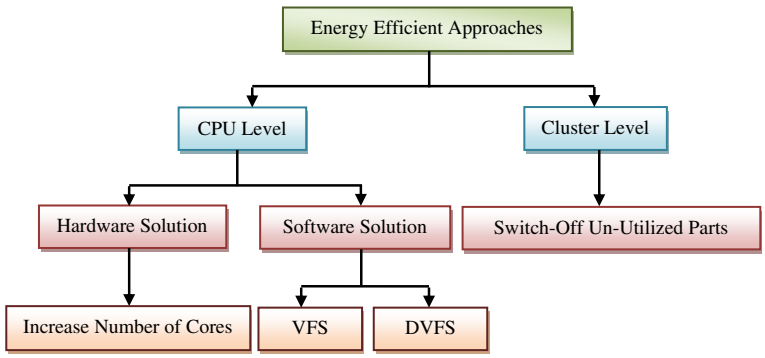

Fig. 7 Grid energy efficient techniques taxonomy

nisms. For resolving interoperability issues between GRIS and GIIS systems, the GRIS uses an inquiry protocol called *Grid resource inquiry protocol* (GRIP) that provides discovery and enquiry functions along with a registration protocol called *Grid resource registration protocol* (GRRP). The MDS supports both static and dynamic data about Grid resources. Web-MDS provides a web browser interface to the users to view monitoring information status of a resource, service registration, query, discovery, and locating the resources [2, 139].

3.5.2 LDAP

The *lightweight directory access protocol* (LDAP) is used as a data model and representation, query language, and protocol by MDS [145]. As a data model, it gives hierarchical structure for representing entities as LDAP objects. The entities are represented as attribute-value pairs organized in a hierarchical form [25] called *directory information tree* (DIT). It is also used as a open and platform independent directory protocol that helps in accessing resource directory information. The LDAP is used for accessing a list of resources of a particular class or having particular property. It is also used to populate resource information like the IP number of a resource, and its total amount of memory. In Grid context, the MDS uses LDAP mainly in its three phases [Sri]; initialization, population, and query phase. In initialization phase, LDAP is used to provide tools that help in defining, referencing, organizing, and modifying resources' information. In population phase, it offers a framework for populating the service directory with information. This feature provides portability between different platforms because LDAP is a platform-independent protocol. In query phase, it supports query-reply exchange. It also helps in communication with the server.

3.5.3 MonALISA

The *monitoring agents in a large integrated services architecture* (MonALISA) is an agent-based distributed monitoring service using JINI and Web Services technologies for hosts and networks in large scale distributed services [100, 145]. It uses modules that collaborate and cooperate with each other in performing monitoring tasks. It also provides the facility to integrate other monitoring and queuing systems. The MonALISA is based on one server station per site and number of JINI registries called *lookup discovery services*. Each discovery service may be registered with other discovery services. The agent-based services are controlled by the server station that schedules, restarts if necessary, and monitors functions of these agents. Each agent-based service can register to a number of discovery services. The server station collects resource status information from other monitoring services available locally using various modules. This information is stored in local repositories or embedded in external storages and properly indexed for the users. This information is provided to the users on demand. The users can directly connect with a service for receiving monitoring information. The users can configure any service or module remotely and dynamically by using an administration graphical user interface GUI [100].

3.5.4 Remos

The *resource monitoring system* (Remos) is used for providing a consistent interface to network-aware applications for dynamically monitoring performance of local area networks (LANs) and wide area networks (WANs) that are independent of the network technicalities [DeG97]. It also provides resource information to network-aware Grid applications. Its architecture is composed of three basic components; collector, modeler, and predictor. The *collector* collects and consolidates information required by the user applications by using different methods, e.g., implementation of sensors. Several collectors may be employed in a hierarchical fashion connected in a predefined way for providing scalability and accommodating heterogeneity of various networks. The upper level collectors in the hierarchy are called master or global collectors and the lower level collectors as local collectors. The master collectors measure the performance of the overall network that connects various LANS, while each local collector obtains performance information about its LAN. Remos interface supports query-based requests. The queries are used to obtain information about specific set of resources on the network. The master collector accepts queries from modelers and forwards it to the local collectors for execution. The processed result is then forwarded to the *modeler.* Remos provides abstraction in the sense that when a modeler

requests some information, then the master collector collects this information from different local collectors and forwards it to the modeler that is completely unaware of the collection process [145]. The modeler interacts with the user application through interface and responds to its queries. The *predictor* predicts the future performance behavior on the basis of previous measurement history.

3.5.5 GridICE

GridICE was developed to assist Grid administrators through a web front end to provide status and utilization information of the resources [106]. It has a centralized architecture that makes it less scalable. The main server consists of Nagios service [145] that monitors Grid resources regularly by forwarding queries to get their status and utilization information. The MDS plugin is used for issuing periodic queries to the index information servers and service providers. The information collected by Nagios from MDS daemon in response of these queries is stored in a database which is used for calculating aggregate statistics like total amount of memory per site, total utilization of resources per site, etc. The users use a web-based front end for accessing the monitoring system. GridICE operates in a five layered architecture which are measurement, publisher, data collector, detection & data analyzer, and presentation [4]. The first layer consists of the *measurement* service which search resources according to some metrics such as total memory usage, total processor usage or total time usage. The second layer furnishes the *publisher* service where the gathered data is provided to all the users in Grid system. The third layer provides the *data collector* service which collects historical monitoring data. The fourth layer offers two types of services:

(a) *detection and notification* service about resources and
(b) *data analysis* service.

The fifth layer presents monitoring information to the user through a web based graphical interface.

3.5.6 Autopilot

Autopilot is an adaptive infrastructure that gives the ability to the applications to adapt theirselves to the dynamically changing environments such as Grid. The autopilot functionality can be used in sensors, actuators, distributed name servers, and clients [110]. The sensors and actuators are used for remote reading and writing purposes. Sensors are the producers which sense the application specific events and generate description of resource demands. On the other hand, actuators are used to configure resource management policies like file cache policy, buffer size etc. The sensors, actuators, and clients coordinate with one another through a component called autopilot manager. The components (sensors and actuators) have properties like name, location, identifier, IP address, and other attribute-value pairs which may be user defined [145]. These properties are defined at creation time of these components. The components register their properties with the registry service or component manager after creation. The users search a component of their interest in registry by using these properties and subscribe theirselves to that component. The manager in response to the user queries provides the starting points of the sensors or actuators that can fulfill the user demands. The starting points are then used by remote clients for connection with the sensors or actuators. The remote user applications access the autopilot system through an API provided at the resource level.

3.5.7 MapCenter

The MapCenter is a Grid information and monitoring system that is used to track resource availability problems [12, 145]. The availability information about Grid resources and their services is gathered and provided through a web interface to the remote users in response of their queries for monitoring and discovery services. The MapCenter supports automatic discovery for the services like TCP, UDP, and HTTP [106]. This system uses a hierarchical view of resources' status information. It deals only with the status information (e.g., availability) and does not keep the configuration or utilization information of the resources. The MapCenter [12] consists of the monitoring, data store, and presentation layers. The *monitoring* layer uses sensors for monitoring Grid resources. The *data store* layer accumulates the internal structures information, while the *presentation* layer provides the facility to generate and visualize different views of the Grid.

3.5.8 RGMA

The *relational Grid monitoring architecture* (RGMA) [147] is a relational system that uses a relational model to combine both information and monitoring services of the Grid system. The *Grid monitoring architecture* (GMA) is mainly composed of five basic components [106]:

(a) *producers* are the data sources (e.g., sensors) that register theirselves to the registry, with the registration being refreshed at specific time intervals by sending heartbeats,
(b) *consumers* (e.g., resource broker) that request information about producers from the registry service,
(c) *registry or directory service* that contains producers and consumers information and indexes to the producers,
(d) *republisher* that exhibits the characteristics of both consumers and producers, and
(e) *schema repository* that accumulates the schema details of various producers and events.

The producers and republishers can be collectively called as *publishers*. Cooke et al. [24] describe the operation of GMA components in the following way. The producers register theirselves with the registry and show what information they can offer to the consumers. This information is then accessed by the consumers. Using this information the consumers know which producers can provide the data relevant to the request query. After specifying the target producers, the consumers directly communicate with them for data provision. MDS is the most prominent system that implements RGMA. RGMA shows the relational view of the GMA components. It provides the facility to extract information of VO resources just like they are stored in a centralized relational database management system (RDBMS).

3.5.9 Hawkeye

Hawkeye [145, 147] is a monitoring and management tool used for the automation of problem detection such as resource failure, insufficient cache, high link usage etc. The *Hawkeye* issues warnings in the aforementioned situations to the Grid and other distributed systems' resources. The architecture of hawkeye is composed of four basic components organized in a hierarchical form [147]:

(a) pool,
(b) manager,
(c) monitoring agent, and
(d) module.

Pool is a collection of computers, with the one of them being used as a manager that connects the rest computers called monitoring agents. The monitoring agents register theirselves with the manager and provide monitoring information to it. The *manager* is a master computer system that collects all this information and stores it in a database which can be accessed through web interfaces. All the user queries about resources status information are forwarded to the manager. The monitoring agents are connected with *modules* that are sensors for advertising the resource information. The modules send updated data to the agents after each minute interval.

4 Resource Allocation Mechanisms

RA mechanisms play an important role in allocating the most appropriate resources to applications. The mechanisms perform the allocation of tasks to the resources in order to ensure QoS to the application according to the user requirements [48]. Sometimes RA mechanisms adopt dynamicity whereby resources are allocated as soon as they are discovered. Such mechanisms are called *dynamic RA mechanisms* and are considered more efficient than the static ones. Another assumption is that RA mechanisms should be designed in such a way to avoid underutilization of resources.

As indicated in Fig. 4, RA mechanisms provide two basic Grid services:

(a) resource monitoring, and
(b) resource scheduling.

Resource monitoring regularly monitors resource performance, capability, usage and future reservations. These resources include processors, disks, memories, and channel bandwidths [48]. The information is then retrieved by the scheduler that decides on the allocation of the application to the underlying resources. Some of the widely used Grid resource discovery and allocation mechanisms based on RMS organizations

are broadly categorized as centralized, distributed, or hybrid as shown in Fig. 8. Table 3 gives a comparison of these mechanisms based on their common features.

4.1 Parameter-Based Approach

A new *parameter-based* approach called Grid potential was proposed in [88] that summarizes the computing power of different Grid resources in a large network. This approach is based on the operating rate of a node (CPU speed, FLOP rating, sustained memory access rate, sustained disk access rate). A data dissemination algorithm was also proposed that is based on the idea of "*swamping algorithm*" [7]. When a resource status information dissemination message comes into a specific node, the message is first validated before processing [88]. The validation process depends on the aforementioned information propagation strategies (*total awareness, neighbourhood awareness and distinctive awareness*). The total awareness strategy does not discard any of the incoming dissemination messages. The neighbourhood awareness permits the incoming message only if the distance from the source to destination node falls within the range of permissible limit. Otherwise, the message is discarded. In distinctive awareness,

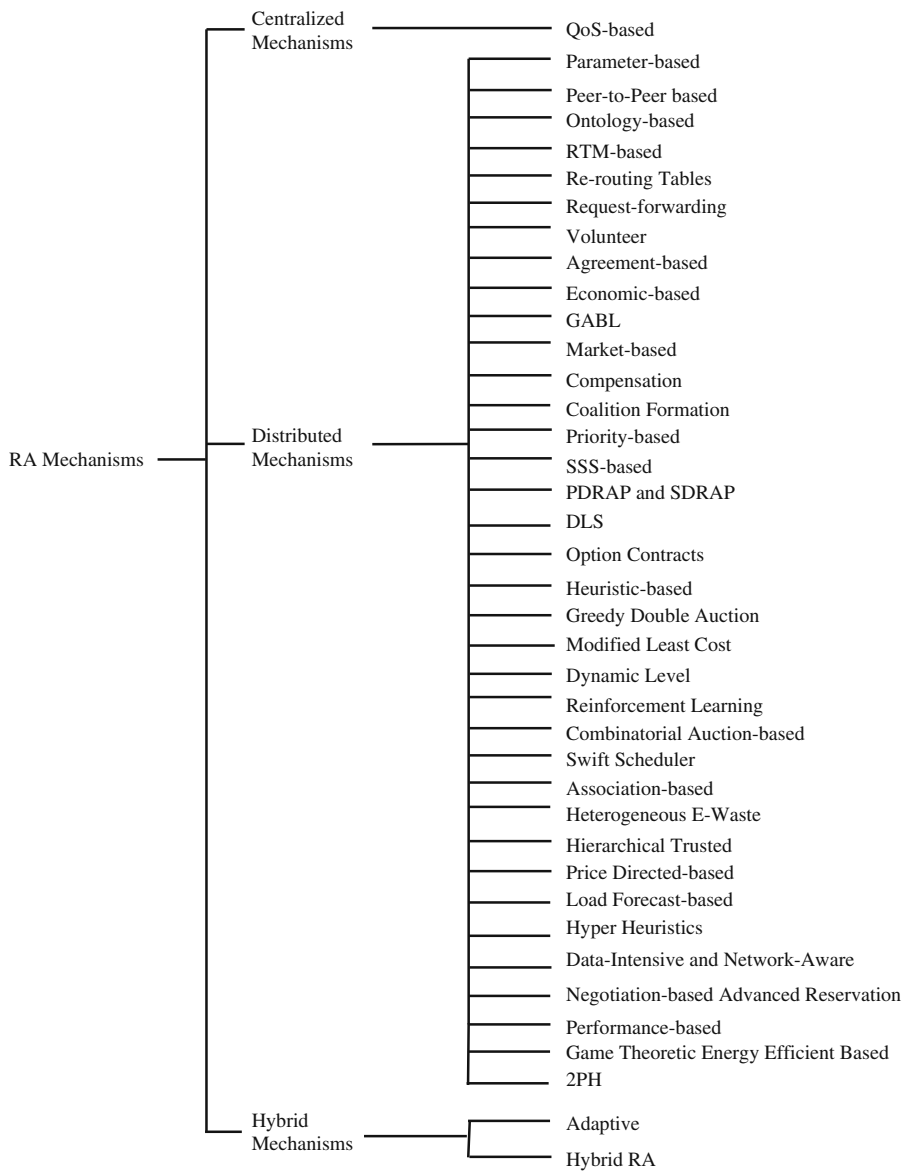

Fig. 8 Resource allocation mechanisms taxonomy

Table 3 Comparison of RA mechanisms

RA Mechanism/ Approach	Searching Mechanism	Application Type	Time Complexity	Optimal	RAS Taxonomy	Operational Environment	Objective Function	Citations to the Mechanisms
. Parameter-based	Dissemination Algorithm	Independent Divisible	Ω (n)	Yes	Distributed	Distributed Computer Networks	Transmission overhead, resource status dissemination	51
Peer-to-Peer	Swamping Algorithm	Atomic Tasks	O (log n)	Yes	Decentralized	Distributed Networks	Network resources	345
Ontology-based	Matchmaking,	Atomic and Gang Matchmaking	O (log log n) Independent Divisible	Near	Distributed Optimal	Heterogeneous	Resource advertisement	38
QoS-based	DIAR Environment	Dependent Workflow	Ω ($\log_2 n$)	NA	Centralized	Multimedia Environment	Processor runtime, storage capacity, network bandwidth	17
RTM-based	SDRT, Routing Algorithm	Dependent Workflow and Independent Divisible	⊖(n)	Yes	Distributed	Distributed Networks	Response time	71
Re-routing tables	Min-Distance Algorithm	Dependent Workflow and Independent Divisible	NA	Yes	Dynamic Distributed	Distributed Networks	Resources, Services	14
Request-Forwarding	Request Forwarding Algorithm	Independent Divisible	⊖(n)	No	Distributed	Distributed Networks	Computing resources	345
Volunteer RA	Volunteer Pooling Algorithm	Independent Divisible	NA	Near Optimal	Distributed	Distributed Computer Networks	Processing power	10
Agreement-based	FairShare Algorithm	Independent Divisible	NA	Yes	Distributed	Distributed Networks	Processing power	10
Economic RA	RFPSA Algorithm	Independent Divisible	NA	Yes	Distributed	Distributed Networks	Processing power	694
GABL	GABL Algorithm	Independent Divisible	O (bm^2)	Yes	Distributed	2D-Mesh Networks	Processor	16

Table 3 (continued)

RA Mechanism/ Approach	Searching Mechanism	Application Type	Time Complexity	Optimal	RAS Taxonomy	Operational Environment	Objective Function	Citations to the Mechanisms
Market mechanism	DAI method	Independent Divisible	NA	NA	Distributed	Distributed Networks	Resource price	21
Compensation mechanism	DAI method	Independent Divisible	NA	NA	Distributed	Distributed Networks	Computation cost	21
Coalition Formation	DAI method	Independent Divisible	NA	NA	Distributed	Distributed Networks	Computation and communication cost	21
Priority-based	Variant	Dependent Workflow and Independent Divisible	NP-hard	Near Optimal	Distributed Computer Networks	Distributed	Computation power, bandwidth, QoS requirements	18
SSS-based	State Space Search, Heuristic Algorithms	Independent Divisible	O (m^3)	Optimal and Sub-optimal	Distributed	Distributed Computers	Sum of execution and communication cost of overall processor	6
PDRAP and SDRAP	PDRAP, SDRAP	Atomic and Independent Divisible	O((n/p+$\log_2$p) N^2)	Yes	Distributed	Parallel Hypercube Machine	Indivisible resources	2
Dynamic RA	Best-fit & Process Migration	Independent Divisible	NA	Yes	Dynamic	Distributed Networks	CPU, Memory, Disk	243
RA by means of option contracts	Mathematical Model	Independent Divisible	NA	Yes	Distributed	Distributed Computing Resources	Resource prices, Services times	8
Heuristics-based	Heuristics-based	Dependent Workflow	Polynomial Time	NA	Distributed	Distributed Data Stream Processing	Processor utilization	3
Greedy Double Auction Mechanism	GDAM	Atomic and Independent Divisible	NA	NA	NA	Heterogeneous Computing Resources	Resource price	2
Modified Least Cost Method	MLCM	Independent Divisible	NA	Near Optimal	NA	Heterogeneous Networks	Computational cost	5

Table 3 (continued)

RA Mechanism/ Approach	Searching Mechanism	Application Type	Time Complexity	Optimal	RAS Taxonomy	Operational Environment	Objective Function	Citations to the Mechanisms
Dynamic Level Scheduling	DLS	Dependent Time	Polynomial Time	NA	Distributed	Distributed Heterogeneous Nodes	Communication overhead and Computational time	750
Reinforcement learning	Minimalist Decentralized Algorithm	Atomic and Independent	NA Divisible	NA	Distributed	Heterogeneous Networks	Efficient tasks distribution	99
Combinatorial Auction-Based RA	Combinatorial Auction-Based RA	Independent Divisible	NA	NA	Distributed	Distributed Computing Resources	Economic efficiency, Revenue maximization and System performance	83
Swift Scheduler	Heuristic algorithm and Shortest	Independent Divisible job First	NA	NA	Distributed	Distributed Systems	Task wait time and computational time	9
Hybrid RA method	Least Cost method and Divisible Load Theory	Independent Divisible	NA	Yes	Dynamic Distributed	Distributed Heterogeneous Networks	Computational cost	3
Association-based RA	Association-based Algorithm	Atomic and Independent Divisible	NA	NA	Distributed	Decentralized based distributed clusters	Computational cost	91
Heterogeneous E-Waste RA	HER	Atomic and Independent Divisible	NA	NA	Distributed	Distributed E-waste computing resources	Computational power	5
Hierarchical Trusted RA	HTRA mechanism	Independent Divisible	NA	NA	Distributed	Dynamic Distributed Networks	Execution delay, Success ratio, Energy consumption, Workload and throughput	25

Table 3 (continued)

RA Mechanism/ Approach	Searching Mechanism	Application Type	Time Complexity	Optimal	RAS Taxonomy	Operational Environment	Objective Function	Citations to the Mechanisms
Price directed-based RA	Resource Agents	Independent Divisible	NA	NA	Distributed	TCP/IP Network Model Supported by JAVASIM	Minimization of task completion time and cost of the used resources for task execution	73
Load forecast-based RA	Grid Information	Independent Divisible Service (GIS)	NA	Yes	Distributed	Computational Grid	Optimization of user's task execution time within the budget constraints	2
Hyper-Heuristic Approach	NA	Independent Divisible	NA	Yes	NA	Test cases	Efficient mapping of jobs to available computing nodes, makespan minimization	9
Adaptive Grid Scheduling	Mobile Agents	Dependent Workflow	NA	NA	Decentralized/ Hybrid	Distributed Systems	Makespan and transmission communication time minimization	4
Data-Intensive and Network Aware Scheduling	DIANA algorithm	Independent Divisible	NA	Yes	NA	Peer to Peer Network	Waiting time, Data transfer time, tasks execution time minimization	46
Negotiation-Based Advanced Reservation	Negotiation-based HEFT	Dependent Workflow	NA	Yes	Distributed	Computational Grid	Makespan minimization	6
Performance-based Scheduling Strategies	SO,DO, SO-AS,DO-AS	Independent	NA	NA	Distributed	Federated Grids	Makespan minimization, Grid performance	6

Table 3 (continued)

RA Mechanism/ Approach	Searching Mechanism	Application Type	Time Complexity	Optimal	RAS Taxonomy	Operational Environment	Objective Function	Citations to the Mechanisms
Game Theoretical Energy Efficient	NBS-PATA	Independent	O(nmlog(m))	Pareto-Optimal	Distributed	Computational Grid	Makespan, Power minimization	5
2PH	H2LL and	Independent Min-Min	NA	NA	Distributed	Computational Grid	Energy Efficiency	9

the incoming message validation takes place on the basis of computational power of the local and remote Grids. This approach could lessen the high communication overhead during resource discovery process and helps in reducing the network congestion. It is observed that the disseminated message complexity in neighbourhood and distinctive awareness approaches is the same, but the total awareness approach exhibits higher complexity than the other two counterparts, with the reason being that the dissemination message is forwarded to all the nodes in the network.

4.2 Peer-to-Peer Approach

Peer-to-Peer (P2P) RA approach was proposed by Adriana Lamnitchi et al. for large distributed resources in [73]. The proposed approach is decentralized and provides efficient resource discovery results with minimum administrative overhead. All the responsibilities are uniformly divided among all nodes known as peers. The peer is a resource that works as a server and a client at the same time. Information about computing resources and services are directly exchanged between systems. A "*unified peer-to-peer database framework* (UPDF)" for large highly distributed systems was proposed in [74], which can be viewed as a Peer-to-Peer database framework for general purpose query support. This database can be viewed as up-to-date global information. The aforementioned approach uses a graph-theoretic approach for scalability and manageability purposes. The presented P2P approach reduces local processing because if a resource description at a site does not match the request query, then the *time-to-live* (TTL) attribute's value of the request message is decremented and forwarded to the next resource. This process of forwarding continues until the TTL becomes zero. This approach is useful in highly scalable environments because the information about the resources and their capabilities are replicated that increase the availability of information in a dynamic system. But maintaining integrity of information is tiresome if a resource leaves a network more frequently.

4.3 Ontology-Based Approach

Ontology specifies the basic layout of a resource and provides interoperability among the resources in heterogeneous environments. A Grid based semantic

service discovery framework was proposed in [83]. A matchmaking mechanism based on ontology can discover distributed resources in an efficient way, such that very close matches of resources to requests are returned due to ontology mapping process. In matchmaking, the service provider registers the service details in a registry database. The user requests service directory for resource provisioning. The matchmaker retrieves all the matched services and the requester then selects the best resource based on the required attributes; subsequently the task is submitted to that resource.

The *ontology-based* approach performs well when Grid environment is highly heterogeneous. A limitation of this approach is that during resource discovery process the resources that do not match to the user request are discarded that reduce the pool of capable resources.

4.4 QoS-Based Approach

QoS-based approach [46] was first proposed for multimedia environments. The approach discovers the occasionally available resources, which are not available all the time using "*discovering intermittently available resources* (DIAR) algorithm". DIAR algorithm uses a graph-theoretic approach. The approach provides the resource discovery and scheduling mechanism in such a way that the required data stream flows continuously to the user even if the servers are not available by provisioning QoS to each user request. Different QoS parameters are processor utilization, storage capacity, and bandwidth. The incoming user request is composed of five parameters; the requested video object identifier, the starting time of the request, the finishing time of the request, the request type, and the QoS. The request type describes the nature of the needed response that can be immediate or delayed. The QoS parameter of the request specifies the required resources and the total time for which these resources are required. The implemented results show that the multiple server strategy for request execution outperforms the single server strategy because at anytime more servers are available for processing a user request. But if the multiple servers cannot give a sufficient continuous time for executing the request or the network bandwidth is limited, then the multiple server strategy is not advisable. The proposed QoS-based approach uses a centralized broker that becomes a bottleneck when the system scalability increases.

4.5 Routing Transferring Model-Based (RTM) Approach

This model was proposed by Li et al. [82]. The model has three basic components:

(a) *resource-requester*,
(b) *resource-router*, and
(c) *resource-provider*.

A routing table stores all the related information (e.g., distance, direction) about resources that are provided by the resource provider. The routing table is updated periodically by accumulating new resource information. When a requester sends a message to the router for requesting some resource, then the router first scans the routing table for the addresses of related resources and chooses the shortest path by which a resource is near to the router. In the sequel the router forwards the request to the concerned service provider or to another router. If the distance of a resource from the router is 1, it means that the service provider is the neighbour of the router. A request potentially passes through multiple routers until it reaches a proper service provider that fulfils its requirements. If multiple providers provide resources that can fulfil the user request, then the proposed RTM strategy selects a provider that is nearest to the requester. The complexity of the aforementioned approach depends on the topology and distribution of resources. The performance of the proposed technique is determined only from the distribution of the resources in the network when the network topology is definite. The limitation of this approach is that if numerous resources exist, then scanning large routing tables is a slow process. Figure 9 shows the RA process of the RTM approach described by the authors in [82]. In the figure, the i^{th} router is represented by R_i, the j^{th} provider by P_j, and the n^{th} resource by r_n. According to the figure, the user needs a resource located at provider P_3. The user sends a request to R_1 which forwards it to R_3 according to its routing table, R_1 knows that forwarding the request through R_3, the required resource is discovered. Therefore, R_1 and R_3 forward the request until it reaches P_3. In Fig. 9, the bold arrows show the flow of the request. The authors examined their proposed system using the Vega Grid

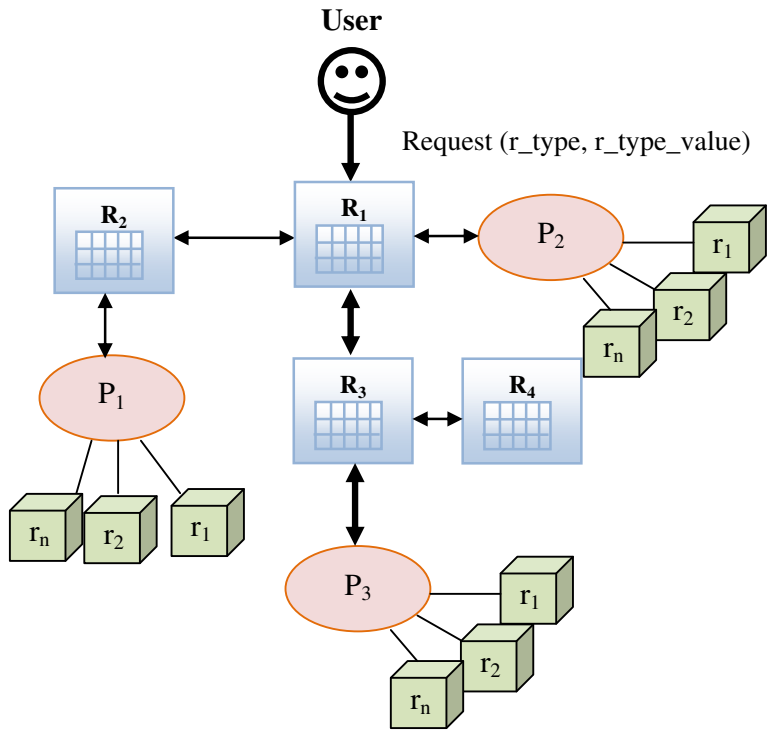

Fig. 9 Resource locating process by routing-transferring model

project and claim that the replication of resources on multiple locations can decrease resource discovery time.

4.6 Re-Routing Tables Mechanism in Dynamic Grids

The resources in dynamic Grids get an offline status when disconnected from the Grid. The request for the aforementioned resources must then be re-routed to be satisfied. The approach investigates the Grid resources in an efficient way similar to routing table's mechanism by considering the dynamic nature of a Grid system. In case of offline resource event, alternative resources are discovered somewhere in the system that can meet the request. After discovering the alternative resources, the routing tables are updated by re-computing the distances from the resource to all relevant routers to achieve the correct re-routing of the request in the network. This *re-routing mechanism* guarantees the discovery of resource in minimum number of hops [52]. The proposed re-routing tables mechanism is useful in dynamic environments where resources leave and join Grid dynamically.

4.7 Request-Forwarding Approach

Request-forwarding is a technique that decides the node to which the user request should be forwarded [73]. The user sends a request to a known node that returns the description of a resource if the request is matched with the local resource. If no match is found locally, the request is forwarded to another node which forwards it further until *time-to-live* (TTL) of the request expires or a match is found. The aforementioned approach is based on the request-forwarding technique.

Lamnitchi et al. in [74] identified four request-forwarding approaches:

(a) *random-walk*,
(b) *learning-based*,
(c) *best-neighbour*, and
(d) *hybrid learning-based* approach.

In *random-walk* approach, the node to forward the request is selected randomly. In *learning-based* approach, a request is forwarded on the basis of previous knowledge. If no node has answered the similar request before, then the request is forwarded

to the randomly chosen node. According to *best-neighbour* approach, every peer providing at least one resource is analyzed. For request forwarding a node is chosen that has answered the highest number of requests and it is called best-neighbour [73, 74]. The *hybrid learning-based* approach accumulates the features of both learning-based and best-neighbour approach. This approach is identical to learning-based approach with the difference that if no similar node is found, then the user request is forwarded to the best-neighbour while in learning-based approach, it is forwarded to a randomly chosen node.

The request forwarding efficiency depends greatly on the resource distribution strategy. If the resources are uniformly distributed then it is more probable that the request will find the corresponding resource in a minimum time. But unbalanced distribution of resources does not affect the efficiency of the best-neighbour approach. In all the aforementioned four request forwarding approaches, the random-walk has advantage that it does not need additional storage for recording the request answering history of the nodes.

4.8 Volunteer Resource Allocation

The *volunteer RA* mechanism works by having all the volunteers donating their idle resources especially processing power (CPU) towards the completion of some task without the demand of any reward. The aforementioned mechanism provides remarkable amount of computing power. The volunteer RA algorithm that is also known as volunteer pooling was proposed in [70]. This algorithm allocates resources to the tasks with a polling procedure. The resource polling steps are shown pictorially in Fig. 10 [70]. The users send jobs to broker at a constant rate until all the jobs are finished. The broker's responsibility is to poll all the resources that can complete the jobs. Upon receiving the response from the resource, the broker distributes jobs to that resource for execution. A resource is a combination of machines. After receiving jobs from the broker, the local scheduler running on each resource assigns the jobs to any idle machine based on some scheduling criteria and the remaining unprocessed jobs are stored in a queue. The jobs from the queue are then accessed on a FIFO basis.

If a volunteer machine fails down, the job is either re-assigned or is placed on the first position in a queue. The volunteer-polling mechanism performs well in terms of Grid utilization when there is a single user, but the completion times for the users requests increase when number of users increase.

4.9 Agreement-Based Resource Allocation

Agreement-based RA mechanism is adopted by organization's privately owned Grids or between collaborating institutions such as educational institutes where

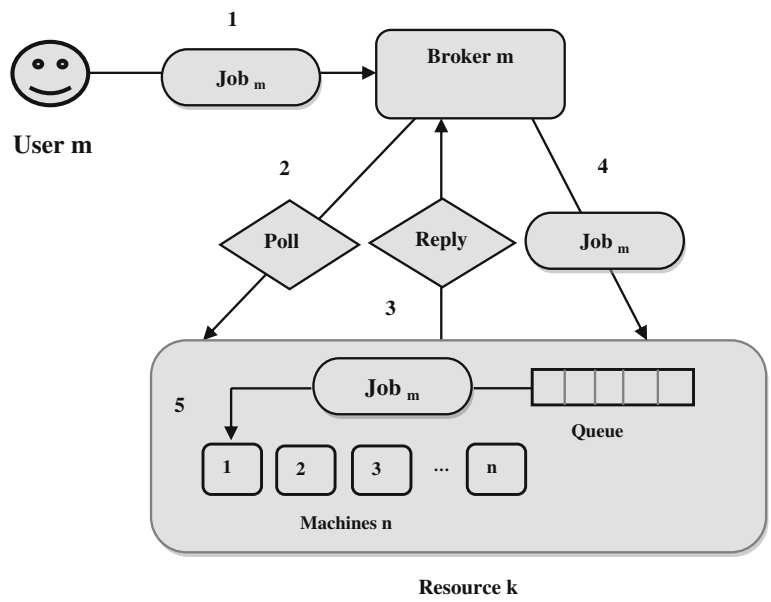

Fig. 10 Volunteer pooling RA steps [70]

there exists high and long-term collaborations. It takes two general forms:

(a) *policy-based* RA, and
(b) *service level-based* RA.

Policy-based RA sub-divides the resources. This follows the concept of VO where the pool of resources is allocated to each VO and then a VO can sub-divide the resources among its members. *Service level-based* RA was proposed by Andreozzi et al. in 2005. In this approach each user provides requirement details showing their expectation of service quality. The broker then takes decision on this SLA and allocates the matched resources [70].

In *agreement-based* strategy, each user sends his jobs towards a specified broker. The broker then randomly selects and assigns the set of jobs to a resource. After allocating a resource, the respective job is then assigned a priority based on the statistics provided by Grid statistics component. Based on this priority, the jobs are placed in a priority queue from where they are executed on machines using a FIFO manner. These steps are shown in Fig. 11 [70].

4.10 Economic Resource Allocation

Economic mechanism relays on bartering or prices [17]. In a *bartering-based* system, all members have resources and deal resources by exchanges; while in a *price-based* system, price is concerned with resources [17]. The *economic RA* is adopted where resource negotiation follows an economic mechanism. In large distributed systems, auction is used to find the best options of resources for executing the jobs. The four basic types of auction protocols are the English, Dutch, Sealed-Bid, and Vickrey [70]. In economic mechanisms the resource selection is not only based on the user optimization requirements (like deadlines) but also on the budget constraints the user willing to pay as well as the resource cost that is set by the owner. In this mechanism, the scheduler discusses the service price with the service provider, with the resource that can execute user application within his budget constraints being selected for execution. The process of discussing the service access cost with the owner of the resource using some economic model is called resource trading. The resource trading is cost efficient if the scheduler selects local resources against remote ones that can fulfil user requirements. The economic mechanism performs well when the user relaxes the deadlines of his application.

4.11 Greedy Available-Busy-List Strategy

A non-contiguous, parallel processor allocation strategy for 2D Mesh called *greedy available-busy-list* (GABL) was proposed by Bani-Mohammad et al. in [98]. In GABL strategy, processor is considered as a main resource. The suggested strategy has both the desired features of contiguous and non-contiguous

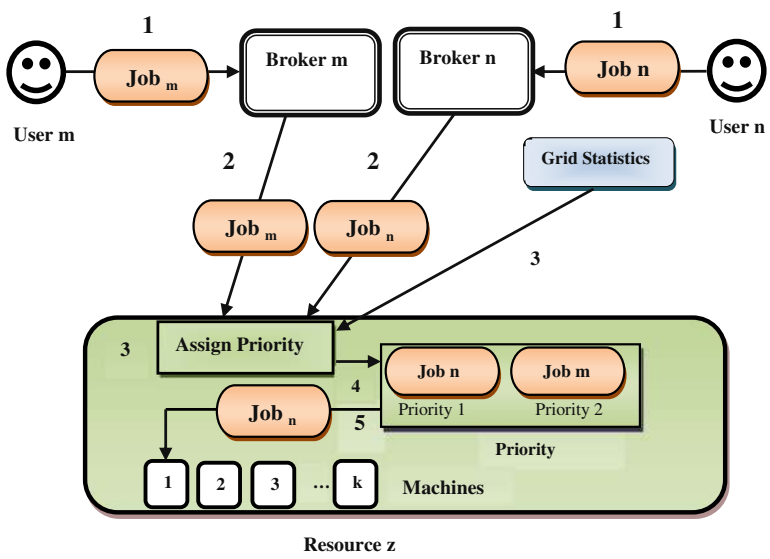

Fig. 11 Agreement-based mechanism steps [70]

processor allocation. GABL divides the request for parallel processing on the basis of available free sub-meshes and maintains a list of allocated meshes. The aforementioned mechanism reduces the average turn-around time of jobs and increases the system utilization. The GABL strategy depends on the number of allocated sub-meshes in the busy list and not on the size of the mesh. It decreases the communication overhead by decreasing the number of sub-meshes allocated to the user application. It is considered as a flexible allocation technique even if the bandwidth is limited and contention is high.

4.12 Market Mechanism

Market mechanism is a distributed RA mechanism based on a common resource marketplace. The negotiation process for the mechanism takes place between a central facilitator and n agents. The facilitator disseminates the related price-per-use information of various resources to the participating agents. After receiving the information, each agent decides the optimal or near optimal request and informs the concerned facilitator about its decision. The facilitator then adjusts the price information and disseminates this information. This process relies on equilibrium where the total number of requested resources is equal to the available resources [140]. The central facilitator that collects and distributes information becomes bottleneck in achieving communication efficiency when the number of resources increases. The market mechanism can be applied in RA problems like travel arrangements, electric power networks, traffic flow networks etc.

4.13 Compensation Mechanism

Compensation mechanism is a type of distributed RA method [140]. The demand for resources and the corresponding compensation plans continuously change. Therefore, the aforementioned mechanism relies on a structured two-stage negotiation method among agents. Each agent disseminates the resource request and compensation plan to other n - 1 agents. After receiving the disseminated information, each agent reports the expected compensation paid by other n - 1 agents and decides the optimal request based on the goal of the local sub-problem. The negotiation process ends and general equilibrium is achieved when all the n agents reach an agreement [140]. This negotiation process is shown in Fig. 12. The compensation mechanism exhibits high communication overhead than the market mechanism and it is considered efficient when the environment is convex. However, if the environment is not convex then this strategy cannot give an efficient resource allocation.

4.14 Coalition Formation Mechanism

For saving cost, the self-interested agents form coalitions by coordinating the activities with other agents [140]. An agent decides to join the *coalition* only to

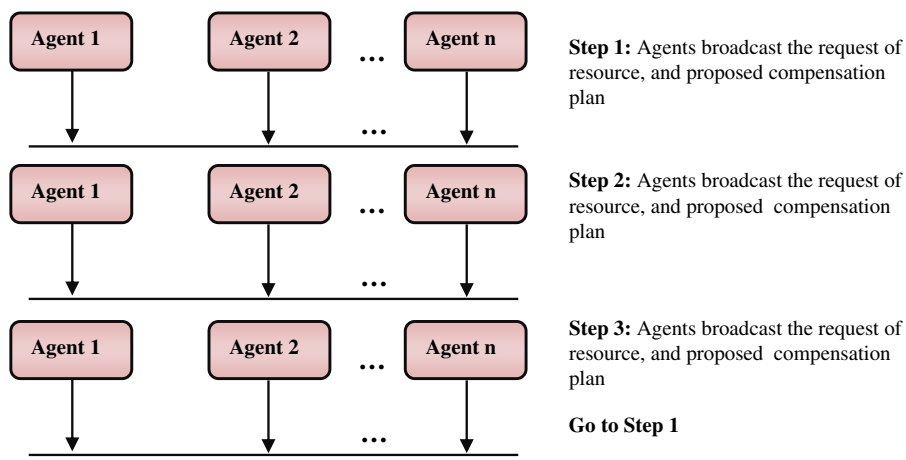

Fig. 12 Negotiation process of the compensation mechanism for RA

save more cost. The coalition models can be classified into two groups:

(a) complementary-based, and
(b) *utility-based* coalition models.

The former model works on the strategy that each party complements the skills and by this way facilitates agents in achieving their goals. The latter model aims at maximizing the profits of the team by distributing the benefits among the coalition members. In a coalition formation process, all agents are divided into a distinctive combination of a coalition group. Then in each coalition group, a subcoalition group with maximum utility is formed known as optimal team. Then each agent of the group divides the profit and resources. The communication overhead is high in coalition formation mechanism because each agent searches for other agents to form coalition.

The choice of allocation strategy among market-based, compensation and coalition formation mechanisms is based on four major factors; the in time information availability, the agents' relationship, the agents' distribution strategy, and the total number of agents. The market mechanism requires incomplete information about the agents while the compensation and coalition formation mechanisms require complete information. The agents are non-cooperative with one another in market and compensation mechanisms while they may cooperate in coalition formation. In a market mechanism the central facilitator becomes the performance bottleneck when the agents are distributed improperly. In such a situation, the compensation and coalition formation methods are considered efficient. The communication and computation costs are high in compensation and coalition mechanisms, so they are suitable when there is a small number of agents' participating in resource allocation.

4.15 Priority-Based Resource Allocation Technique

It is a distributed RA strategy that takes into account factors like computation power of nodes, the upper limit of shared programs, the transmission bandwidth of peripherals, the communication channel bandwidth, and the QoS requirements of applications [44]. Four types of resources are targeted: disks, printers, memory, and resident programs. The resources are allocated with or without the limitation of network bandwidth, with dynamic environment, and the specified QoS of applications. In a dynamic environment, the priority-based strategy finds the resources that give minimum completion time of all the applications. The concurrency among applications is increased to minimize execution times. The strategy works in two stages. At a first stage, a user requests a resource, and the dispatcher, dispatches the application into the pools of priorities. At a second stage, the RA algorithms schedule the applications in each pool by using an objective function and allocate resources. The objective function favours the scheduling of the least resource demanding applications. A variant of Dijkstra's algorithm is used for routing and finding the maximum bandwidth path between the RA node and a target node. Figure 13 shows the referenced priority-based allocation strategy. Priority-based allocation strategy suits well the parallel applications execution where minimum execution time of all applications is required. The analysis shows that when there is no bandwidth limitation then this strategy results in a near optimal allocation. In case the network bandwidth is limited, then the priority-based allocation still processes the same number of concurrent applications as in the case of no bandwidth limitation. However, in the former case the resources' utilization is affected.

4.16 State Space Search (SSS) Technique

In *state space search* (SSS) technique, processor is considered as the main resource for allocation. A modified *A*- algorithm* [133] is used to find the optimal and sub-optimal allocation of the program modules in a distributed computing system. The node of a state space search tree represents the module allocated to a processor. The search tree consists of nodes to be searched and the communication path between the nodes. The searching is started from the root node, traversing other nodes and finally terminated when the node with the minimum allocation cost is reached that can fulfil the application execution demands. Two types of cost are considered, namely, *execution* and *communication* cost (between modules allocated to different processors). The cost function is the sum of execution and inter-module communication costs over all processors. The main objective of the SSS technique is to allocate the program modules to the processors so that the cost function is minimized,

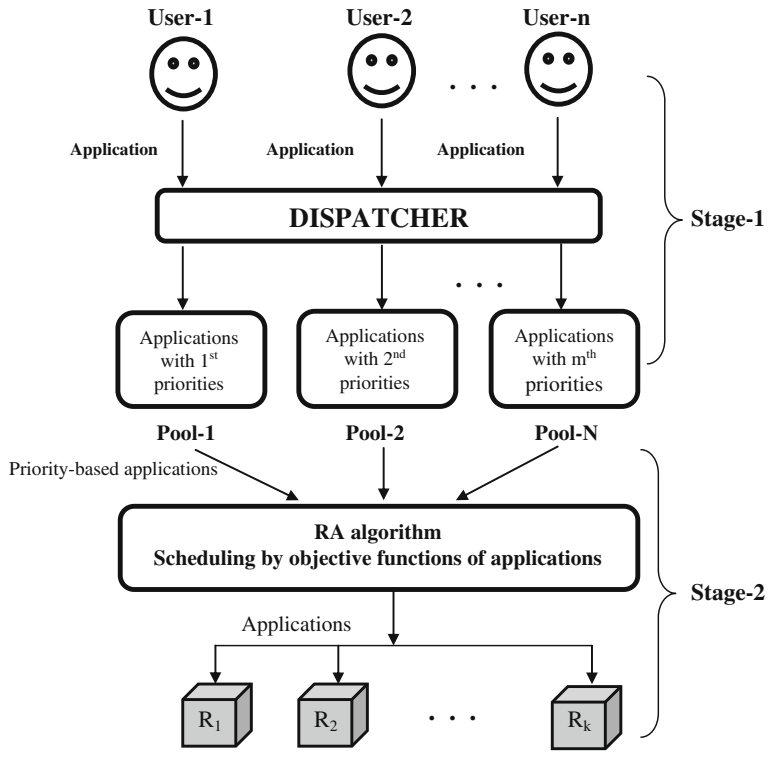

Fig. 13 Flow diagram of the priority-based RA mechanism

without violating the storage or load constraints of the processors. The SSS strategy also reduces the number of search nodes in the search space. If there is large number of nodes in a search space, then this strategy is not preferred because it increases the communication cost.

4.17 PDRAP and SDRAP Technique

Discrete RA problem (DRAP) is a combinatorial optimization problem where limited numbers of resources are allocated to agents such that the objective function of some performance measure is optimized. The general DRAP is an *NP*-hard problem [47, 121]. In the aforementioned technique, divide-and-conquer policy is implemented by considering *sequential discrete RA problem* (SDRAP) and *parallel discrete RA problems* (PDRAP), the later aimed to run on a hypercube [121]. In SDRAP, the number of agents appears in the ratio of power of two. The implementation idea of this SDRAP technique is to apply the divide and conquer rule on the original DRAP by involving only two agents initially for resource allocation. Each agent has its own utility function. The intermediate results generated in the problem solving process are calculated which generate an optimal objective value for the whole problem.

In PDRAP the agents are located adjacent to each other which mean that they are neighbouring nodes. Thus the communication speed is higher than broadcasting or passing message to nonadjacent nodes. Also, no special routing scheme is needed because there is a direct link for communication between adjacent nodes which reduces message propagation delays. Performance-wise, PDRAP was proven superior when the total number of agents was larger than the number of processing nodes on the hypercube computer.

4.18 Dynamic Resource Allocation Mechanism

Many RA mechanisms are based on a static approach. The main flaws in static approaches are that they are restricted by the administrator's intervention and dynamic change of applicant requirements [45]. To overcome the flaws, an alternate dynamic approach of RA has been proposed in [45, 48]. In *dynamic RA* mechanism, the Grid engine dynamically deploys

the Grid applications to the resources as soon as they are discovered. If the RA violates the required level of service at runtime due to change in applications requirements, then the applications are redeployed to other resources. The aforementioned approach of dynamic RA combines the best-fit and process migration approaches. In the best-fit mechanism, the process is mapped to the smallest number of available resources that can fulfil its requirements. When the allocated resources get congested then each process receives a small amount of processing time. Therefore, the process is migrated to other available resources. Dynamic RA mechanism is preferred when context switching does not matter a lot because process migration is allowed during run time when application requirements get change or congestion occurs.

4.19 Resource Allocation by Means of Option Contracts

Option contracts allow the holder to buy or sell an asset (Grid resource) at any pre-determined price for future use. This pre-determined price is known as strike-price. Options can be bought or borrowed and then sold. After submitting the jobs, users have the option to decide how much they can pay to utilize a special required resource [71]. The resources are considered to be limited, so additional utilization cost means additional price. The aforementioned is called spot-price. This price of a resource may fluctuate depending on the users and resource providers' performance. These price fluctuations put adverse effects on applications and schedulers [22], and make the Grid unreliable in terms of execution costs. To reduce the effect of the large price fluctuations, option contracts allow the users to benefit from expected price changes.

4.20 Heuristics-Based Resource Allocation

In a distributed data stream processing environment [8], polynomial time heuristics have sub-optimal solution for operator mapping. According to the aforementioned scheme, stream processing application is structured in the form of binary trees. However, each application has to perform several operations that are represented in the form of operator.

Four types of relevant operators mapping problems were described in the proposed approach which emphasized on how to minimize the CPU utilization allocated to different jobs or applications. Minimum CPU utilization has been achieved by sharing or reusing the intermediate results across different applications. The aforementioned strategy results in better throughput with QoS requirement. The problem is *NP*-hard [8, 50] therefore, polynomial time heuristics were proposed as a sub-optimal solution. The analysis shows that a heuristic-based allocation increases the probability of finding a valid mapping. The top-down approach outperforms the bottom-up one particularly when it is combined with a breadth-first heuristic for traversing the application tree.

4.21 A Greedy Double Auction Mechanism (GDAM)

An auction-based RA mechanism is proposed in [28]. Specifically, the mechanism is based on double auctions where both users and resource owners submit their bids. Double auctions are classified into two categories. The first category is named SDA (Single-unit Double Auction), where at most one unit of resource can be traded in one auction. The second category is known as MDA (Multiple-unit Double Auction), where more than one unit of resource can be traded in one auction. Because MDA is more suitable for a huge number of buyers and sellers, authors propose an algorithm that falls into the second category. The proposed RA mechanism, called GDAM, tries to make the resource consumers and providers trade as more as possible under the guarantee of their demands. Through an experimental evaluation, authors show that GDAM outperforms the traditional MDA mechanisms on both the total trade amount and the user satisfaction percentage. They also experimentally show that as the number of the auctioneers becomes larger the economic efficiency of their algorithm also increases, rendering in that way GDAM fully suitable for Grid environments. The high trade amount value shows the maximum usage of resources by this mechanism. Also the user satisfaction is nearly 100 %. It can be concluded that the use of GDAM strategy is suitable when the number of participants is very high.

4.22 Modified Least Cost Method for Grid Resource Allocation

A RA algorithm for Grid environments to allocate jobs to computing nodes has been proposed in [119, 120]. The proposed algorithm (called MLCM) is based on a modified version of Least Cost Method (LCM). Specifically, the aim of MLCM is to find the portion of each job that is allocated to each processor, so as to minimize the overall computation cost. The paper formulates the problem in a rigorous mathematical way and proposes a Linear Programming (LP) model. Authors provide a comparative performance analysis of their proposed method, with four other Grid RA methods existing in the literature named *divisible load theory* (DLT) method, *northwest corner method* (NWCM), *least cost method* (LCM), and *vogel's approximation* method (VAM).

According to their experimental evaluation that takes into account different Grid scenarios, MLCM results in allocation schemes that produce less computational cost as compared to other well-known RA strategies.

4.23 Dynamic Level Scheduling (DLS) Mechanism

A dynamic level scheduling (DLS) algorithm is proposed in [123] that selects the best task-resource pair for the next scheduling by using dynamically changing priorities of the tasks. The DLS algorithm takes into account the interprocessor communication overhead during the phase of mapping precedence constrained communicating tasks onto heterogeneous processing architectures. The heterogeneous architectures may have limited or irregular connecting structures. The DLS schedules communication between communicating resources by reserving the resources that are used in transferring data for the duration of data transmission. The DLS calculates the dynamic level of task-resource pair to minimize the computational time of the application on a particular resource. The aforementioned technique eliminates the shared resource contention by performing scheduling and routing simultaneously. The DLS mechanism performs very well by making trade-off between load balancing and inter-processor communication. It is considered best in the situations where time is a critical factor in scheduling decisions. It can also be used effectively in iterative scheduling strategies.

4.24 Resource Allocation Using Reinforcement Learning

Reinforcement learning is a mixed Machine Learning and Artificial Intelligence technique that enables machines and software agents to automatically adapt their learning behaviour to maximize system performance. Based on this technique, the authors in [39] studied the minimalist decentralized algorithm for the allocation of heterogeneous Grid resources. The heterogeneous reinforcement learning agents share Grid computational resources for their computational needs. The agents do not communicate with each other. Instead, they receive the start-time and the end time of tasks assigned on particular resources, which serve as reinforcement signals. When a job is completed, the corresponding agent receives a reinforcement signal, and calculates a metric which is translated as a reward for the chosen resource. Specifically, the aforementioned metric is calculated as the sum of the waiting time of the respective task in queue and the execution time of that task on the chosen resource.

To select resources, the agents use a Q-learning approach where they keep for each action a Q-value that is calculated by using the corresponding reward of each resource. The Q-value shows the efficiency of a particular resource in the past. The authors have compared the reinforcement learning technique with two well known resource selection techniques:

(a) the *random selection rule* where resources are selected randomly with uniform probability between resources and
(b) the *least loaded rule* where a resource with minimum load is selected for task execution. The authors experimentally showed that the proposed reinforcement learning technique outperforms the aforementioned techniques.

4.25 Combinatorial Auction-Based Resource Allocation

The authors in [27] proposed a *combinatorial auction-based* protocol for RA. The protocol works in four phases:

(a) requests information from local market for auction (LMA),
(b) generates all possible combinations between submitted tasks and resources,
(c) determines the allocation of tasks onto resources, and
(d) sends the tasks to the chosen Grid service providers along with their rewards.

The LMA helps Grid service providers to share their characteristics, and users to find the best matched resources according to tasks' requirements. The LMA consists of external auctioneers that collect information on combination of resources and the corresponding price value of these combinations from the user brokers. Based on this price value, the LMA executes an approximation algorithm for determining the winner for the combinatorial auction, sends the winner information to the brokers, and informs the Grid service providers about the tasks to be executed on them. In this method a user bids one value for combination of Grid resources which meet the task requirements. In this way a user bids on group of resources instead of bidding on individual resources for task execution. The authors have simulated the combinatorial auction-based protocol on SimGrid simulator.

4.26 Swift Scheduling Mechanism

A dynamic scheduling technique called *Swift Scheduling* (SS) has been proposed in [127] that combines the characteristics of *Heuristic* approach and *Shortest Job First* (SJF) method. Heuristic approaches are used usually when an optimal solution cannot be reached. In SJF, the job with the smallest possible execution time has higher priority in scheduling. In SS technique, the incoming tasks are stored in a job queue and the available resources description is stored in a resource queue. The SS scheduler selects the resources from the resource queue using a heuristic function. The heuristic function selects optimized resource for a specific task that can complete it within minimum time. The SS mechanism takes into account the storage and processing requirements of a task along with tasks and resources priority. The main objective of this technique is to reduce the waiting time of tasks in queue and their overall computational time. The authors have analysed the results by comparing the SS technique with the *first-come first-serve* (FCFS), SJF, and *simple fare task order* (SFTO) methods. After simulation, it was deduced that SS completed all tasks with minimum time and minimum cost by utilizing the maximum amount of resources. The SS technique best fits for executing real-time applications where the execution time affects the processing results. The implementation results show that the SS technique has significant impacts on allocating Grid computational resources to astrophysics, high energy physics, and biotechnology applications.

4.27 Hybrid Resource Allocation Method

Hybrid RA (HRA) method [119, 120] is the modified version of *Divisible Load Theory* (DLT) method by integrating it with *Least Cost Method* (LCM). The DLT method is an optimal RA [144] and scheduling technique that divides the computing load into small parts which are then allocated to static processors on the basis of their computing capacities using LCM. In LCM method, the equal sized task portions are allocated to the processors incurring the least allocation cost retrieved from a cost table. The HRA technique allocates the Grid resources by combining DLT and LCM in such a way to minimize the total computational cost. Specifically, HRA divides the tasks into equal sized portions and allocates them to the processing resources using the LCM technique. In case a tie occurs during the allocation phase, then the tie is broken by selecting the resource that can host the task of maximum size. HRA technique works well in situations when user has limited budget and requires Grid resources of high capacities.

4.28 Association-Based Resource Allocation

Association-based Grid RA mechanism was proposed by Manavalasundaram et al. in [87]. The authors have proposed new computational economy that regulates resources demand and supply, provides incentives to the resource owners for leasing, and helps in RA.

Based on computational economy, the sharing of coordinated distributed clusters is called *Grid-Association* that enables logical coupling of cluster resources. In Grid-Association resource providers are named cluster owners, while resource consumers are termed as end-users. Grid Association Agent (GAA) is an entity representing a resource management system, which enables the coordinated resource sharing between clusters. A cluster can get membership of the association by instantiating a GAA entity. GAA plays the role of resource coordinator, spanning over all the clusters. In case local resources are insufficient for meeting users' requirements, then the association-based mechanism uses resources from Grid-Association. This method provides complete autonomy to each user (resource producer and consumer). In the first stage, the user submits a task to the local scheduler for RA. In case local resources are not available or incapable of fulfilling user requirements, then the task in question is migrated to the fastest or cheapest resource (meeting QoS requirements) in other cluster within the association.

4.29 Heterogeneous E-Waste Resource Allocation Mechanism

The *Electronic waste* (E-waste) is a name used for electronic products approaching to the end of their useful life. E-waste Grid consists of components that are likely to be non-uniform and possibly less reliable than new state of the art components. E-waste resources are heterogeneous in nature, so it is impossible to predict with what power/speed tasks will execute over these resources. Allocation of E-waste Grid resources is a challenging task that gained the attention of research community in the last few years. *Heterogeneous E-Waste Resource* (HER) allocation mechanism [84, 85] is specifically designed for the allocation of E-waste computing resources that are geographically dispersed. HER is an auction-based mechanism working in a continuous fashion that allocates resources without requiring the users to know how long the task will take to execute. Periodically, the nodes within the system execute a small test application, with the execution time being recorded in a database for benchmarking purposes.

Each node is credited points based on its performance (execution time) when executing the test application. Along with the score points expiry times are also recorded in the database, with the score points being removed from the corresponding nodes after exceeding the aforementioned expiry times. Each time the test application is executed, the old score points are replaced with the new ones. Nodes that cannot finish the test application are marked as unreliable or faulty nodes and are excluded from executing tasks. Given that each node within the system has been credited its quantum, then a task submitted to the Grid is immediately allocated according to a pre-processed auction, provided that there are sufficient resources. Each node bids for tasks based on a pre-specified percentage of its credited points. Each task asks a specified amount of points and is assigned on the node bidding the highest amount of points. When a task is assigned on some node, the asked points are deleted from the node in question, reducing in that way node's ability for future auctions while processing the respective task. After a task is completed, the points asked from the task in question are returned to the node executed the respective task.

4.30 Hierarchical Trusted Resource Allocation Mechanism

The *Hierarchical Trusted RA* (HTRA) mechanism [131] was designed for trusted scheduling, execution and monitoring of jobs in mobile Grid environment (e.g., laptops devices and personal digital assistants). In HTRA architecture shown in Fig. 14, each local cluster in mobile Grid consists of a set of resources and a cluster head (CH). The workload in each cluster is measured by the monitoring agent (MA) which sends the resource status information to the CHs. The CHs send this information to master server (MS) that groups and controls the respective local clusters. The MS stores information about the status and price of each resource on each local cluster. The user submits the task (T) and required resource details to the MS which divides each submitted task into sub-tasks (t_1,...,t_n), selects the best CHs based on minimum load and latency and assigns the sub-tasks to these CHs. Each CH finds the available processing power and average load on resources within its local cluster and allocates to the trusted members.

The trust of a node is calculated by the formula given in [131]. If a CH cannot allocate resources in its cluster, then the MS forwards this request information to another CH and this process continues

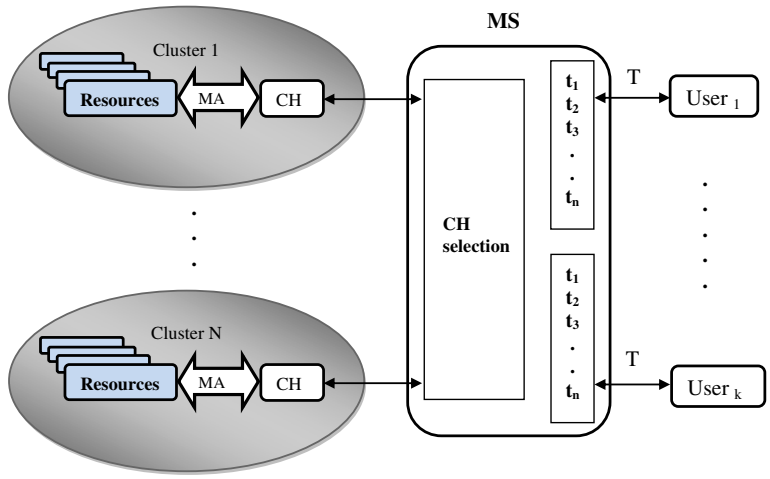

Fig. 14 HTRA Architecture

until the task is assigned successfully. After completing tasks execution, the CHs collect the resultant data of processed sub-tasks from the corresponding resources and send them back to the MS. The MS aggregates the completed sub-tasks results collected from each CH, stores them in its database and sends them back to the user. The HTRA mechanism was tested by measuring average execution delay, average success ratio, average energy consumption, average workload and throughput using network simulator NS-2. The results were obtained by simulating HTRA against *hierarchical resource allocation architecture* (HRAA) and *hierarchical campus-wide mobile Grid* (HCMG) using above mentioned metrics (i.e., execution delay, success ratio, energy consumption, workload and throughput) with various execution rates (i.e., 250 Kb, 500 Kb, 750 Kb, 1000 Kb) and various number of tasks (i.e., 2, 4, ..., 10). The results show the supremacy of the HTRA mechanism over HRAA and HCMG.

4.31 Price Directed Proportional Resource Allocation Mechanism

In this paper, a market-based mechanism is proposed, which uses a three layer system model [79]. The lower layer is composed of Grid resources that are owned and allocated by Grid resource agents residing at the nodes within the system. The middle layer plays the role of Grid market consisting of three types of agent:

(a) Grid task agents,
(b) Grid resource agents, and
(c) Grid request agents.

The top layer is the application layer where Grid request agents play the role of interfaces through which users interact with Grid market. Grid resource agents sell the resources of the underlying Grid to Grid task agents, with the latter ones making buying decisions within budget constraints to obtain the required computational resources. The objective of task agents is an optimization problem where the aim is to complete tasks at minimum cost. The price-directed RA algorithm consists of two parts:

(a) the Grid resource agent part, and
(b) the Grid task agent part.

Grid resource agent broadcasts the starting prices. At each iteration, they calculate and announce the new prices to Grid resource task agents and this process is repeated until the demand is not equal to the supply of the resources. On the other extreme, Grid task agents determine their optimal allocation and make negotiations with Grid resource agents in an iterative fashion until the total demand equals to the total amount of available resources. All the interactions between Grid task/resource agents take place through the Grid market. JAVASIM network simulator is used to implement the algorithm. The simulation is run on a TCP/IP network model supported by JAVASIM.

4.32 Load Forecast-based Allocation algorithm

Every Grid user wants to complete its tasks within the budget constraint as quickly as possible. This situation becomes a multi-player game in which every user competes for processing resources. To obtain maximum satisfaction within limited budget constraint a load forecast-based allocation algorithm is proposed in [23, 146]. In this algorithm, first Grid resources register themselves with Grid Information Service (GIS). User discovers resources by querying the aforementioned service. In response of the request, GIS returns a list of resources. According to the information of Grid resources, Grid users submit their bidding functions. The resource prices are formed by Grid resources that transfer their feedback to Grid users. The submission of bidding by users continues until a convergent condition is satisfied. Analysis processing elements of different speeds were created and then integrated to produce a single processing element. The resulting processing element or Grid resource may be considered as a single processor with high processing speed, or shared memory multi-processors, or cluster of computers having distributed memories. The resources have initially no load, but later on they were loaded with workload estimated based on load conditions of the resources. The results of *load forecast* method was the best because each user has information about the other users which helps in forecasting resource load.

4.33 Hyper-Heuristic Approach

Because optimal mapping of jobs to the available processing nodes on the Grid is an NP-complete problem [10, BeC01, 50], heuristics and meta-heuristics have been proposed to solve the aforementioned problem. Usually, heuristics result in local optimum solutions. For that reason, meta-heuristic are used to help heuristics escape from the local optimum traps. A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to produce a high quality solution. The most well-known meta-heuristic algorithms are: *Genetic Algorithm* (GA), *Local Search* (LS), *Simulated Annealing* (SA) and *Tabu Search* (TS). Because heuristic and meta-heuristic approaches require substantial expertise in both the problem domain and appropriate heuristic techniques, [10] introduces a more general optimization methodology called hyper-heuristic (HH) to solve the aforementioned problem. *Hyper-heuristic* operates at a higher level of abstraction than the current state of the art meta-heuristics. The above HH comes from the hybridization of GA with other meta-heuristic or heuristic algorithms. In that way the proposed HH illustrates a set of strategies used to select a heuristic from a set of low level heuristics (LLH). A single iteration of HH approach is decomposed into two steps, heuristic selection and movement acceptance. The selection process uses greedy strategy that selects any LLH randomly based on its previous performance.

The task model is composed of independent, indivisible tasks arrive in a random manner. Each task has parameters like task-id and number of required processing cycles. On arrival the tasks are placed in a queue of unscheduled tasks and then are allocated to the ready resources in the form of batches. If at any stage the number of available resources is less than the number of tasks, then most-into-list (MIL) scheduling heuristic is used for generating randomized initial population. The MIL assigns random number of tasks to resources in a round robin fashion. The remaining sorted tasks are assigned to the resources using two step process. In first step, a task that has longest processing time is assigned to the fastest resource. In second step, the next task with shortest processing time is assigned to the fastest resource. So in this way all the tasks are assigned to the resources.

The performance of the HH technique was compared with GA, GA-LS, GA-SA, and GA-TS for 50 runs of the sample test case for scheduling 200 tasks on 20 resources, 100 tasks on 30 resources, and 400 tasks on 250 resources. In all the experiments, the HH outperforms the other counterparts with minimum makespan.

4.34 Adaptive Grid Scheduling Mechanism

The *Adaptive Grid Scheduling* (AGS) [114] is a decentralized and hybrid scheduling mechanism that combines both the static and dynamic techniques for scheduling tasks on computational Grid. The AGS method detects efficiently the Grid resources failure and recovers the system to its normal state. Working strategy of AGS relies on a two phase process. In the first phase called discovery and monitoring

phase, the AGS discovers and monitors Grid computational resources. When the *directed acyclic graph* (DAG) tasks arrive then in the second phase called scheduling-rescheduling phase, the AGS schedules the tasks by using a static scheduling technique. After the transmission of tasks to the allocated resources, each resource re-schedules the tasks if necessary. This task transmission process optimizes the makespan of the DAG by using its unique property called *in advance task transmission* (ITT). In ITT the tasks are transmitted in advance to the allocated resources before waiting for the busy resources to be free from execution. The ITT process avoids communication link failure which shows the reliability of this technique.

4.35 Data-Intensive and Network Aware Scheduling Technique

Most of the applications that need computing resources are data intensive. Scheduling such applications on Grid resources take into account not only the execution time of the tasks on these resources but also the time required for fetching and transferring data for the tasks from distributed locations. In [91], the authors proposed a data intensive and network aware (DIANA) technique that considers network characteristics (data transfer time, delays, bandwidth, jitter etc) along with the processing power of the computational resources for making scheduling decisions. Main target of the paper [91] is to reduce the queue and execution times of data-intensive tasks. The DIANA scheduler considers the sites for allocating the tasks that have low load, short waiting queue, required data, and network stability. The network stability is determined by the network monitoring system that stores all the information in its database which is used for scheduling decisions. The scheduler creates a global cost matrix on the basis of data, computation, and network costs for each site. Then the site with the least cost is selected for tasks execution. This system works well in case when there are queues of long jobs waiting for execution.

4.36 Negotiation-Based Advanced Reservation

Grid applications demanding various resources are scheduled as a result of negotiation between the user and service provider. In [104], a *negotiation–based Heterogeneous Earliest Finishing Time (HEFT)* algorithm is proposed which is the extension of the existing list scheduling technique for minimizing the workflow makespan by using an *advanced reservation* mechanism. The traditional HEFT algorithm operates in three phases:

(a) weighting phase,
(b) ranking phase, and
(c) mapping phase.

To support the negotiation-based scheduling, the HEFT algorithm was extended by including a fourth phase i.e, the negotiation phase. In the weighting phase, the workflow activities are assigned weights that are equal to the predicted execution times of the activities. In the ranking phase, a value is assigned as a rank to each workflow activity by traversing the workflow graph from the bottom to top. This rank value for the activity is calculated by adding the weights assigned to the activity in weighting phase and the maximum rank value of all the successors. The workflow activities are sorted on the basis of these rank values. In the mapping phase, the activity is scheduled on the resource that gives minimum finishing time. In negotiation phase, a negotiation is made between the scheduler and resource manager of all the resources by traversing the activity list. After successful negotiation, a resource that is near in fulfilling all the requirements of the user request is selected and reserved.

4.37 Performance-Based Scheduling Strategies

In [77] the authors develop performance-based scheduling strategies for high throughput computing applications. The performance of these strategies is analyzed by scheduling independent tasks on federated Grids. The resources in these Grids are termed as internal or external resources. Internal resources are the Grid local resources existing in the same Grid, while external resources are the resources existing in other Grids. Four strategies that are; static objective (SO), dynamic objective (DO), static objective and advanced scheduling (SO-AS), and dynamic objective and advanced scheduling (DO-AS) are considered each of which executes a set of independent tasks. The performance of each Grid in the federation is determined by the total number of jobs it has completed. The makespan of the tasks is minimized and

the throughput is increased by scheduling tasks first to the Grid internal resources. The strategies were simulated by using the GridWaySim testbed. It was concluded that DO-AS outperforms all the other three counterparts in minimizing the makespan by doing the best distribution of the jobs on the Grid resources.

4.38 Game Theoretical Energy Efficient Resource Allocation Mechanism

The *game theoretical energy efficient* technique [Suk09] aims at allocating the deadline constrained tasks to the heterogeneous Grid resources in such a way to minimize the makespan of the tasks and the total power consumed by executing these tasks. Each Grid resource is assumed to be equipped with *dynamic voltage scaling* (DVS) module. DVS technique draws out task execution time by scaling CPU speed and voltage dynamically. The DVS technique is used to run the CPU at lower frequency to reduce power consumption [66]. In the above mentioned game theoretical technique task is also characterized by the resource architecture (CPU type, speed in GHz, bus types, memory etc) along with the deadline constraint where it will be executed. Each resource is considered as a player. The game theoretical allocation technique tries to allocate the task to at least one computational Grid resource in a way to fulfill all the characteristics and deadline constraints of the task and in turns a metatask. This is called as feasible task mapping. The total power consumed is the sum of all the powers utilized by individual task. The metatask is considered to be scheduled with minimum total power consumed by adjusting the DVS level of each resource so that they can guarantee task execution within deadline constraints. The technique was compared with min-min heuristic using linear programming tool called LINDO. The results show that the proposed technique outperforms min-min 5–10 % when considering power as optimization criteria. The time complexity of this technique is $O(nmlog(m))$ where n is the total number of tasks and m the total number of computational Grid resources.

4.39 Two-Phase Heuristic Mechanism

The *two-phase heuristic* (2PH) approach [103] is the extension of min-min heuristic used to schedule independent tasks in a distributed environment with minimum energy consumption. The 2PH technique additionally implements the local search operator called H2LL which performs the load balancing on Grid computational resources. The H2LL operator moves the task from the heavy loaded resource to the least loaded resource that has minimum completion time among all the candidate resources including the moved task. The sensitivity analysis shows that H2LL influences the quality of schedule, so the implementation results show that 2PH heuristic produces good results than min-min heuristic in much less time (in 3 milliseconds) on Xeon E5400 server class machine. The 2PH technique can be used to schedule independent tasks with less energy consumption where min-min can be used.

4.40 Traits of Resource Allocation

RA mechanisms in Grid systems can be compared based on some common characteristics they adopt for solving computing- and data-intensive applications. Searching mechanism, job type, time complexity, optimality, allocation strategy, operational environment and objective function are some of the common and basic characteristics that should be examined in each RA mechanism. These features are briefly described as follows.

4.40.1 Searching Mechanism

The process of finding the resources and activities is known as the searching mechanism. Searching is a vital step during the RA process; the faster the searching the quicker the overall RA process will be. In this survey, different RA strategies explicitly define the searching process. Some of the strategies have implicit searching mechanisms.

4.40.2 Application Type

Grid supports different types of applications and the RA mechanisms are developed on the basis of these application requirements. The applications may be dependent workflow applications, independent divisible applications, or atomic tasks. Workflow applications consist of many divisible tasks with data and execution dependency. Divisible applications also consist of multiple tasks with no data and execution dependencies and hence no communication among

each other. The atomic tasks are indivisible single tasks.

4.40.3 Time Complexity

Time complexity of RA mechanism can be defined as the amount of time the mechanism or the algorithm takes to complete the process. For instance in [115], the time complexity for single-RA algorithm is $O(n^3)$ and $O(n^3 log(1/))$ for two-resource allocation algorithm where two resources are allocated. The complexity for every mechanism may vary based on the design of the RA mechanism. Moreover, complexity of resource scheduling strategies plays an important role particularly in real time computing systems where an immediate response for the RA request is required.

4.40.4 Optimality

Optimality can be defined as the RA strategy that performs best against the objectives. In this survey, Grid RA mechanisms have been evaluated for the delivery of optimal solutions. Since every RA mechanism has to achieve an objective function, optimality is measured on the basis of achieving that function.

4.40.5 Resource Allocation Strategy (RAS) Taxonomy

The process of dispatching resources to activities is called RA strategy (RAS) *taxonomy*. There are two types of RAS taxonomy:

(a) centralized, and
(b) distributed.

Resources that are allocated through single point are known as centralized RAS taxonomy. Alternatively, resources that are allocated from multiple points are known as distributed RAS taxonomy.

4.40.6 Operational Environment

The environment and platform on which the RA mechanism can be executed is known as the operational environment. For example *"radio RA for data services"* is executed on universal mobile telecommunications system (UMTS) Networks.

4.40.7 Objective Function

Every RA mechanism has an objective function specifically designed for a specific purpose of the mechanism, for example, minimizing the cost of tasks execution on a resource. Mostly, the main objective function of the mechanisms is the efficient allocation of resources.

4.40.8 Citations to the Mechanisms

Citation means reference to a published or unpublished work. In broader sense, it demonstrates the importance or validity of that mechanism. The above mentioned citations in Table 3 were taken up to the date of submission of this paper.

5 Immediate Extension of Grid Resource Management

Many emerging IT technologies like *Cloud computing*, *Autonomic computing*, *Mobile computing* etc are the enhanced paradigms that are developed quickly based on the Grid technology in recent years. Cloud computing provides opportunity to the users to use Cloud resources as services known as *infrastructure as a service, platform as a service,* and *software as a service.* Autonomic computing manage applications execution in dynamically changing and unpredictable environments without any user's manual intervention [15, 102, 109] by regularly checking and monitoring the changing conditions automatically. Mobile computing extends the computing facilities to the smart mobile devices in order to enhance their capabilities on demand for accommodating computational- and storage-intensive applications [1, 15, 59, 111, 118].

In the following subsection, we discuss about the resource management in Cloud computing environment in detail.

5.1 Cloud Computing

Of all these paradigms, Cloud computing appears to be the most promising one that delivers reliable services through virtualized computing and storage technologies. Buyya et al. [19] define Cloud as:

"Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and

virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers".

Cloud computing is a computing platform that provides opportunity to the users to use the Cloud resources as services that satisfy the predefined SLA. The Cloud computing infrastructure is fully supported and evolved from Grid computing but the difference come in technical and non-technical characteristics like resource pooling mechanisms on-demand, scalability, virtualization support, resource usage under pay-per-use rule, guaranteed SLA, and self service [105]. The aim of shifting the Grid technology to Cloud was to deliver economy-based reliable storage and computing resources to the users as consistent services similar to the physical resources through the data centers that aggregate processing and computing power of the Cloud. Cloud is a service oriented platform that provides three types of basic services at different levels:

(a) Software as a Service (SaaS),
(b) Platform as a Service (PaaS), and
(c) Infrastructure as a Service (IaaS).

The user requests a service through QoS parameters which are recorded in a predefined SLA. The objective of delivering the three types of services is to give a facility to the users to use existing softwares, platforms, and infrastructures on the fly without investing in new ones and wasting capacities in personnel trainings [59].

Cloud and Grid computing paradigms share a lot in developing resource allocation mechanisms. Grid computing supports bath scheduling of tasks to computing resources which are managed by a local resource manager that assigns the resources for a specific duration of time specified by the user. If the required resources are not available for the user's requested time duration, then the tasks wait in the wait queue till the availability of resources. The resource allocation strategy in Cloud computing minimize the tasks wait time because multiple resources (may be infinite [26]) are shared by multiple users at the same time without waiting in long queues [35] by using virtualization. Virtualization provides encapsulation and abstraction to the Cloud which in turn gives efficient and cost effective utilization of resources. In Grid environment, the organizations want full control of the resources and do not fully virtualizes the resources, but somehow virtualization is also provided like Nimbus [19, 35]. Users' preferences (reliability, trust, cost, security of their operations) and Cloud services performance (e.g., response time that is subject to the network traffic) fluctuates dynamically in Cloud environment, therefore the resource selection approach will also match them dynamically. The user strives to minimize the resource usage cost while the provider tends to maximize it. The resources in Cloud are more reliable than Grid [16]. But in Cloud, the most challenging task in the resource matching process is the inter-Cloud communication when the requester and provider use different notations for describing their services and requirements because service description in terms of syntax and semantics is difficult to enforce [26]. So symmetric attributes-values based matching of user requirements and request is not possible. This drawback is covered by the semantic web service that applies an QoS ontology for understanding different representations. The QoS ontology describes QoS type, dynamicity and importance. Based on the QoS preferences, the ontology based matchmaking approach helps users in deploying their applications on the most suitable and proper IaaS provider.

6 Summary and Conclusions

In this paper, we have reviewed, compared, and highlighted number of resource discovery and allocation mechanisms. The existing surveys on RA mechanisms cover very few methods on Grid RA, and each one focuses on a single type of underlying architecture; centralized, distributed, static or dynamic. We have consolidated all of the paradigms in a single platform comprehensively and discussed:

(a) the motivations for considering efficient Grid resource management,
(b) resource discovery by considering execution cost, processing time, power consumption and suitable information propagation strategies of the resources, and
(c) the high energy consumption issues in RA mechanisms.

We have compared these mechanisms for Grid

resources based on their common features. Time complexity, searching mechanism, allocation strategy, optimality, operational environment and objective function are the common and basic features that all surveyed RA mechanisms adopt for solving computing- and data-intensive applications. We have also suggested the incorporation of energy-aware features in RA mechanisms because in different situations these mechanisms perform much better but analyzing based on energy consumption, will waste much of the resource power.

We believe that this comprehensive analysis of leading research in Grid domain provides readers with an understanding of the essential concepts of the RA mechanisms in Grid systems and helps them to identify the important and outstanding issues for further investigation. The paper also helps in choosing the best-suited RA mechanism for a particular application or system, and can be proved as an efficient remedy for the researchers working in this area.

The literature shows a lot of work on Grid RA mechanisms but still these mechanisms need improvements to face many challenges. One of the challenges is the lack of generality to make them transparent for every type of service provisioning. These mechanisms need to efficiently accommodate tasks of diverse requirements in order to facilitate the users without caring about the capabilities and limitations of the mechanism. Dealing with dynamicity on application demands is another challenge because most applications demands change dynamically during execution that needs to be accommodated at run time. So, these challenges can degrade system performance if not addressed properly.

References

1. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., Buyya, R.: Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. In: IEEE Communication Surveys & Tutorials. IEEE Communications Society Press, USA (2013, in press)
2. Ali, M., Dong, Z.Y.: RSA-Grid: A grid computing based framework for power system reliability and security analysis. In: Proceedings of IEEE PES General Meeting, Montreal, 6–10 June (2006)
3. Amarnath, B.R., Somasundaram, T.S., Ellappan, M., Buyya, R.: Ontology-based grid resource management. Softw. Pract. Exper. **39**, 1419–1438 (2009)
4. Andreozzi, S., De Bortoli, N., Fantinel, S., Ghiselli, A., Rubini, G.L., Tortone, G., Vistoli, M.C.: GridICE: A monitoring service for grid systems. Futur. Gener. Comput. Syst. **21**, 559–571 (2005)
5. Arora, M., Das, S.K., Biswas, R.: A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments. In: Proceedings of the IEEE International Conference on Parallel Processing Workshops (ICPPW'02), pp. 499–505 (2002)
6. Batista, D.M., Fonseca, N.L.S.: A brief survey on resource allocation in service oriented grids. In: 1st IEEE Workshop on Enabling the Future Service-Oriented Internet – Globecom. Nov 26–30, pp. 1–5 (2007)
7. Balter, M.H., Leighton, T., Lewin, D.: Resource discovery in distributed networks. In: 18th ACM-SIGACT/SIGOPS Symposium on Principles of Distributed Computing (PODC '99), pp. 229–238. Atlanta (1999)
8. Benoit, A., Casanova, H., Sonigo, V.R., Robert, Y.: Resource allocation for multiple concurrent in-network stream-processing applications. J. Parallel Comput. **37**(8), 331–348 (2011)
9. Bethel, W., Siegerist, C., Shalf, J., Shetty, P., Jankun-Kelly, T.J., Kreylos, O., Ma, K.L.: VisPortal: deploying grid-enabled visualization tools through a web-portal interface, Technical Report LBNL-52940, Lawrence Berkeley National Laboratory (2003)
10. Bhanu, S.M.S., Gopalan, N.P.: A hyper-heuristic approach for efficient resource scheduling in grid. Int. J. Comput. Commun. Control **III**(3), 249–258 (2008)
11. Bode, B., Halstead, D.M., Kendall, R., Lei, Z.: The portable batch scheduler and the maui scheduler on linux clusters. In: Proceedings of 4th Annual Linux Showcase and Conference, vol. 4, pp. 1–9. Atlanta (2000)
12. Bonnassieux, F., Harakaly, R., Primet, P.: MapCenter: An open grid status visualization tool. In: Proceedings of the ISCA 15th International Conference on Parallel and Distributed Computing System. Louisville (2002)
13. Bradley, D., Harper, R., Hunter, S.: Workload-based power management for parallel computer systems. IBM. J. Res. Dev. **47**(5), 703–718 (2003)
14. Buyya, R., Chapin, S., DiNucci, D.: Architectural models for resource management in the grid. In: Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing, pp. 18–35. Springer Verlag Series, Germany, Banglore, India (2000)
15. Buyya, R., Calheiros, R.N., Li, X.: Autonomic cloud computing: Open challenges and architectural elements. In: Proceedings of the 3rd International Conference of Emerging Applications of Information Technology (EAIT 2012, IEEE Press, USA), Kalkota, 29 Nov–01Dec 2012
16. Buyya, R., Ranjan, R.: Federated resource management in grid and cloud computing systems. Futur. Gener. Comput. Syst. **26**, 1189–1191 (2010)
17. Buyya, R.: Economic-based distributed resource management and scheduling for grid computing. PhD thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, p. 180 (2002)
18. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. Proc. IEEE **93**(3), 698–714 (2005)

19. Buyya, R. et al.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Futur. Gener. Comput. Syst. (2009). doi:10.1016/j.future.2008.12.001
20. Cai, C., Wang, L., Khan, S.U., Tao, J.: Energy-aware high performance computing: A taxonomy study. In: 17th IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 953–958. Taiwan (2011)
21. Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., Gautam, N.: Managing server energy and operational costs in hosting centers. ACM Sigmet Perform. Eval. Rev. **33**(1), 303–314 (2005)
22. Cheliotis, G., Kenyon, C., Buyya, R., Melbourne, A.: Grid Economics: 10 lessons from finance. Technical Report, Zurich Research Lab, Melbourne (2003)
23. Cheng, C., Zhi-Jie, L.: Parallel algorithm for grid resource allocation based on nash equilibrium. In: Proceeding of 5th International Conference on Machine Learning and Cybernetics. pp. 4383-4388. Dalian (2006)
24. Cooke, A., Gray, A.J., Ma, L., Nutt, W., Magowan, J., Oevers, M., Taylor, P., Byrom, R., Field, L., Hicks, S., Leake, J., Soni, M., Wilson, A.: R-GMA: An information integration system for grid monitoring. In: Springer Lecture Notes Computer Science, vol. 2888, pp. 462–481 (2003)
25. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid information services for distributed resource sharing. In: Proceedings of the 10th International Symposium on High Performance Distributed Computing, pp. 181–194 (2001)
26. Dastjerdi, A.V., Buyya, R.: A taxonomy of QoS management and service selection methodologies for cloud computing: Methodology, systems, and applications. In: Wang, L., Ranjan, R., Chen, J., Benatallah, B (eds.) ISBN: 9781439856413, pp. 109–131. CRC Press, Boca Raton (2011)
27. Das, A., Grosu, D.: Combinatorial auction-based protocols for resource allocation. In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), vol. 14, pp. 251.1 (2005)
28. Ding, D., Luo, S., Gao, Z.: A greedy double auction mechanism for grid resource allocation. In: Proceedings of the 15th International Conference on Job Scheduling Strategies for Parallel Processing (JSSPP'10), pp. 35–50. Atlanta (2010)
29. Elyada, A., Ginosar, R., Weiser, U.: Low-complexity policies for energy-performance tradeoff in chip-multiprocessors. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **16**(9), 1243–1248 (2008)
30. Elnozahy, E.N., Kistler, M., Rajamony, R.: Energy-efficient server clusters. In: Proceedings of the 2nd International Conference on Power-Aware Computer Systems (PACS '02), pp.179–197. Berlin (2002)
31. Etsion, Y., Tsafrir, D.: A short survey of commercial cluster batch schedulers. Technical Report 2005-13, School of Computer Science and Engineering, Hebrew University of Jerusalem (2005)
32. Fernandez, D., Mehri Dehnavi, M., Gross, W.J., Giannacopoulos, D.: Alternate parallel processing approach for FEM. IEEE Trans. Magn. **48**(2), 399–402 (2012)
33. Fidanova, S., Durchova, M.: Ant algorithm for grid scheduling problem. In: Proceedings of the 5th International Conference on Large-Scale Scientific Computing (LSSC'05), pp. 405–412. Berlin (2006)
34. Foster, I., Roy, A., Sander, V.: A quality of service architecture that combines resource reservation and application adaptation. In: Proceedings of the 8th International Workshop on Quality of Service, pp. 181–188 (2000)
35. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop 2008(GCE'08), pp. 1–10 (2008)
36. Freeh, V.W., Pan, F., Kappiah, N., Lowenthal, D.K., Springer, R.: Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster. In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05), IEEE Computer Society, p. 4. 1. Washington, DC (2005)
37. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A computation management agent for multi-institutional grids. Clust. Comput. **5**(3), 237–246 (2002)
38. Garg, S.K., Buyya, R.: Exploiting heterogeneity in grid computing for energy-efficient resource allocation. In: Proceedings of 17th International Conference on Advanced Computing and Communications (ADCOM '09), pp. 14–18. Bengaluru (2009)
39. Galstyan, A., Czajkowski, K., Lerman, K.: Resource allocation in the grid using reinforcement learning. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), pp. 1314–1315. New York (2004)
40. Galstyan, A., Czajkowski, K., Lerman, K.: Resource allocation in the grid with learning agents. J. Grid Comput. **3**(1), 91–100 (2005)
41. Hsu, C., Feng, W.: A feasibility analysis of power awareness in commodity-based high-performance clusters. In: Proceedings of 7th IEEE International Conference on Cluster Computing (CLUSTER '05). Boston (2005)
42. Hsu, C., Feng, W., Archuleta, J.S.: Towards efficient supercomputing: A quest for the right metric. In: Proceedings of 1st IEEE Workshop on High-Performance, Power-Aware Computing (in Conjunction with the 19th International Parallel & Distributed Processing Symposium). Denver (2005)
43. Hsu, C., Feng, W.: A power-aware run-time system for high performance computing. In: Proceedings of 2005 ACM/IEEE Conference on Supercomputing. Seattle (2005)
44. Huang, Y., Chao, B.: A prioity-based resource allocation strategy in distributed computing networks. J. Syst. Softw. **58**(3), 221–233 (2001)
45. Huedo, H., Montero, R.S., Llorente, L.M.: A framework for adaptive execution in grids. Softw. Pract. Exper. **34**(7), 631–651 (2004)
46. Huang, Y., Vekatasubramanian, N.: QoS-based resource discovery in intermittently available environments. In: Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 '02) (2002)

47. Ibaraki, T., Katoh, N.: Resource Allocation Problems: Algorithmic Approaches. MIT Press, Cambridge (1988)
48. Ismail, L.: Dynamic resource allocation mechanisms for grid computing environment. In: Proceedings of 3rd IEEE International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities, pp. 1–5. Lake Buena (2007)
49. Jayasudha, A.R., Purusothaman, T.: Grid scheduling using differential evolution for solving multi-objective optimization parameters. Int. J. Comput. Sci. Eng. **02**(07), 2322–2327 (2010)
50. Kamalam, G.K., Bhaskaran, V.M.: New enhanced heuristic min-mean scheduling algorithm for scheduling meta-tasks on heterogeneous grid environment. Eur. J. Sci. Res. **70**(03), 423–430 (2012)
51. Katoh, N., Ibaraki, T.: Resource allocation problems. In: Du, D.-Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. 2, pp. 159–260. Springer (1998)
52. Karaoglanoglou, K., Karatza, H.: Resource discovery in a dynamical grid based on re-routing tables. Simul. Model. Pract. Theory **16**(6), 704–720 (2008)
53. Kandagatla, C.: Survey and Taxonomy of Grid Resource Management Systems. University of Texas, Austin (2003)
54. Kertesz, A., Kacsuk, P.: A taxonomy of grid resource brokers. In Distributed and Parallel Systems, Springer US, 6th Austrian-HungarianWorkshop on Distributed and Parallel Systems (DAPSYS'06), pp. 201–210. USA (2007)
55. Khan, S., Ahmad, I.: A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. IEEE Trans. Parallel Distrib. Syst. **20**(3), 346–360 (2009)
56. Khan, S.U.: A game theoretical energy efficient resource allocation technique for large distributed computing systems. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 48-54. Las Vegas (2009)
57. Khan, S.U.: A goal programming approach for the joint optimization of energy consumption and response time in computational grids. In: 28th IEEE International Performance Computing and Communications Conference (IPCCC), pp. 410–417. Phoenix (2009)
58. Khargharia, B., Hariri, S., Yousif, M.S.: Autonomic power and performance management for computing systems. Clust. Comput. **11**(2), 167–181 (2008)
59. Khan, A.N., Kiah, M.L.M., Khan, S.U., Madani, S.A.: Towrds secure mobile cloud computing: A survey. Futur. Gener. Comput. Syst. (2012). doi:10.1016/j.future.2012./08.003
60. Khan, S.U., Min-Allah, N.: A goal programming based energy efficient resource allocation in data centers. J. Supercomput (2011). doi:10.1007/s11227-011-0611-7
61. Kim, K.H., Buyya, R., Kim, J.: Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In: Proceedings of 7th IEEE International Symposium on Cluster Computing and Grid, pp. 541-548. Rio de Janeiro (2007)
62. Kolodziej, J., Khan, S.U., Xhafa, F.: Genetic algorithms for energy-aware scheduling in computational grids. In: 6th IEEE International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing (3PGCIC), pp. 17–24. Barcelona (2011)
63. Kolodziej, J., Khan, S.U., Wang, L., Byrski, A., Min-Allah, N., Madani, S.A.: Hierarchical genetic-based grid scheduling with energy optimization. Clust. Comput. doi:10.1007/s10586-012-0226-7
64. Kolodziej, J., Khan, S.U.: Multi-level hierarchical genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment. Inf. Sci. **214**, 1–19 (2012)
65. Kolodziej, J., Khan, S.U.: Data scheduling in data grids and data centers: A short taxonomy of problems and intelligent resolution techniques. Trans. Comput. Collect. Intell. X, 103–119 (2013)
66. Kolodziej, J., Khan, S.U., Wang, L., Kisiel-Dorohinicki, M., Madani, S.A., Niewiadomska-Szynkiewicz, E., Zomaya, A.Y., Xu, C.-Z.: Security, energy, and performance-aware resource allocation mechanisms for computational grids. Futur. Gener. Comput. Syst. **31**, 77–92 (2014)
67. Kolodziej, J., Khan, S.U., Wang, L., Zomaya, A.Y.: Energy efficient genetic-based schedulers in computational grids (Forthcoming)
68. Koopman, B.: The optimum distribution of effort. Oper. Res. JSTOR **9**(1), 52–63 (1953)
69. Krauter, K., Buyya, R., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Softw. Pract. Exper. **32**(2), 135–164 (2002)
70. Krawczyk, S., Bubendorfer, K.: Grid resource allocation: allocation mechanisms and utilization patterns. In: Proceedings of 6th Australasian Symposium on Grid Computing and e-Research (AusGrid '08), vol. 82. Wollongong (2008)
71. Krawczyk, S., Bubendorfer, K.: Grid resource allocation by means of option contracts. IEEE Syst. J. **3**(1), 49–64 (2009)
72. Kutten, S., Peleg, D.: Asynchronous resource discovery in peer-to-peer networks. Comput. Netw. **51**, 190–206 (2007)
73. Lamnitchi, A., Foster, I.: On fully decentralized resource discovery in grid environments. In: Proceedings of 2^{nd} IEEE International Workshop on Grid Computing. Denver (2001)
74. Lamnitchi, A., Foster, I., Nurmi, D.: A peer-to-peer approach to resource discovery in grid environments. In: Proceedings of 11th Symposium on High Performance Distributed Computing. Edinburgh (2002)
75. Laure, E., Stockinger, H., Stockinger, K.: Performance engineering in data grids. Concurr. Comput. Pract. Exper. **17**(2–4), 171–191 (2005)
76. Lawson, B., Smirni, E.: Power-aware resource allocation in high-end systems via online simulation. In: Proceedings of 19th Annual International Conference on Supercomputing (ICS '05). pp. 229–238. Cambridge (2005)
77. Leal, K., Huedo, E., Liorente, I.M.: Performance-based scheduling strategies for HTC applications in complex federated grids. Concurr. Comput. Pract. Exper. **22**, 1416–1432 (2010)
78. Li, J., Khan, S.U., Ghani, N.: Semantics-based resource discovery in large-scale grids. In: Zomaya, A.Y., Sarbazi-Azad, H. (eds.) Large Scale Network-centric Computing Systems, chap. 17. Wiley, Hoboken (2013). ISBN: 978-0-470-93688-7

79. Li, C., Li, L.: Competitive proportional resource allocation policy for computational grid. Futur. Gener. Comput. Syst. Elsevier **20**(6), 1041–1054 (2004)
80. Lindberg, P., Leingang, J., Lysaker, D., Khan, S.U., Li, J.: Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. J. Supercomput. **59**(1), 323–360 (2012)
81. Li, F., Qi, D., Zhang, L., Zhang, X., Zhang, Z.: Research on novel dynamic resource management and job scheduling in grid compuing. In: Proceedings of 1^{st} IEEE International Multi-Symposiums on Computer and Computational Sciences, (IMSCCS'06), vol. 1, pp. 709–713 (2006)
82. Li, W., Xu, Z., Dong, F., Zhang, J.: Grid resource discovery based on a routing-transferring model. In: Proceedings of 3^{rd} International Workshop on Grid Computing (GRID '02), pp. 145–156. Springer, Baltimore (2002)
83. Ludwig, S.A., Santen, P.V.: A grid service discovery matchmaker based on ontology description. Euroweb 2002, The Web and the Grid, From E-science to E-business (2002)
84. Lynar, T.M., Herbert, R.D.: Allocating grid resources for speed and energy conservation. In: Proceedings of 6th International Conference on Information Technology and Applications, pp. 55–60. Vietnam (2009)
85. Lynar, T.M., Herbert, R.D., Chivers, W.J.: Simon: A grid resource allocation mechanism for heterogeneous e-waste computers. In: Proceedings of 7th Australian Symposium on Grid Computing and e-Research (AusGrid'09), pp. 69–76. Wellington (2009)
86. Lynar, T.M., Herbert, R.D., Simon Chivers, W.J.: Resource allocation to conserve energy in distributed computing. Int. J. Grid Util. Comput **2**(1), 1–10 (2011)
87. Manavalasundaram, V.K., Duraiswamy, K.: Association based grid resource allocation algorithm. Eur. J. Sci. Res. **78**(2), 248–258 (2012)
88. Maheswaran, M., Krauter, K.: A parameter-based approach to resource discovery in grid computing systems. In: 1st IEEE/ACM International Workshop on Grid Computing, pp. 181–190 (2000)
89. Marzolla, M., Mordacchini, M., Orlando, S.: Peer-to-peer systems for discovering resources in a dynamic grid. Parallel Comput. **33**, 339–358 (2007)
90. Mastroianni, C., Talia, D., Verta, O.: Designing an information system for grid: Computing hierarchical, decentralized P2P and super-peer models. Parallel Comput. **34**, 593–611 (2008)
91. McClatchey, R., Anjum, A., Stockinger, H., Ali, A., Willers, I., Thomas, M.: Data intensive and network aware grid scheduling. J. Grid Comput. **5**(1), 43–64 (2007)
92. Menasc'e, D., Casalicchio, E.: A framework for resource allocation in grid computing. In: Proceedings of IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '04), pp. 259–267. Volendam (2004)
93. Mehri Dehnavi, M., Fernandez, D.M., Giannacopoulos, D.D.: Enhancing the performance of conjugate gradient solvers on graphic processing units. IEEE Trans. Magn. **47**(5), 1162–1165 (2011)
94. Mehri Dehnavi, M., Fernandez, D.M., Giannacopoulos, D.D.: Enhancing the performance of conjugate gradient solvers on graphic processing units. IEEE Trans. Magn. **47**(5), 1162–1165 (2011)
95. Meisner, D., Gold, B., Wenisch, T.: PowerNap: Eliminating server idle power. In: Proceeding of 14th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 205–216. Washington (2009)
96. Meshkova, E., Riihijarvi, J., Petrova, M., Mahonen, P.: A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Comput. Netw. **52**, 2097–2128 (2008)
97. Min-Allah, N., Hussain, H., Khan, S.U., Zomaya, A.Y.: Power efficient rate monotonic scheduling for multi-core systems. J. Parallel Distrib. Comput. **72**(1), 48–57 (2011)
98. Mohammad, S.B., Khaoua, M.O., Ababneh, I.: An efficient non-contiguous processor allocation strategy for 2D mesh connected multicomputer. Int. J. Inf. Sci. **177**(14), 2867–2883 (2007)
99. Netto, M.A.S., Buyya, R.: Resource co-allocation in grid computing environment. In: A Handbook of Research on P2P and Grid Systems for Service Oriented Computing: Models, Methodologies, and Applications, pp. 476–494. IGI Global, New York (2010)
100. Newman, H.B., Legrand, I.C., Galvez, P.: MonALISA: A Distributed Monitoring Service Architecture, CHEP03, La Jolla, California, March 24–28 (2003)
101. Orgerie, A.C., Lefevre, L., Gelas, J.P.: Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. In: Proceedings of 14th IEEE International Conference on Parallel and Distributed Systems, pp. 171–178. Melbourne (2008)
102. Parashar, M., Hariri, S.: Autonomic computing: An overview. In: Lecture Notes Computer Science, vol. 3566, pp. 247–259. Springer, Berlin (2005)
103. Pinel, F., Pecero, J.E., Bouvry, P., Khan, S.U.: A two-phase heuristic for the scheduling of independent tasks on computational grids. In: ACM/IEEE/IFIP International Conference on High Performance Computing and Simulation (HPCS), pp. 471–477. Istanbul (2011)
104. Prodan, R., Wieczorek, M.: Negotiation-based scheduling of scientific grid workflows through advanced reservations. J. Grid Comput. **8**, 493–510 (2010)
105. Ranjan, R., Buyya, R., Parashar, M.: Special section on autonomic cloud computing: technologies, services, and applications (2011). doi:10.1002/cpe
106. Ranjan, R., Harwood, A., Buyya, R.: A taxonomy of peer-to-peer based complex queries: a grid perspective. arXiv:0610163 (2006)
107. Ranjan, R., Harwood, A., Buyya, R.: Peer-to-peer based resource discovery in global grids: A tutorial. IEEE Commun. Surv. Tutor. **10**(2), 6–33 (2008)
108. Rajan, A., Rawat, A., Verma, R.K.: Virtual computing grid using resource pooling. In: IEEE Proceedings of 2008 International Conference on Information Technology, (ICIT '08), pp. 59–64 (2008)

109. Rahman, M., Ranjan, R., Buyya, R., Benatallah, B.: A taxonomy and survey on autonomic management of applications in grid computing environments. Concurr. Comput. Pract. Exper **23**, 1990–2019 (2011)
110. Ribler, R.L., Vetter, J.S., Simitci, H., Reed, D.A.: Autopilot: Adaptive control of distributed applications. In: Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing (HPDC'98), pp. 172–179 (1998)
111. Sanaei, Z., Abolfazli, S., Gani, A., Buyya, R.: Heterogeneity in mobile cloud computing: Taxonomy and open challenges. In: IEEE Communications Surveys and Tutorials, ISSN: 1553-877X. IEEE Communications Society Press, USA (2013, In press)
112. Salapura, V., Bickford, R., Blumrich, M., Bright, A.A., Chen, D., Coteus, P., Gara, A., Giampapa, M., Gschwind, M., Gupta, M., Hall, S., Haring, R.A., Heidelberger, P., Hoenicke, D., Kopcsay, G.V., Ohmacht, M., Rand, R.A., Takken, T., Vranas, P.: Power and performance optimization at the system level," Proceedings of the 2nd Conference on Computing Frontiers (CF '05), pp. 125–132. Ischia (2005)
113. Said, M.P., Kojima, I.: S-MDS: Semantic monitoring and discovery system for the grid. J. Grid Comput. **7**(2), 205–224 (2009)
114. Saleh, A.I., Sarhan, A.M., Hamed, A.M.: A new grid scheduler with failure recovery and rescheduling mechanisms: discussion and analysis. J. Grid Comput. **10**(2), 211–235 (2012)
115. Shabtay, D.: Single and two-resource allocation algorithms for minimizing the maximal lateness in a single machine. J. Comput. Oper. Res. **31**(8), 1303–1315 (2004)
116. Yeo, C.S., Buyya, R.: A taxonomy of market-based resource management systems for utility-driven cluster computing. Softw. Pract. Exper. **36**(13), 1381–1419 (2006)
117. Sharma, A., Bawa, S.: Comparative analysis of resource discovery approaches in grid computing. J. Comput. **3**(5), 60–64 (2008)
118. Shiraz, M., Gani, A., Khokhar, R.H., Buyya, R.: A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. In: IEEE Communications Surveys & Tutorials, ISSN: 1553-877X. IEEE Communications Society Press, USA (2013, in press)
119. Shah, S.N.M., Mahmood, A.K.B., Oxley, A.: Hybrid resource allocation method for grid computing. In: Proceedings of the 2nd International Conference on Computer Research and Development (ICCRD'10), pp. 426–431. Kuala Lumpur (2010)
120. Shah, S.N.M., Mahmood, A.K.B., Oxley, A.: modified least cost method for grid resource allocation. In: Proceedings of 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC '10), pp. 218–225. Huangshan (2010)
121. Shao, B., Rao, R.: A parallel hypercube algorithm for discrete resource allocation problems. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **36**(1), 233–242 (2006)
122. Sharma, R., Soni, V.K., Mishra, M.K., Bhuyan, P.: A survey of job scheduling and resource management in grid computing. World Acad. Sci. Eng. Technol. **64**, 461–466 (2010)
123. Sih, G.C., Lee, E.A.: A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Trans. Parallel Distrib. Syst. **4**(2), 175–187 (1993)
124. Sim, K.M.: A survey of bargaining models for grid resource allocation. ACM SIGecom Exch. **5**(5), 22–32 (2006)
125. Skillicorn, D.B.: Motivating computational grids. In: Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '02), pp. 401–406 (2002)
126. Somasundaram, T.S., Balachandarl, R.A., Kandasamy, V., Buyya, R., Raman, R., Mohanram, N., Varun, S.: Semantic-based grid resource discovery and its integration with the grid Service broker. In: International Conference on Advanced Computing and Communications (ADCOM), Surathkal, pp. 84–89. (2006)
127. Somasundaram, K., Radhakrishnan, S.: Task resource allocation in grid using swift scheduler. Int. J. Comput. Commun. Control **IV**(2), 158–166 (2009)
128. Tangmunarunkit, H., Decker, S., Kesselman, C.: Ontology-based resource matching in the grid –The grid meets the semantic web. In: Fensel, D., et al. (eds.) SWC 2003, LNCS, pp. 706–721 (2870)
129. Tesauro, G., Das, R., Chan, H., kephart, J.O., Lefurgy, C., Levine, D., Rawason, F.: Managing power consumption and performance of computing systems using reinforcement learning. In: Proceedings of 21st Annual Conference on Neural Information Processing Systems. Vancouver (2007)
130. Thayananthan, V., Alzahrani, A., Qureshi, M.S.: Analysis of key management and quantum cryptography in RFID networks. Int. J. Acad. Res. Part A **4**(6), 145–150 (2012)
131. Thenmozhi, S., Tamilarasi, A.: A hierarchical trusted resource allocation architecture for mobile grid environment. Eur. J. Sci. Res **59**(4), 510–521 (2011)
132. Trunfio, P., Talia, D., Papadakis, H., Fragopoulou, P., Mordacchini, M., Pennanen, M., Popov, K., Vlassov, V., Haridi, S.: Peer-to-peer resource discovery in grids: models and systems. Futur. Gener. Comput. Syst **23**, 864–878 (2007)
133. Tom, A., Murthy, S.R.: An improved algorithm for module allocation in distributed computing systems. J. Parallel Distrib. Comput. **42**(1), 82–90 (1997)
134. Verma, A., Ahuja, P., Neogi, A.: PMAPPER: Power and migration cost aware application placement in virtualized systems. In: Proceedings of the 9th ACM/IFIP/USENIX International Middleware Conference, vol. 5346, pp. 243–264. Leuven (2008)
135. Verma, A., Ahuja, P., Neogi, A.: Power-aware dynamic placement of HPC applications. In: Proceedings of 22nd Annual International Conference on Supercomputing, pp. 175–184. Athens (2008)
136. Vivekananth: Trusted resource allocation in grid computing by using reputation. Int. J. Comput. Sci. Commun. **1**(2), 23–25 (2010)
137. Wang, L., Lu, Y.: Efficient power management of heterogeneous soft real-time clusters. In: Proceedings of

IEEE 2008 Real-Time Systems Symposium, pp. 323–332. Barcelona (2008)
138. Wang, L., Tao, J., Marten, H., Streit, A., Khan, S.U., Kolodziej, J., Chen, D.: MapReduce across distributed clusters for data-intensive applications. In: 26^{th} IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 2004–2011. Shanghai (2012)
139. Wan, L., Xie, Z., Wu, L., Lin, J.: Research on the key technologies of geospatial information grid service workflow system. In: IEEE 18th International Conference on Geoinformatics, pp. 1–5. Beijing (2010)
140. Wu, T., Ye, N., Zhang, D.: Comparison of distributed methods for resource allocation. Int. J. Prod. Res. **43**(3), 515–536 (2005)
141. Yeo, C.S., Buyya, R., Assunção, M.D., Yu, J., Sulistio, A., Venugopal, S., Placek, M.: Utility computing on global grids, chap 143. In: Bidgoli, Hossein (ed.) The Handbook of Computer Networks, ISBN: 978-0-471-78461-6. Wiley, New York (2007)
142. Yousif, A., Abdullah, A.H., Latiff, M.S.A., Bashir, M.B.: A taxonomy of grid resource selection mechanism. Int. J. Grid Distrib. Comput. **4**(3), 107–118 (2011)
143. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. J. Grid Comput. **3**, 171–200 (2005)
144. Yu, D., Robertazzi, T.G.: Divisible load scheduling for grid computing. In: Proceedings of 15th International Conference on Parallel and Distributed Computing and Systems (PDCS'03) (2003)
145. Zanikolas, S., Sakellariou, R.: A taxonomy of grid monitoring systems. Futur. Gener. Comput. Syst. **21**, 163–188 (2005)
146. Zhi-jie, L., Cun-rui, W.: Resource allocation optimization based on load forecast in computational grid. Int. J. Eng. Res. Appl. (IJERA) **2**(3), 1353–1358 (2012)
147. Zhang, X., Freschl, J.L., Schopf, J.M.: A performance study of monitoring and information services for distributed systems. In: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, (HPDC '03), pp. 270–281 (2003)