A Survey of Task Allocation and Load Balancing in Distributed Systems

Yichuan Jiang, *Senior Member, IEEE*

Abstract—In past decades, significant attention has been devoted to the task allocation and load balancing in distributed systems. Although there have been some related surveys about this subject, each of which only made a very preliminary review on the state of art of one single type of distributed systems. To correlate the studies in varying types of distributed systems and make a comprehensive taxonomy on them, this survey mainly categorizes and reviews the representative studies on task allocation and load balancing according to the general characteristics of varying distributed systems. First, this survey summarizes the general characteristics of distributed systems. Based on these general characteristics, this survey reviews the studies on task allocation and load balancing with respect to the following aspects: 1) typical control models; 2) typical resource optimization methods; 3) typical methods for achieving reliability; 4) typical coordination mechanisms among heterogeneous nodes; and 5) typical models considering network structures. For each aspect, we summarize the existing studies and discuss the future research directions. Through the survey, the related studies in this area can be well understood based on how they can satisfy the general characteristics of distributed systems.

Index Terms—Distributed systems, task allocation, load balancing, networks, resource allocation, survey, taxonomy

1 Introduction

Distributed systems, in which the distributed computational units are connected and organized by networks to meet the demand of large-scale and high performance computing, have received considerable attention over the past decades [1], [2], [3]. There are many types of distributed systems, such as grids [4], peer-to-peer (P2P) systems [5], ad hoc networks [6], cloud computing systems [7], pervasive computing systems [8], and online social network systems [9].

Currently, the applications in distributed systems are various, such as web service, scientific computation, and file storage. In general, an application in a distributed system can be divided into a number of tasks and be executed on different nodes; the performance of an application in a distributed system is dependent on the allocation of the tasks comprising the application onto the available nodes, referred to as the *task allocation problem* [10], [11]; if too many tasks are crowded on certain nodes, tasks could be switched from heavy-burdened nodes to light-burdened ones to reduce the waiting time of tasks at nodes, which is called *load balancing* [4]. It is commonly stated that task allocation and load balancing are crucial to the distributed systems [12]. In past decades there are significant amount of studies for this area, which could cause people to be puzzled by the enormous amount of wide variety of research results.

Although there have been some related surveys previously, but most of which only made a very preliminary review on the state of art of one single type of distributed system, such as the survey of load balancing in grids [91], [92], the survey of load balancing in cloud computing [93], [94], and the survey of load balancing in P2P systems [95]. Then, how to correlate the related studies in varying types of distributed systems and make a general taxonomy on them?

To solve the above problem, first this survey paper summarizes the typical characteristics of general distributed systems, such as the distributed control, resource distribution, open environments, heterogeneous nodes, and network structures. Then, the survey paper categorizes the existing studies on task allocation and load balancing based on how they can satisfy the characteristics of distributed systems, which includes the following aspects: control models, resource optimization, ensuring reliability, coordination mechanisms, and measures to consider network structures. With this survey, a general and macroscopic review of task allocation and load balancing for varying types of distributed systems can be achieved, which can have higher generality much than the previous preliminary surveys for only one single type of systems.

The remainder of this survey is organized as follows. In Section 2, we discuss the problems and objectives of task allocation and load balancing in distributed systems; in Section 3, we review the control models of task allocation and load balancing; in Section 4, we review the resource optimization methods in task allocation and load balancing; in Section 5, we review the reliability of task allocation and load balancing; in Section 6, we review the coordination among heterogeneous nodes for task allocation and load balancing; in Section 7, we review the related work considering network structures; and finally, we conclude our survey in Section 8.

• *The author is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China, and the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: yjiang@seu.edu.cn.*

Manuscript received 29 Oct. 2014; revised 23 Jan. 2015; accepted 23 Feb. 2015. Date of publication 26 Feb. 2015; date of current version 20 Jan. 2016.
Recommended for acceptance by A. Dubey.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2407900

2 The Problems and Objectives of Task Allocation and Load Balancing in Distributed Systems

2.1 Problems

2.1.1 Overview on Distributed Systems

Currently, a large number and variety of applications have been extensively developed for distributed systems linking computers and other computing devices together in a seamless and transparent way. In distributed systems, various nodes act autonomously and cooperate with each other, which can achieve the purposes of resource sharing, openness, concurrency, scalability, fault-tolerance, and transparency [16], [17]. In summary, real distributed systems have the following general characteristics.

- *Lack of a global control unit*. Due to the large-scale, dynamic, and heterogeneous nature of distributed system [17], it is difficult to implement a global control unit for the entire system. For example, in ad hoc networks, wireless nodes can communicate with each other in the absence of a fixed infrastructure and without central control [18]; in sensor networks, many sensor nodes are densely deployed and may not have global identifications because of the large amount of overhead and large number of sensors [19]; in P2P systems, a decentralized architecture is applied in which individual nodes in the network act as both suppliers and users of resources, in contrast to the centralized client-server model where client nodes request access to resources provided by central servers [5].
- *Distribution and sharing of resources*. There are various resources in the systems, such as data, replicas of popular web objects, processors, computational capacity, and disk storage. Resources are distributed among nodes, and nodes often agree to share their local resources with each other to implement tasks [20]. To implement the effective sharing of resources, resource management is very crucial, which often includes resource placement, resource provision, resource discovery, and resource negotiation [5], [21], [22]. Moreover, resource caching is often used to improve the performance of resource access, where some nodes cache the resources to serve future requests instead of fetching resources from the original nodes [23].
- *Openness and unreliability of systems*. Distributed systems are often open, and some nodes and network structures may be unreliable. For example, there are no efficient ways to prevent malicious peers from joining open systems, thus the distributed systems are very vulnerable to abuses by selfish and malicious users [24]. Moreover, due to the autonomy of nodes, some nodes may fail to operate independently. Therefore, reliability and fault-tolerance are crucial for real open distributed systems.
- *Heterogeneous nodes*. In some distributed systems, the nodes are heterogeneous. For example, in grids, the nodes may be attributed to different organizations [20]; in sensor networks, the nodes may have different capacities and sensing ranges [19], [25]; in social network systems, the nodes may have different response times and thresholds for diffusion [26]. When tasks are executed, heterogeneous nodes should coordinate with each other to maximize the total utilities. Moreover, heterogeneous nodes may compete for critical resources when they execute tasks.
- *Constraints of network structures*. In a distributed system, all nodes are constrained by the network structure. In the network structure, each node can only communicate with its neighboring nodes; the communication between two non-neighboring nodes should be relayed by other intermediate nodes [1], [11]. Therefore, the system performance is significantly influenced by the network structures.

Accordingly, the tasks in distributed systems always encounter the following situations:

- *The tasks may be implemented by negotiation among nodes without a central control*. In a distributed system, the tasks that require more than one node may be implemented in a manner such that many nodes coordinate with each other and negotiate resources. Because distributed systems often lack a global control unit, as stated above, the negotiation in the task allocation may be implemented by nodes autonomously.
- *The tasks are always implemented by accessing required resources distributed in the networks*. To execute tasks efficiently, one of the key elements is to improve the performance of accessing the resources necessary for the tasks. Therefore, resource optimization in networks can significantly influence the performance of tasks.
- *The tasks may be implemented unreliably*. Distributed systems are open and may be invaded by malicious users [9]; moreover, there may be unreliable resources and communications in distributed systems [16]. Therefore, the tasks may be implemented unreliably, and sometimes the desired results may not be achieved [27].
- *The tasks performed by heterogeneous nodes may vary significantly and compete for critical resources.* Because the nodes are heterogeneous and the number of users may vary dynamically in real time, the tasks may vary significantly at different times and at different nodes. Moreover, the users at different nodes are often self-motivated and will attempt to maximize their own benefits. Therefore, the execution of tasks of different users may compete for critical resources, which may bring about conflicts in the systems.
- *The performance of tasks may be influenced significantly by network structures.* First, the network structures may influence the resource access performance for tasks; thus, the resource optimization in task allocation should consider the network structures. Second, the interaction and coordination among nodes often obey certain network structures, thus a suitable network structure can improve the coordination performance of nodes in task allocation.

2.1.2 Task Allocation and Load Balancing in Distributed Systems

Without loss of generality, the tasks in distributed systems are always implemented by accessing required resources in the networks [1], [28], [29]. For example, a file storage task will seek the required memory resources in the network. Therefore, many existing models of task allocation and load balancing have been implemented based on the accessibility of required resources [11], [27], [28], [29], i.e., a node's probability of being allocated tasks is determined by its accessibility to required resources for the tasks. We now provide the formal definition of task allocation in distributed systems, which is implemented based on the resource requirements of tasks.

Definition 1 (Task allocation in distributed systems [27]) *Given a distributed system, $N = <A, E>$, where A is the set of nodes and each node owns a different set of resources, and $\forall <a_i, a_j> \in E$ indicates the existence of a network link between node a_i and a_j, the set of resources in node a_i is assumed to be R_{ai}, and the set of resources required by task t is assumed to be R_t. If task t arrives at the network, the task allocation in N can be defined as the mapping of task t to a set of nodes, A_t, which can satisfy the following situations:*

1) *The resource requirements of t can be satisfied, i.e., $R_t \subseteq \cup_{\forall a_i \in A_t} R_{a_i}$;*
2) *The predefined objective can be achieved by the task execution of A_t, such as minimizing the execution time [85] or maximizing the reliability [86].*
3) *The nodes in A_t can execute the allocated tasks under the constraint of the network structure, e.g. $\forall\ a_t, a_j \in A_t \Rightarrow P_{ij} \subseteq E$, where P_{ij} denotes the interaction path between a_i and a_j.*

According to Definition 1, if a node has a higher probability of accessing the necessary resources for tasks, it may be allocated more tasks. However, if too many tasks are crowded on certain nodes with high probabilities of accessing the tasks' required resources, the tasks will require significantly more time to wait for the necessary resources [1], [4], [11]. Therefore, we should now apply load balancing to the task allocation.

Definition 2 (Load balancing in task allocation [27]. *Given a distributed systems, $N = <A, E>$, where A is the set of nodes, $\forall a_i \in A$, the team of tasks that queue for resource r_k of node a_i can be denoted as Q_{ik}. The size of Q_{ik} is s_{ik} and the processing capability of a_i is v_i; the original probability of node a_i to receive tasks (which need k type resources) is $P_i(k)$, which can be calculated by considering the resources of a_i. We should perform load balancing when s_{ik} is too large, which is implemented by discounting the probability of a_i's receiving new tasks, $P_i(k)$, to a revised probability, $DP_i(k)$:*

$$DP_i(k) = \psi(s_{ik}/v_i) \cdot P_i(k), \tag{1}$$

where ψ is an attenuation function, $0 \leq \psi \leq 1$; the value of $\psi(s_{ik}/v_i)$ decreases monotonically from 1 to 0 as s_{ik}/v_i increases.

2.1.3 Typical Problems

Because of the typical characteristics of distributed systems stated in Section 2.1.1, some typical problems occur during the task allocation and load balancing, shown as follows.

- *Control models*. To achieve the globally optimal result for task allocation and load balancing, a centralized control model is required to collect the status information of the entire system in real time; however, it is difficult to implement such totally centralized control model because distributed systems are always large, dynamic, and lacking global control units. By contrast, the totally decentralized approach may require relatively high computational costs from the nodes, which may place heavy loads on large systems and make the task allocation processes difficult to control effectively [27].
- *Resource optimization*. Many existing models of task allocation and load balancing have been implemented based on the optimization of accessibility of required resources [11], [27], [28], [29]. Then, how to measure the resource accessibility and how to optimize such a performance index are key problems. For example, the following two typical situations are always observed in reality: 1) if a node has plentiful resources, it may have high resource accessibility; 2) if a node does not have plentiful resources by itself, but it can obtain enough resources from other interacting nodes easily, then it can also be allocated many tasks. Therefore, the optimization for resource accessibility should not only consider the access performance of nodes' own resources but also that of nodes' contextual resources [11].
- *Reliability.* In open distributed systems, some nodes may be unreliable. Therefore, a key problem is how to guarantee reliable resource access as well as optimize the resource access performance [27]. Therefore, we should design some reliability-oriented approaches to find a task allocation result such that the reliability is maximized [30], [31].
- *Coordination among heterogeneous nodes.* Heterogeneous nodes do not cooperate in making decisions while they execute tasks [32]; each node may optimize its own resource access performance independently of the others. Therefore, how to make these noncooperative nodes coordinate to improve the overall performance of tasks should be well-solved; moreover, how to incentivize self-interested nodes to contribute their resources to others is also a key problem [28].
- *Considering network structures.* The communication and interaction of nodes are constrained by the network structures. Then, how to measure the influence of network structures on the performance of tasks is a problem. The task allocation and load balancing should consider the localities of nodes in the network structures.

2.2 Objectives

There are varying objectives of task allocation in existing studies: *minimize the response time of tasks, minimize the* *makespan of tasks, maximize the throughput of tasks, and maximize the reliability of tasks.*

Response time of a task is the time elapsed between its arrival and its initial execution. Minimizing the response time of tasks means minimizing the waiting time of tasks at the allocated nodes, which depends on the current loads of the nodes [33]. Two typical situations are minimizing the total response time of all tasks and minimizing the mean response time of all tasks.

Makespan minimization is the most widely applied optimization objective in the task allocation domain. The makespan is defined as the time span between the start of the first task to be handled and the end of all tasks to be finished, which denotes the total time for handling the entire set of tasks [4], [34]. The makespan often includes the following aspects: the time to transfer tasks to different nodes, the waiting time of tasks at nodes, the time for accessing resources, the time for communicating resources, the time for processing, and so on.

Throughput, a suitable performance metric when the application is to compute a large number of tasks, is defined as the number of tasks completed by the system per time unit under steady state condition [14] or the total workload of completed tasks per time unit [35].

The reliability of task allocation is defined as the probability that the tasks can be allocated to nodes and executed successfully [10], [27]. Generally, there are two types of reliabilities in distributed systems [10]: one is the node-related reliability, such as the reliability of resources and the reliability of computation; another is the path-related reliability, such as the reliability of communication paths among allocated nodes.

In reality, it is difficult to satisfy all of the above objectives simultaneously. For example, to improve the reliability, the redundancy mechanism is often used but which may increase the makespan of tasks. Therefore, a compromise should be made among those objectives.

It is noticeable that, load balancing is not only a mechanism but also an objective of task allocation, which can optimize the response time, makespan, or throughput of tasks. For example, to minimize response time, a load balancing mechanism can be used to transfer tasks from heavily loaded nodes to lightly loaded nodes [4], [11], [27]. As the response time is reduced, the makespan of all tasks can be reduced accordingly so that the amount of tasks completed by the system per time unit can also be improved.

3 Control Models in Task Allocation and Load Balancing

Control models represent how to control the entire processes of task allocation and load balancing. In general, there are two prevalent control models in existing studies; one is the *centralized control model*, and the other is the *decentralized control model*.

The centralized model uses a central controller that should know the status information of the entire system in real time [36]. The central controller will make all decisions based on the information that is sent from other nodes [37]. The centralized model can be implemented easily, however, it may be sometimes infeasible in real distributed systems that are large and dynamic, where the global information cannot be achieved in real time; moreover, the central controller may become a performance bottleneck of the system.

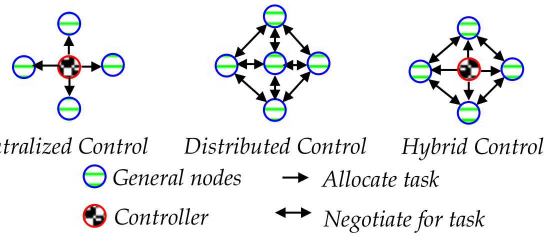

Fig. 1. A simple illustration for the three control models.

In contrast, the distributed model can be used in those large and dynamic distributed systems, where the tasks can be allocated by nodes themselves and the global control unit is not needed; the nodes make their decisions based on their own perceived information about the system. Fault-tolerance can be achieved because each node can act as the allocator. However, a drawback of the distributed model is that the nodes face relatively high computational costs which may place heavy loads on large systems and make the task allocation processes difficult to control effectively [27].

To solve the problems of totally centralized and totally distributed models, a few hybrid control models, which can combine the centralized and decentralized models within a system, are used. Fig. 1 is a simple description of these three control models.

3.1 Centralized Control Model

Generally, the centralized control model is often employed in small systems, and it uses a global model to calculate allocations [39]. For example, in a small private grid, the system provides the computation and data storage as a closed and private resource, and it is often controlled by a central controller [38].

Lin and Raghavendra [40] proposed a dynamic load-balancing policy with a central job dispatcher called the LBC policy for distributed systems, which is expected to provide near-minimum average task response time for distributed systems with high speed communication subnets. The LBC policy in [40] makes load-balancing decisions based on global state information, which has centralized information and location rules. Godfrey et al. [41] stored load information of the peer nodes in P2P systems in a number of directories that periodically schedule reassignments of virtual servers to achieve better balancing, and they reduced the distributed load balancing problem to a centralized problem at each directory.

In summary, although the centralized control model is simple and can achieve the globally optimal result, it is often infeasible in reality because only a small amount of real distributed systems are static. Therefore, in related studies there are only a small amount of studies that adopt fully centralized control models. Moreover, the centralized control model is sometimes used as a baseline to compare with the distributed control model [38].

3.2 Distributed Control Model

In the distributed control model, the negotiation and coordination among nodes for tasks are crucial. To implement effective negotiation and coordination among autonomous nodes, some economically and socially inspired mechanisms are often adopted. Therefore, we can now categorize the related studies according to their adopted negotiation and coordination mechanisms. In summary, the existing related studies include the following: 1) *market-based approaches*, such as game theory, auctions, and contract mechanisms; 2) *society-based approaches*, such as strategy diffusion, coalition formation, group mechanisms, and the Matthew effect; 3) *agent-based approaches*, which utilize the autonomous modeling and distributed computation ability of agents to perform task allocation and load balancing; and 4) *other alternative approaches for various specific objectives*, where nodes will negotiate and coordinate to achieve some specific objectives, such as higher reliability or lower communication costs.

3.2.1 Market-Based Approaches

The market mechanism is a term from economics referring to the exchange within a system of value and time tradeoffs to produce the best distribution of goods. In distributed systems, the nodes are often independent and make decisions autonomously based on their policies and the market-like techniques are often used to perform task allocation and load balancing.

The nodes often have specialized knowledge about their own capabilities but limited knowledge about other individuals and the large-scale structure. To solve that problem, Walsh and Wellman [42] presented a decentralized market protocol for allocating tasks among nodes that contend for scarce resources, where nodes trade tasks and resources at prices determined by an auction protocol. Moreover, Ghosh et al. [43] proposed a game theory pricing model to address load balancing issues in mobile grids, which maximizes the revenue of the grid user and yet is comparable to other load balancing schemes in terms of the overall system response time.

Grosu and Chronopoulos [32] presented a game theory framework for obtaining a user-optimal load balancing scheme in heterogeneous distributed systems, and they derived a new distributed load balancing algorithm based on the structure of the Nash equilibrium; such approach can achieve the advantage of distributed structure, low complexity, and optimality of allocation for each user. Penmatsa and Chronopoulos [44] formulated the load balancing problem as a non-cooperative game among the users who try to minimize the expected response time of their own tasks, which can provide fairness to all the users in the system.

Auction is another used market-like technique for the task allocation and load balancing in distributed systems that describes the mechanism or set of negotiation rules for tasks. For example, Izakian et al. [45] introduced a continuous double auction method for grid resource allocation in which resources are considered as provider agents and users as consumer agents, which is efficient in terms of successful execution rates, resource utilization rates, and fair profit allocation. Choi et al. [46] presented the consensus-based auction algorithm to perform decentralized task allocation, which utilizes a market-based decision strategy as the mechanism. Garg et al. [47] proposed a novel meta-scheduler based on the double auction mechanism.

3.2.2 Society -Based Approaches

There are many societal mechanisms, such as strategy diffusion and coalition formation, which describe the collective decision making of actors to coexist.

Strategy diffusion is a common phenomenon in the collective motion of society, which denotes the large scale of strategy penetrations of certain agents on other agents [50]. In distributed systems, the negotiation between neighboring nodes for tasks can be described as a diffusion process; for example, Rotaru and Nägeli [48] introduced the diffusion mechanism into the dynamic load balancing in heterogeneous systems and gave a direct explicit expression of the balancing flow generated by a generalized diffusion algorithm. The advantage of the diffusion mechanism is that each node only knows the information of neighbor nodes and the global optimal result can be achieved finally.

A coalition is an alliance among individuals, in which individuals cooperate in joint action. In distributed systems, some nodes can form a coalition to execute a task if such a task cannot be performed by a single node [90]. For example, Shehory and Kraus [49] presented algorithms that enable the nodes to form coalitions and assign a task to each coalition, and these algorithms have low ratio bounds and low computational complexities. Low et al. [51] presented a task allocation scheme via self-organizing swarm coalitions for distributed mobile sensor network coverage, which uses the concepts of ant behavior to self-regulate the regional distributions of sensors in proportion to that of the moving targets to be tracked. Wang and Jiang [52] proposed a community-aware task allocation model for social network systems where a node's cooperation domain is constrained in a community and each node can negotiate only with its intracommunity member nodes; the proposed model performs very closely to the benchmark exponential brute-force optimal and the network flow-based greedy algorithm in terms of overall system profit.

Moreover, there are a few studies that utilize other typical social mechanisms for task allocation. For example, Jiang and Huang [1] used the Matthew effect (i.e., the rich get richer mechanism) for the task allocation of distributed systems with resource caching, where the nodes that were heavily burdened by tasks may have certain preferential rights to new tasks in the future. Through experimental validation, it was proven that such Matthew effect-based task allocation can effectively reduce tasks' execution time.

In summary, the aforementioned studies are still in the preliminary stage of purely applying some special societal mechanisms, and the final performance often depends on the matching degree between the societal mechanisms and the system environments. In the future, we think the learning and self-adaption in society should be introduced to implement a real self-adaptable task allocation for dynamic and complex systems. Moreover, a systematic methodology for applying societal mechanisms should be developed.

3.2.3 Agent-Based Approaches

In the agent-based approaches, the agents are distributed in the network and span across the nodes; the agents can interact with nodes and cooperate with other agents in order to solve complex problems for tasks [53]. Totally, autonomous agents can observe their surrounding environment and perform simple local computations leading to actions of task allocation and load balancing.

Liu et al. [4] provided an agent-based decentralized task allocation mechanism for minigrids. In the proposed model, an agent will be automatically dispatched to a task immediately after the task is randomly submitted to a node; then, the agent will carry the task to search for an appropriate node and the dispersion of agents can realize load balancing. To reduce the waiting time of the task, the agent will take the task to a node where the queue of waiting agents is small. In the model, agents' behaviors can be classified into two types, leaving and queuing; an agent carrying a task will leave a node if it finds there are too many agents queuing at such node; an agent carrying a task will decide to queue at a node if it finds the number of already queuing agents at such node acceptable.

Chow and Kwok [54] proposed a multiagent-based load balancing model for a cluster of workstations, where the agents can span across all the machines of a cluster. The proposed algorithm works by associating each agent with a credit value that depends on the agent's affinity to a machine, the agent's current workload, the agent's communication behavior, the agent's mobility, etc. To realize load balancing among workstations, the credits of agents are examined, and the agent with a lower credit value is migrated to a relatively lightly loaded workstation.

3.2.4 Other Alternative Approaches for Various Specific Objectives

There are some distributed control approaches that cannot be categorized into the above three types; now, we also introduce some representative studies of them here.

Di and Wang [35] proposed a novel autonomous resource allocation scheme to maximize the throughput of self-organizing P2P grid systems, which have high adaptability to dynamic environments by proactive and convex-optimal estimation of nodes' volatile states. Hong and Prasanna [55] considered the task allocation problem for computing large sets of equal-sized independent tasks on heterogeneous computing systems, and developed a decentralized adaptive algorithm for adaptive task allocation. Beaumont et al. [56] presented a distributed task scheduler to maximize the throughput of each application while ensuring a fair sharing of resources between applications.

Antonis et al. [57] proposed an adaptive distributed hierarchical scheme, the Virtual Tree Algorithm (VTA), which creates a virtual binary tree structure over the actual network topology. Shah et al. [58] proposed adaptive and decentralized load balancing algorithms to address several issues that are imperative to grid environments such as handling resource heterogeneity and sharing, and communication latency.

3.2.5 Summary for Distributed Control Model

In summary, the main characteristic of distributed control model is that the nodes can coordinate with each other autonomously for task allocation. With the autonomous and distributed task allocation, the control can be adapted for varying system scales and the failure of some nodes, thus the robustness is high. However, as each node can only know the information about neighbors, the results may be only locally optimal but not globally optimal. Moreover, the negotiation among nodes may produce additional computation costs.

3.3 Hybrid Control Model

It is generally recognized that the distributed control model is implicitly designed for infinitely large distributed systems, while small systems are regarded as being controllable using the traditional centralized model. However, some real systems do not fit conveniently into these "small" or "large" categories [39]; particularly, some systems may be dynamic, and the scales especially may be variable during operation. Therefore, it is now difficult to decide whether centralized or distributed control models are used. Moreover, totally centralized models and totally decentralized models both have their respective drawbacks.

To solve the above problems, we can consider integrating the centralized and distributed control models in a system. For example, Zaki et al. [59] presented four strategies based on control models (either centralized or distributed) and the required information (either global or local): global centralized load balancing, global distributed load balancing, local centralized load balancing, and local distributed load balancing. They integrated those four strategies and presented a customized load balancing scheme.

Jiang and Jiang [11] provided a spectrum between a totally centralized approach and a totally decentralized approach for performing task allocation: the centralized heuristic is utilized to control the overall status information, and the distributed heuristic is utilized to achieve the flexibility of task allocation. The approach in [11] is realized as follows: a node is allocated as a manager for a task using a centralized heuristic, and, if the manager lacks the necessary resources, it negotiates with other nodes for contractors using a distributed heuristic; finally, the manager and the contractors constitute the allocated nodes. Then, in [27], Jiang et al. substantially extended the hybrid control architecture in [11] by taking into account the nodes' negotiation reputations during task allocation, where tasks can obtain dependable resources in the least access time.

Moreover, Abdallah and Lesser [60] proposed a mediator-agent architecture for hybrid control of task allocation, where some nodes are mediators and other nodes are general agents. The task allocation process is as follows: a mediator receives task announcements with an associated payoff; then, the mediator will decompose the task into a set of subtasks and contract out subtasks to neighboring agents, independent of whether the other agents are mediators or not.

3.4 Summary and Future Research Directions

In summary, the comparisons among the three control models can be shown as Table 1. From the table, we can see that each control model has its characteristics and application environments.

Next, we discuss some challenges in existing studies and present some insights on future research directions on the

TABLE 1
The Comparisons among the Three Control Models

Control Models	Controller	Functions of Nodes	System Scale	Performance	Robustness
Central control	There is one central controller	All nodes are passive for task allocation	Small	Globally optimal	Poor
Distributed control	No controllers	All nodes are autonomous for task allocation	Large	Locally optimal	Good
Hybrid control	Some nodes can act as controllers	Nodes can act as either passive or autonomous role	Medium	Medium	Medium

control models of task allocation and load balancing in distributed systems, which are shown as follows.

- In some current distributed systems, such as cloud computing systems [77], there are customized services; accordingly, the control model may also be dynamically customized by customers. Therefore, how to customize control models to satisfy the requirements of customers is a future research issue.
- Current large distributed systems may be composed of several subsystems, and each subsystem may have its own unique control model such that there is more than one type of control model in the overall system. Therefore, how to coordinate the different control models of different subsystems to satisfy the requirements of the overall system is a research challenge.
- Current distributed systems may often be dynamic. Therefore, a static control model may be unsuitable. In the future, the self-adaptation and evolution of control models for dynamic distributed systems should be investigated.

4 Resource Optimization in Task Allocation and Load Balancing

In distributed systems, some resources are placed at the nodes within the networks and can be accessed and shared by users to execute tasks [14], [32]; typical resources are computation resources, communication resources, data resources, and storage resources. Without loss of generality, task execution can be described through nodes' operations when accessing necessary resources distributed in the networks [1], [5], [6], [27], [28]; also, task allocation often means allocating tasks to resources [34], [35].

To achieve good performance of tasks, many existing studies have been implemented based on resource optimization [11], [27], [28], [29]. The related works of resource optimization mainly considered two types of resources: 1) *a self-owned resource-based approach* implemented based on nodes' self-owned resource statuses, i.e. a node's probability of being allocated tasks is determined only by its self-owned resources, and the performance of tasks is mainly related to the allocated nodes' own resource statues; and 2) *a contextual resource-based approach* implemented based on not only nodes' self-owned resource statuses but also their contextual resource statuses because nodes may cooperate with others within their contexts, and the performance of tasks are related not only to the allocated nodes' resource statuses but also their contextual resource statuses.

In existing related studies, there are many optimization objectives, such as minimizing the resource access time, maximizing the resource access reliability, and minimizing the resource access conflicts of different tasks.

4.1 Self-Owned Resource-Based Optimization

In some systems, the resources are often dynamic and of great heterogeneity; moreover, the resources may become unreliable due to disconnection of power or communication. To optimize the resource allocation for tasks in grids, Xu et al. [34] presented a Chemical Reaction Optimization algorithm inspired by the interactions between molecules in a chemical reaction, which can optimize the resources for tasks with three objectives: makespan, flowtime, and reliability. Letting *Makespan* be the completion time of the last finished task, *Flowtime* be the total time consumed by all tasks, and *T-aborted* represent the total wasted time caused by the aborted tasks in terms of reliability, the objective of resource optimization for tasks in [34] is to:

$$\min(\alpha \times Makespan + \beta \times (Flowtime/m) + \gamma \times (T - aborted/m)), \tag{2}$$

where $\alpha + \beta + \gamma = 1$ and $1 \geq \alpha, \beta, \gamma \geq 0$, and m is the number of resources. *Flowtime* and *T-aborted* have higher order of magnitude than *Makepsan* and, thus, are normalized by m.

Izakian et al. [45] presented a continuous double auction method to perform the resource optimization for tasks in grids, which can achieve the objective of successful execution rates and resource utilization rates. Das and Grosu [61] proposed a combinatorial auction-based resource allocation protocol in which a user bids a price value for each of the possible combinations of resources required for its task's execution. Xue et al. [6] proposed a new price-based resource optimization framework in wireless ad hoc networks to achieve optimal resource utilization and fairness among competing end-to-end flows.

Moreover, some unpredictable situations may take place in open and dynamic distributed systems. Therefore, a robust resource optimization method for tasks is needed. Sugavanam et al. [62] studied the problem of finding a static resource allocation for tasks to maximize the robustness of makespan against the errors in task execution time estimates. Jiang and Jiang [12] considered the variation of underlying network topologies of systems and presented a robust resource optimization model; the provided model takes into account the factors of network topologies and node distributions and implements effective task allocation and resource negotiation for the current network topology;

TABLE 2
The Comparisons between the Two Resource Optimization Methods

Methods	Basic principle	Advantages	Disadvantages	Application Environments
Self-owned resource-based optimization	A node's probability of being allocated tasks is determined only by its self-owned resources; thus only the allocated nodes' own resources are optimized.	Easy to implement and the information for self-owned resources can be easily achieved	The resource access performance may be poor if the allocated nodes cannot finish the tasks by themselves	Nodes often execute the tasks autonomously
Contextual resource-based optimization	A node's probability of being allocated tasks is determined by not only its own resources but also the resources in its contexts; thus both the allocated nodes' own resources and their contextual resources are optimized.	Nodes contribute their resources for cooperating to execute tasks; the resource access time among allocated nodes can be optimized	A large amount of computations are needed and high costs may be produced	Cooperative distributed systems in which many nodes often cooperate for executing tasks

finally, robustness and adaptation for the dynamic underlying network topologies can be achieved.

In summary, the self-owned resource-based optimization methods only consider the resources of nodes themselves, thus the coordination among nodes for resources may be neglected. Therefore, such method may perform well only in the situations where most nodes can finish the allocated tasks autonomously.

4.2 Contextual Resource-Based Optimization

In distributed systems, nodes often need to negotiate with other nodes within their contexts when they execute tasks. Therefore, the capacity of a node to execute tasks is determined by not only its own resources but also its contextual nodes' resources [11]. For example, assume task *t* needs the resource set $\{r_1, r_2, r_3\}$; now let node a_1 hold the resource set $\{r_1, r_2\}$, and a_2 hold the resource set $\{r_1\}$. According to the perspective of self-owned resource-based optimization, a_1 may have higher performance for providing necessary resources to execute *t*. However, now it is assumed that a_2 can easily share resources $\{r_2, r_3\}$ with its interacting counterparts, but a_1 cannot share any resources from other nodes, thus we should allocate task *t* to a_2 but not a_1.

Therefore, a new task allocation idea based on contextual resource optimization is presented [11]: if a node does not own plentiful resources by itself, but it can obtain enough resources from other nodes in context easily, it may also be allocated tasks; the number of allocated tasks on a node is directly proportional to not only its own resources but also the resources of its interacting nodes.

Let there be a node a_i, and the set of nodes within its context is C_i. Obviously, every node within C_i will contribute differently to the resource performance of a_i; the contribution of a node within C_i to node a_i is determined by the distance between such node and a_i in the network. Now, the concept of the contextual resource enrichment factor of node a_i for resource r_k can be defined as follows:

$$\Phi_i(k) = \sum_{a_j \in C_i} \left(n_j(k) \bullet \frac{1/d_{ij}}{\sum_{a_j \in C_i} 1/d_{ij}} \right), \tag{3}$$

where $n_j(k)$ is the amount of resource r_k owned by node a_j, and d_{ij} is the distance between a_i and a_j in the network; $a_i \in C_i$. Then, based on the above metrics for contextual resources of nodes, Jiang and Jiang [11] presented the objective of contextual resource optimization for task allocation: if a task needs resource r_k, let the set of nodes be A, then the task will be allocated to the following node:

$$a_* = \arg\max_{\forall a_i \in A} \Phi_i(k). \tag{4}$$

Moreover, Jiang and Huang [1] considered the contextual resources in the systems with resource caching: if a node's contextual nodes have richer experiences of executing tasks, the node may have higher access to the resources (required by the tasks) even if it seldom accessed the resources by itself. Therefore, Jiang and Huang [1] proposed the following novel idea of task allocation based on contextual preferential attachment: contextually-experienced nodes receive more new tasks than less contextually-experienced nodes. With such model, the contextual resource access time can be optimized.

Moreover, in real-world situations, multiplex networks are often observed where there are multiple types of links between nodes, and each type of link may have a different relative bias in communicating different types of resources. Therefore, in the systems, there may be several network layers, where each network layer is composed of the same type of links and the involved nodes. By considering this new situation, Jiang et al. [63] proposed the network layer-oriented task allocation model, which considers not only the contextual nodes but also the contextual layers.

In summary, the contextual resource-based optimization methods consider the coordination among nodes, so they can be fitted for current cooperative distributed systems where many nodes often cooperate for executing tasks. However, these optimization methods need a large amount of computation, which may produce high costs for the large-scale systems.

4.3 Summary and Future Research Directions

In summary, the comparisons among the two resource optimization methods can be shown as Table 2. From the table, we can see that each method has its own advantages and disadvantages.

Next, we discuss some challenges in existing studies and present some insights on future research directions on the resource optimization of task allocation and load balancing in distributed systems, which are shown as follows.

- There are often many tasks that are executed concurrently in systems, and the concurrent tasks may compete for resources. Existing related studies mainly perform resource optimization according to the current resource situation in the system and cannot predict the future situation where other tasks may also consume resources. Therefore, how to make resource optimization by considering and predicting the resource usages of other concurrent tasks is a research challenge.
- The resource distribution in systems may be dynamic; more seriously, the resource optimization schemes may be disabled due to the drastic change of resource distribution. Therefore, in the future, the resource optimization schemes should adapt to the dynamic change of resource distribution.
- In some distributed systems, such as cloud computing systems, virtualization technology is often used to provide virtualized resources for higher-level applications. Virtualization technology may create challenges for resource optimization because the virtualization provides the capability of pooling computing resources from clusters of servers and dynamically assigning or reassigning virtual resources to applications on-demand [88]. Such issue should be investigated in the future research of resource optimization.

5 Reliability in Task Allocation and Load Balancing

Due to the openness, dynamics, and heterogeneity of distributed systems, the nodes or networks may be unreliable; even more, some users may use the systems to perform malicious actions. Reliability in task allocation and load balancing is of critical importance for distributed systems, especially for some mission critical applications, such as financial systems and military ad hoc networks.

The reliability of tasks is the probability that all tasks can be executed successfully in the system. The reliability-oriented task allocation problem is to find a task allocation result such that the reliability is maximized [30], [31]. Typically, there are two classes of approaches: one is the redundancy-based approaches, which mainly implement redundancy of some elements so that the overall task execution is successful even if some parts of the elements fail; the other is the non-redundancy based approaches, which mainly adopt certain mechanisms to optimize the reliability for tasks.

5.1 Redundancy-Based Approaches

There are two types of redundancy to achieve the reliability for executing tasks: one is the redundancy of resources, and the other is the redundancy of tasks.

5.1.1 Redundancy of Resources

Redundancy of resources means that more resources will be provided for tasks than necessary, which can avoid the situation where the tasks may be executed unsuccessfully due to the unreliability of some resources. Generally, it is demonstrated that redundancy in resources can improve reliability. However, the reliability-oriented task allocation based on resource redundancy is NP-hard, so it is necessary to find some heuristic algorithms that can achieve near-optimal results efficiently and simply [31].

Moreover, redundancy may be an expensive approach and incurs higher related costs, thus some researchers aim to maximize the reliability of task allocation with less extra hardware and/or software costs. For example, Kang et al. [13] proposed a simple and effective iterative greedy algorithm to find the best possible solution within a reasonable amount of computation time; the algorithm first uses a constructive heuristic to obtain an initial allocation and iteratively improves it in a greedy way. Hsieh [64] proposed a hybrid genetic algorithm to determine the optimal resource redundancy policies so that the system cost is minimized. Elegbede et al. [65] proposed an algorithm, ECAY, which allows the allocation of both reliability and redundancy to each subsystem for target reliability for minimizing the system cost.

5.1.2 Redundancy of Tasks

Reliability can be achieved by the fault-tolerance of tasks, which can be implemented by the introduction of redundant copies of tasks [66]: a task consists of several copies that are resident on and executed by a number of separate nodes; the final result can be achieved by a majority voting rule so that the mistake of any single copy of a task can be suppressed. For example, Cherkassky and Chen [67] described a simple yet effective method to improve the reliability via redundant allocation of tasks to computers, where tasks are allocated to computers redundantly using the k-circular shifting algorithm so that, if some computers fail during the execution, all tasks can be completed on the remaining computers. Dai and Levitin [68] presented a method to maximize the reliability of grid service tasks, where the systems can divide tasks into execution blocks and assign the same execution blocks to several independent resources for redundant execution. Tom and Murthy [69] presented a method to allocate the subtasks to nodes in a manner that maximizes the probability of being able to successfully access all the files required for execution.

However, the reliability model based on task copies and majority voting will produce additional computation time and costs. Therefore, many studies aimed to reduce the computational costs caused by the redundant copies of tasks; some representative studies are shown as follows.

Jiang et al. [70] considered the reliability of mobile tasks in distributed systems with malicious nodes and presented a novel task migration fault-tolerance model based on integrity verification, which can reduce the complexity and redundancy costs by comparing with the traditional fault-tolerance approaches based on task copies and majority voting. In [70], the mobile task does not need to produce replica at every migration step; the replica of the mobile task need to be produced only when the task integrity is tampered with by a malicious node. Let the number of task migration steps be n, and the number of candidate nodes in every migration step be *m*; the complexity of the task migration communication degrees in [70] is $O(n^*m)$ by comparing with the complexity of $O(n^*m)^2$ in previous benchmark studies.

Karit and Murthy [30] used the idea of branch and bound with underestimates for reducing the computation and presented a heuristic algorithm that obtains suboptimal task allocations for maximizing reliability in a reasonable amount of computational time, in which the list of modules of a task are reordered to allow a subset of modules that do not communicate with one another to be assigned last for further reduction in the computation.

5.2 Non-Redundancy-Based Approaches

The redundancy-based approaches introduced in Section 5.1 may impose extra hardware or software costs to the systems; such costs may significantly influence the performance of the system while the number of tasks is large; moreover, in many situations, the redundancy is not practical. Therefore, many studies have been conducted that aim to present a proper allocation mechanism to improve system reliability without redundancy so that the additional hardware or software costs will not be required.

5.2.1 Optimization Mechanisms for Reliability

To maximize reliability for tasks, optimization mechanisms are used to find the optimal or near-optimal task allocation strategy.

To measure the unreliability in the task execution, various cost functions were presented in related studies; based on the defined cost functions, some schemes are presented to maximize the reliability. For example, Faragardi et al. [71] presented an optimization method for reliable task allocation that consists of a cost function representing the unreliability caused by the execution time of tasks on nodes and inter-processor communication time; the aim of the method is to find a task allocation under which the overall reliability of the systems is maximized. Moreover, Attiya and Hamam [10] developed an allocation model for reliability also based on such a cost function, and presented a heuristic algorithm derived from the well-known simulated annealing technique to quickly solve the reliability problem.

There are also some studies that introduced other related techniques into ensuring reliability for tasks. For example, Yin et al. [72] presented a hybrid particle swarm optimization (HPSO) algorithm for finding the near-optimal task allocation within a reasonable time for maximizing reliability, which is robust against different problem sizes, task interaction densities, and network topologies. Kang et al. [73] proposed a new swarm intelligence technique based on the honeybee mating optimization algorithm for the task allocation, which combines the power of simulated annealing and genetic algorithms with a fast problem specific local search heuristic to find the best possible solution within a reasonable computation time for maximizing reliability. Moreover, Srinivasan and Jha [74] devised a new heuristic based on the concept of clustering to allocate tasks for maximizing reliability.

In summary, the existing studies mainly aim to use heuristics to achieve the theoretically reliable performance with minimum costs for some special cases. However, the generality and practicability of existing approaches may not be ideal.

5.2.2 Trust-or Reputation-Based Approaches

In distributed systems, some nodes may take subjective initiative for generating malicious or deviant behaviors [9]; for example, some nodes may fabricate their resource status information during the task allocation for achieving the tasks, but they do not really contribute all their free resources if the allocated tasks require those resources [27]. To measure the reliability of nodes for tasks, the concepts of trust and reputation are often used.

1) Trust-based approaches. Trust denotes that one party is willing to rely the actions on another part. Given the background that some nodes may not successfully complete their allocated tasks, the trust between nodes should be taken into account when allocating tasks. Trust in task allocation often denotes that the allocator is willing to rely on the actions of a node for providing reliable execution of such task.

Ramchurn et al. [75] developed a class of trust-based mechanisms that can take into account multiple subjective measures of the probability of a node succeeding at a given task and produce allocations that maximize social utility and ensure that no node obtains a negative utility. Shen et al. [76] developed a trust model to construct a novel mechanism that motivates sensor agents to limit their greediness or selfishness to make task allocation, and they modeled the sensor allocation optimization problem with trust-in-loop negotiation game and solved it using a subgame perfect equilibrium.

2) Reputation-based approaches. Reputation of a social actor means the opinion about such actor as defined by others [87]. Now, the concept of reputation can be introduced to measure the reliability in distributed systems. A node's reliability is not merely a binary property perceived by another node, but instead is often a statistical one based on such node's prior performance and behavior as perceived by other nodes [78].

To achieve reliable resources with the least access time to execute tasks in undependable social network systems, Jiang et al. [27] presented a task allocation model based on the negotiation reputation mechanism, where a node's past behaviors in the resource negotiation of task execution can influence its probability to be allocated new tasks in the future. In this model, the node that contributes more reliable resources with less access time during task execution is rewarded with a higher negotiation reputation, and may receive preferential allocation of new tasks. The task allocation model based on negotiation reputation in [27] is superior to the traditional resource-based allocation approaches and game theory-based allocation approaches in terms of both the task allocation success rate and task execution time and that it usually performs close to the ideal approach (in which deceptive nodes are fully detected) in terms of task execution time. In addition, the model in [27] considered the reputation of communication paths; the reputation of communication paths between two nodes can be adapted according to the two nodes' past negotiations in task execution; for example, if the two nodes can negotiate for

TABLE 3
The Comparison among the Approaches for Reliability

Approaches		Basic principle	Advantages	Disadvantages	Application Environments
Redundancy-based		Implement redundancy of some elements so that the overall task execution is successful even if some parts of the elements fail	Good failure-tolerance performance	Produce additional computation and resource costs	Small-scale systems
Non-redundancy-based	Optimization mechanisms	Use optimization mechanisms to find the optimal or near-optimal task allocation strategy	Can achieve a theoretically reliable performance with minimum costs	The optimization mechanism is not general and may be only useful for some special cases	Some special cases under certain constraints
	Trust/reputation mechanisms	Trust or reputation is used to ensure the reliability: the nodes should interact to construct trust relations for task allocation; and a node's reputation is determined by its past experiences for executing tasks.	There are many mature trust/reputation mechanisms that can be used. And this approach can adapt for different system scales	The construction of trust relation between nodes may need costs or infrastructure. The past experiences of a node may be undependable so that incorrect reputation may be achieved.	Relatively general environments

finishing a task through the path, then the reputation of that path will be gained, and vice versa.

Another representative study is that Yahyaoui [79] presented a distributed reputation-based game theoretical model for task allocation: each node submits a cost for achieving a specific task and computes the reputation-based cost, and the node with the minimal reputation-based cost will be allocated the task.

5.3 Summary and Future Research Directions

In summary, a comparison of the different approaches for ensuring reliability of task is shown as Table 3. Next, we discuss some challenges in existing studies and present some insights on future research directions on the reliability of task allocation and load balancing in distributed systems, which are shown as follows.

- In a distributed system, there may be more than one mechanism for ensuring reliability. For example, hybrid clouds are composed of private and public clouds, and the private cloud may have a reliability mechanism different from that of the public cloud. Therefore, how to coordinate different reliability mechanisms and ensure their consistency is a future research issue.
- In most existing related studies, the reliability mechanisms are often predesigned offline. However, the systems may be large and dynamic; thus, how to make the system learn to adopt a suitable reliability mechanism online should be solved in the future.
- Existing reliability mechanisms mainly aim to ensure the reliability during the stage of task allocation. However, how can we do if the perfect reliability cannot be ensured during the task allocation stage? We think that, in the future, the reliability may also be achieved during the task execution stage, i.e., a reliable task execution mechanism should be investigated.

6 Coordination among Heterogeneous Nodes in Task Allocation and Load Balancing

In many distributed systems, the nodes are heterogeneous to provide varying distributed applications. For example, in a grid, the nodes may be attributed to different organizations. Heterogeneous nodes may not cooperate in making decisions [32], and, each node may optimize its own resource access performance independently of the others when the node executes tasks. Therefore, there are some related studies aiming to make these noncooperative nodes coordinate to improve the overall performance of tasks, in which some mechanisms are used to incentivize self-interested nodes to contribute their resources to others for executing tasks [28].

6.1 Game Theory-Based Approaches

In existing studies, game theory is often used to make heterogeneous nodes coordinate to reach an equilibrium. For example, Subrata et al. [80] modeled the grid load-balancing problem as a noncooperative game with the objective of reaching the Nash equilibrium, where the nodes are heterogeneous in different processing power. Grosu and Chronopoulos [32] considered a distributed system that consists of n heterogeneous nodes shared by m users and formulated the load balancing problem as a noncooperative game among users under the assumption that the users would attempt to minimize the expected response time of their own jobs.

Even-Dar et al. [81] investigated the load balancing in the Internet and related it to potential games, and studied the number of steps required to reach a pure Nash equilibrium in a load balancing scenario where each user behaves selfishly and attempts to migrate to a machine which will minimize its cost; moreover, they found that load balancing in unrelated nodes is a generalized ordinal potential game, load balancing in related nodes is a weighted potential

TABLE 4
The Comparison between Game Theory-Based and Graph Theory-Based Approaches for the Coordination of Nodes

Coordination mechanisms	Implementation manner	Performance	System scales
Game theory-based	Distributed manner	Adapt for the dynamic situations and have good robustness	Large scale
Graph theory-based	Centralized manner	Globally optimal	Small scale

game, and load balancing in related nodes and unit weight tasks is an exact potential game.

6.2 Other Approaches

In addition to game theory, there are some other approaches to achieve coordination among heterogeneous nodes for task allocation; typically, graph theory tools are often used for modeling and analyzing the coordination relations among heterogeneous nodes [89].

For example, a task interaction graph, which is an undirected graph where two tasks communicate if there is an edge between the nodes, can be used to model the coordination among heterogeneous nodes [82]. Based on the task interaction graph, Ucar et al. [83] considered the heterogeneous nodes where the execution cost of a task depends on the node on which it is executed, and they formally defined the task allocation as the optimization problem for the sum of the execution and communication related to the task interaction graph and the task-node expected computation time matrix.

Hong and Prasanna [55] modeled the interconnection between heterogeneous nodes as a graph and used an extended network flow representation to make nodes coordinate in the task allocation to maximize the throughput of the system. Bajaj and Agrawal [84] investigated the allocation of parallel tasks to the heterogeneous system, where the nodes may not be identical and take different amounts of time to execute the same task, and introduced directed acyclic graphs (DAGs) to represent the applications and provide a task duplication-based scheduling algorithm for a network of heterogeneous systems.

6.3 Summary and Future Research Directions

In summary, the coordination mechanisms among heterogeneous nodes are similar to the general coordination mechanisms in society, among which the game theory and graph theory are often used. The comparison between the two types of approaches can be shown as Table 4.

Next, we discuss some challenges in existing studies and present some insights on future research directions on the coordination among nodes in distributed systems, which are shown as follows.

- In current related studies, the roles of nodes are always assumed to remain fixed during the task allocation and execution. However, such an assumption cannot fully reflect reality. In fact, nodes can dynamically change their roles in the system, e.g., an untruthful node can perform truthful action when it finds that such an action can derive more benefits; or a competitive node can also perform cooperative actions in the system. Therefore, the coordination among dynamically transformable nodes in task allocation is a research challenge.
- In existing studies, each node often makes decision individually according to its own strategy. However, some nodes may form a cluster or a community, and the nodes within a cluster or a community may perform some related strategies for allocating tasks. In the future, we plan to investigate the coalition mechanism and collective decision-making model of nodes organized by clusters or communities in distributed systems.

7 Considering Network Structures in Task Allocation and Load Balancing

In distributed systems, network structures are very important and can influence the coordination among nodes for executing tasks. Generally, there are two types of network structures: one is that the nodes are interconnected by physical communication network structures; the other is that the nodes are interconnected by virtual interaction network structures, such as social network systems [96].

The first type of network structures are the physical communication networks, which may influence the communication performances among nodes; thus the resource access of nodes for executing tasks is significantly related to this type of network structures. For example, if a node locates at an outlying place in the network structure, it may needs much communication costs if such node wants to cooperate with other nodes. Therefore, both the resources and locality of a node should be taken into account in task allocation; the experiment results in [15] showed that the locality-based task method can reduce the communication costs for a single task more effectively than the resource-based one, but the resource-based method can generally obtain better load balancing performance for parallel tasks than the locality-based one.

The second type of network structures are the virtual interaction networks, which are composed of the interaction relations among nodes; an example of this type is social network systems. Jiang et al. [27] made a series of experiments on the task allocation in some typical social network structures: small world networks, scale-free networks, random networks, regular ring networks, and the Jazz bands networks; they validated the influences of different social networks on the task allocation model.

In summary, some challenges in existing studies and future research directions on the effects of network structures in task allocation and load balancing of distributed systems can been shown as follows.

- In reality, the physical communication networks and virtual interaction networks may be correlated with each other. Therefore, the correlated effects between these two types of network structures in task allocation should be investigated in the future.

- In real distributed systems, especially large systems, there may be hybrid network structures; for example, ad hoc structure and P2P structure may coexist in the physical communication network in a system, or the small world structures and random structures may be hybrid in the virtual interaction network in a system. Therefore, the effects of hybrid network structures on task allocation will be investigated in the future.

8 Conclusions

Task allocation and load balancing have been intensively researched in past decades; a large number of related studies and results have been presented concerning this topic. However, there are many types of distributed systems, which means that the task allocation and load balancing models in different types of systems are also different; therefore, people may often be puzzled by the enormous amount of varying related studies.

To solve the above problem, in this survey, we perform a systematic review of task allocation and load balancing by analyzing the general characteristics of distributed systems. This survey summarizes the following general characteristics of distributed systems: 1) lack of a global control unit; 2) distribution and sharing of resources; 3) openness and unreliability of systems; 4) heterogeneous nodes; and 5) constraints of network structures. Based on these typical characteristics, this paper reviews the existing studies on task allocation and load balancing with respect to the following aspects: control models, resource optimization, reliability, coordination, and considering network structures. For each aspect, we summarize the existing studies and discuss the future research directions. Through this survey, the related studies on task allocation and load balancing can be understood well by considering how they can satisfy the characteristics of distributed systems.

Currently, distributed systems are developing rapidly, and new distributed computing topics are continually presented. For example, big data and cloud computing may create many new problems, such as the need for high throughput, interoperability between hybrid clouds, security and privacy. Therefore, task allocation and load balancing should consider these new emerging problems in distributed systems. Moreover, current systems may often be dynamic, scalable, and evolutionary; and how to respond quickly to rapid changes in those systems may bring great challenges to task allocation and load balancing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61170164, and No. 61472079), the Funds for Distinguished Young Scholars of the Natural Science Foundation of Jiangsu Province (No. BK2012020), and the Program for Distinguished Talents of Six Domains in Jiangsu Province (No. 2011-DZ023). Yichuan Jiang is the corresponding author.

References

[1] Y. Jiang and Z. Huang, "The rich get richer: Preferential attachment in the task allocation of cooperative networked multiagent systems with resource caching," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 5, pp. 1040–1052, Sep. 2012.

[2] Y. Jiang, J. Hu, and D. Lin, "Decision making of networked multiagent systems for interaction structures," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 6, pp. 1107–1121, Nov. 2011.

[3] D. Ye, M. Zhang, and D. Sutanto, "Self-adaptation based dynamic coalition formation in a distributed agent network: A mechanism and a brief survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 1042–1051, May 2013.

[4] J. Liu, X. Jin, and Y. Wang, "Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 7, pp. 586–598, Jul. 2005.

[5] W. Rao, L. Chen, A. Wai-Chee Fu, and G. Wang, "Optimal resource placement in structured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 1011–1026, Jul. 2010.

[6] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: A price-based approach," *IEEE Trans. Mobile Comput.*, vol. 5, no. 4, pp. 347–364, Apr. 2006

[7] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 24–32, Jun. 2013.

[8] P. Lukowicz, A. Pentland, and A. Ferscha, "From context awareness to socially aware computing," *IEEE Pervasive Comput.*, vol. 11, no. 1, pp. 32–41, Jan.–Mar. 2012.

[9] Y. Jiang and J. C. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2743–2759, Oct. 2014.

[10] G. Attiya and Y. Hamam, "Task allocation for maximizing reliability of distributed systems: A simulated annealing approach," *J. Parallel Distrib. Comput.*, vol. 66, no. 10, pp. 1259–1266, 2006.

[11] Y. Jiang and J. Jiang, "Contextual resource negotiation-based task allocation and load balancing in complex software systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 641–653, May 2009.

[12] Y. C.Jiang, J. Jiang, "A multi-agent coordination model for the variation of underlying network topology," *Expert Syst. Appl.*, vol. 29, no. 2, pp. 372–382, 2005.

[13] Q. Kang, H. He, and J. We, "An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 73, no. 8, pp. 1106–1115, 2013.

[14] B. Hong and V. K. Prasanna, "Adaptive allocation of independent tasks to maximize throughput," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 10, pp. 1420–1435, Oct. 2007.

[15] Y. Jiang and Z. Li, "Locality-sensitive task allocation and load balancing in networked multiagent systems: Talent versus centrality," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 822–836, 2011.

[16] W. Jia and W. Zhou, *Distributed Network Systems: From Concepts to Implementations*. Boston, MA, USA: Springer, 2005.

[17] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*, 2nd ed. Reading, MA, USA: Addison-Wesley 1994.

[18] S. J. Kim and G. B.Giannakis, "Optimal resource allocation for MIMO ad hoc cognitive radio networks," *IEEE Trans. Inf. Theory*, vol. 57, no. 5, pp. 3117–3131, May 2011.

[19] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[20] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Softw.: Prac. Exp.*, vol. 32, no. 2, pp. 135–164, 2001.

[21] D. Lin, T. Ishida, Y. Murakami, and M. Tanaka, "QoS analysis for service composition by human and web services," *IEICE Trans. Inf. Syst.*, vol. E97-D, no. 4, pp. 762–769, 2014.

[22] B. Tan and L. Massoulie, "Optimal content placement for peer-to-peer video-on-demand systems," in *Proc. 30th IEEE Int. Conf. Comput. Commun.*, Shanghai, China, Apr. 2011, pp. 694–702.

[23] L.Yin and G.Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 1, pp. 77–89, Jan. 2006.

[24] P. Yi, Y. Wu, F. Zou, and N. Liu, "A survey on security in wireless mesh networks," *IETE Tech. Rev.*, vol. 27, no. 1, pp. 6–14, 2010.

[25] T. Zhu, A. Mohaisen, P. Yi, and T. Don, "DEOS: Dynamic energy-oriented scheduling for sustainable wireless sensor networks," in *Proc. 31st Annu. IEEE Int. Conf. Comput. Commun.*, Orlando, FL, USA, Mar. 2012, pp. 2363–2371.

[26] Y. Jiang and J. C. Jiang, "Diffusion in social networks: A multiagent perspective," *IEEE Trans. Syst., Man, and Cybern. Syst.*, vol. 45, no. 2, pp. 198–213, Feb. 2015.

[27] Y. Jiang, Y. Zhou, and W. Wang, "Task allocation for undependable multiagent systems in social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 8, pp. 1671–1681, Aug. 2013.

[28] M. M. Weerdt, Y. Zhang, and T. Klos, "Multiagent task allocation in social networks," *Auton. Agents Multi-Agent Syst.*, vol. 25, no. 1, pp. 46–86, 2012.

[29] B. An, V. Lesser, and K. M. Sim, "Strategic agents for multiresource negotiation," *J. Auton. Agents Multi-Agent Syst.*, vol. 23, no. 1, pp. 114–153, 2011.

[30] S. Kartik and C. S. R. Murthy, "Task allocation algorithms for maximizing reliability of distributed computing systems," *IEEE Trans. Comput.*, vol. 46, no. 6, pp. 719–724, Jun. 1997.

[31] C. C. Chiu, Y. S. Yeh, and J. S. Chou, "A fast algorithm for reliability-oriented task assignment in a distributed system," *Comput. Commun.*, vol. 25, no. 17, pp. 1622–1630, 2002.

[32] D. Grosu, and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *J. Parallel Distrib. Comput.*, vol. 65, no. 9, pp. 1022–1034, 2005.

[33] T. Kunz, "The influence of different workload descriptions on a heuristic load balancing scheme," *IEEE Trans. Softw. Eng.*, vol. 17, no. 7, pp. 725–730, Jul. 1991.

[34] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1624–1631, Oct. 2011.

[35] S. Di and C.-L. Wang, "Decentralized proactive resource allocation for maximizing throughput of P2P grid," *J. Parallel Distrib. Comput.*, vol. 72, no. 2, pp. 308–321, 2012.

[36] Z. Tong, Z. Xiao, K. Li, and K. Li," Proactive scheduling in distributed computing—A reinforcement learning approach," *J. Parallel Distrib. Comput.*, vol. 74, no. 7, pp. 2662–2672, Jul. 2014.

[37] A. Y. Zomaya, and Y. H. Teh, "Observations on using genetic algorithms for dynamic load-balancing,"*IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 9, pp. 899–911, Sep. 2001.

[38] T. Eymann, M. Reinicke, O. Ardaiz, P. Artigas, L. Díaz de Cerio, F. Freitag, R. Messeguer, L. Navarro, and D. Royo, "Decentralized vs. centralized economic coordination of resource allocation in grids," in *Proc. 1st Eur. Across Grid Conf.*, 2004 pp. 9–16.

[39] J. van der Horst, and J. Noble, "Distributed and centralized task allocation: When and where to use them," in *Proc. 4th IEEE Int. Conf. Self-Adaptive Self-Org. Syst. Workshop*, Budapest, Hungary, Sept. 2010, pp. 1–8.

[40] H.-C. Lin and C. S. Raghavendra, "A dynamic load-balancing policy with a central job dispatcher (LBC)," *IEEE Trans. Softw. Eng.*, vol. 18, no. 2, pp. 148–158, Feb. 1992.

[41] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in dynamic structured P2P systems," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Hong Kong, China, Mar. 2004, pp. 2253–2262.

[42] W. E. Walsh and M. P. Wellman, "A market protocol for decentralized task allocation," in *Proc. 3rd Int. Conf. Multi-Agent Syst.*, Paris, France, Jul. 1998, pp.325–332.

[43] P. Ghosh, N. Roy, S. K. Das, and K. Basu, "A game theory based pricing strategy for job allocation in mobile grids," presented at the 18th Int. Parallel Distributed Process. Symp., Santa Fe, NM, USA, April 26–30, 2004.

[44] S. Penmatsa, and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 4, pp. 537–555, 2011.

[45] H. Izakian, A. Abraham, and B. T. Ladani, "An auction method for resource allocation in computational grids," *Future Generation Comput. Syst.*, vol. 26, no. 2, pp. 228–235, 2010.

[46] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[47] S. K. Garg, S. Venugopal, J. Broberg, and R. Buyya, "Double auction-inspired meta-scheduling of parallel applications on global grids," *J. Parallel Distrib. Comput.*, vol. 73, no. 4, pp. 450–464, 2013.

[48] T. Rotaru and H. H. Nägeli, "Dynamic load balancing by diffusion in heterogeneous systems," *J. Parallel Distrib. Comput.*, vol. 64, no. 4, pp. 481–497, 2004.

[49] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, no. 1/2, pp. 165–200, 1998.

[50] Y. Jiang, "Concurrent collective strategy diffusion of multiagents: The spatial model and case study," *IEEE Trans. Syst., Man Cybern. C*, vol. 39, no. 4, pp. 448 -458, Jul. 2009.

[51] K. H. Low, W. K .Leow, and M. H. Ang Jr., "Task allocation via self-organizing swarm coalition in distributed mobile sensor network," in *Proc. 19th Nat. Conf. Artif. Intell.*, San Jose, CA, USA, Jul. 2004, pp. 28–33.

[52] W. Wang and Y. Jiang, "Community-aware task allocation for social networked multiagent systems," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1529–1543, Sep. 2014.

[53] Ö. Babaoglu, H. Meling, and A. Montresor, "Anthill: A framework for the development of agent-based peer-to-peer systems," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst.*, Vienna, Austria, Jul. 2002, pp. 15–22.

[54] K.-P. Chow and Y.-K. Kwok, "On load balancing for distributed multiagent computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 8, pp. 787–801, Aug. 2002.

[55] B. Hong and V. K. Prasanna, "Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput," presented at the 18th Int. Parallel Distributed Processing Symp., Santa Fe, NM, USA, Apr. 2004.

[56] O. Beaumont et al., "Centralized versus distributed schedulers for bag-of-tasks applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 698–709, May 2008.

[57] K. Antonis, J. Garofalakis, I. Mourtos, and P. Spirakis, "A hierarchical adaptive distributed algorithm for load balancing," *J. Parallel Distrib. Comput.*, vol. 64, no. 1, pp. 151–162, 2004.

[58] R. Shah, B. Veeravalli, and M. Misra, "On the design of adaptive and decentralized load-balancing algorithms with load estimation for computational grid environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 12, pp. 1675–1686, Dec. 2007.

[59] M. J. Zaki, W. Li, and S. Parthasarathy, "Customized dynamic load balancing for a network of workstations," in *Proc. 5th IEEE Int. Symp. High Perform. Distrib. Comput.*, Syracuse, NY, USA, Aug. 1996, pp. 282–291.

[60] S. Abdallah and V. Lesser, "Modeling task allocation using a decision theoretic model," in *Proc. 4th Int. Joint Conf. Auton. Agents Multiagent Syst.*, Utrecht, Netherlands, Jul. 2005, pp. 719–726.

[61] A. Das and D. Grosu, "Combinatorial auction-based protocols for resource allocation in grids," presented ta the 19th IEEE Int. Parallel Distributed Processing Symp., Denver, CO, USA, Apr. 2005.

[62] P. Sugavanam, H. J. Siegel, A. A. Maciejewski, M. Oltikar, A. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. Krishnamurthy, S. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, A. Pippin, "Robust static allocation of resources for independent tasks under makespan and dollar cost constraints," *J. Parallel Distrib. Comput.*, vol. 67, no. 4, pp. 400–416, 2007.

[63] Y. Jiang, Y. Zhou, and Y. Li, "Reliable task allocation with load balancing in multiplex networks," *ACM Trans. Auton. Adaptive Syst.*, vol. 10, no. 1, article 3, Feb. 2015.

[64] C.-C. Hsieh, "Optimal task allocation and hardware redundancy policies in distributed computing systems," *Eur. J. Oper. Res.*, vol. 147, no. 2, pp. 430–447, 2003.

[65] A. O. Charles Elegbede, C. Chu, K. H. Adjallah, and F. Yalaoui, "Reliability allocation through cost minimization," *IEEE Trans. Rel.*, vol. 52, no. 1, pp. 106–111, Mar. 2003.

[66] J. A. Bannister and K. S. Trivedi, "Task allocation in fault-tolerant distributed systems,"*Acta Informatica*, vol. 20, no. 3, pp. 261–281, 1983.

[67] V. Cherkassky and C. I. H. Chen, "Redundant task-allocation in multicomputer systems," *IEEE Trans. Rel.*, vol. 41, no. 3, pp. 336–342, Sep. 1992.

[68] Y. S. Dai and G. Levitin, "Optimal resource allocation for maximizing performance and reliability in tree-structured grid services," *IEEE Trans. Reliability*, vol. 56, no. 3, pp. 444–453, Sep. 2007.

[69] P. Ajith Tom and C. Siva Ram Murthy, "Algorithms for reliability-oriented module allocation in distributed computing systems," *J. Syst. Softw.*, vol. 40, no. 2, pp. 125–138, 1998.

[70] Y. C. Jiang, Z. Y. Xia, Y. P. Zhong, and S. Y. Zhang, "Defend mobile agent against malicious hosts in migration itineraries," *Microprocessors Microsyst.*, vol. 28, no. 10, pp. 531–546, 2004.

[71] H. R. Faragardi, R. Shojaee, and M. A. Keshtkar, and H. Tabani, "Optimal task allocation for maximizing reliability in distributed real-time systems," in *Proc. IEEE/ACIS 12th Int. Conf. Comput. Inf. Sci.*, Niigata, Japan, Jun. 2013, pp. 513–519.

[72] P. Y. Yin, S. S. Yu, P. P. Wang, and Y. T. Wang, "Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization," *J. Syst. Softw.*, vol. 80, no. 5, pp. 724–735, 2007.
[73] Q. M. Kang, H. He, H. M. Song, and R. Deng, "Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization," *J. Syst. Softw.*, vol. 83, no. 11, pp. 2165–2174, 2010.
[74] S. Srinivasan and N. K. Jha, "Safety and reliability driven task allocation in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 3, pp. 238–251, Mar. 1999.
[75] S. D. Ramchurn, C. Mezzetti, A. Giovannucii, J. A. Rodriguez-Aguilar, R. K. Dash, and N. R. Jennings, "Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty," *J. Artif. Intell. Res.*, vol. 35, no. 1, pp. 119–159, 2009.
[76] D. Shen, G. Chen, K. Pham, and E. Blasch, "A trust-based sensor allocation algorithm in cooperative space search problems," *Proc. SPIE 8044, Defense, Secur., Sensing. Int. Soc. Optics Photonics*, vol. 8044, 2011, pp. 80440C-1–80440C-9.
[77] A. A. Almutairi, M. I. Sarfraz, and S. Basalamah, "A distributed access control architecture for cloud computing," *IEEE Softw.*, vol. 29, no. 2, pp. 36–44, Mar./Apr. 2012.
[78] J. Sonnek, A. Chandra, and J. B. Weissman, "Adaptive reputation-based scheduling on unreliable distributed infrastructures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 11, pp. 1551–1564, Nov. 2007.
[79] H. Yahyaoui, "A reputation-based game for tasks allocation," in *Proc. 11th Int. Conf. Enterprise Inf. Syst., Lecture Notes. Bus. Inf. Process.*, 2009, vol. 24, pp. 728–736.
[80] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Game-theoretic approach for load balancing in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 1, pp. 66–76, Jan. 2008.
[81] E. Even-Dar, A. Kesselman, and Y. Mansour, "Convergence time to Nash equilibrium in load balancing," *ACM Trans. Algorithm*, vol. 3, no. 3, article 32, 2007.
[82] C. C. Hui and S. T. Chanson, "Allocating task interaction graphs to processors in heterogeneous networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 9, pp. 908–925, Sep. 1997.
[83] B. Ucar, C. Aykanat, K. Kaya, and M. Ikinci, "Task assignment in heterogeneous computing systems," *J. Parallel Distrib. Comput.*, vol. 66, no. 1, pp. 32–46, 2006.
[84] R. Bajaj and D. P. Agrawal, "Improving scheduling of tasks in a heterogeneous environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 2, pp. 107–118, Feb. 2004.
[85] P. R. Ma, E. Y. S. Lee, and M. Tsuchiya, "A task allocation model for distributed computing systems," *IEEE Trans. Comput.*, vol. C-31, no. 1, pp. 41–47, Jan. 1982.
[86] S. M. Shatz, J. P. Wang, and M. Goto, "Task allocation for maximizing reliability of distributed computer systems," *IEEE Trans. Comput.*, vol. 41, no. 9, pp. 1156–1168, Sep. 1992.
[87] J. M. Pujol, R. Sangüesa, and J. Delgado, "Extracting reputation in multi agent systems by means of social network topology," in *Proc. 1st Int. Conf. Auton. Agents Multiagent Syst.*, Bologna, Italy, Jul. 2002, pp. 467–474.
[88] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.
[89] Y. Zhang, Z. Chang F. Y. L. Chin, H.-F. Ting, and Y. H. Tsin, "Online uniformly inserting points on grid," *Inf. Process. Lett.*, vol. 111, no. 16, pp. 773–779, 2011.
[90] S. Kraus, O. Shehory, G. Taase, "Coalition formation with uncertain heterogeneous information," in *Proc. 2nd Int. Conf. Auton. Agents Multiagent Syst.*, Melbourne, Australia, Jul. 2003, pp. 1–8.
[91] Y. Li and Z. Lan, "A survey of load balancing in grid computing," in *Proc. 1st Int. Conf. Comput. Inf. Sci.*, 2005, pp. 280–285.
[92] J.S. Raj and R. Fiona, "Load balancing techniques in grid environment: A survey," in *Proc. Int. Conf. Comput. Commun. Informat.*, Coimbatore, India, Jan. 2013, pp. 1–4.
[93] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: challenges and algorithms," in *Proc. 2nd Symp. Netw. Cloud Comput. Appl.*, London, England, Dec. 2012, pp. 137–142.
[94] T. Desai and J. Prajapati, "A survey of various load balancing techniques and challenges in cloud computing," *Int. J. Sci. Technol. Res.*, vol. 2, no. 11, pp. 158–161, 2013.
[95] V. Patange and D. D.Gatade, "Survey of load balancing approaches in peer-to-peer network," *Int. J. Soft Comput. Eng.*, vol. 3, no. 2, pp. 422–424, 2013.
[96] D.-B. Thuan, T. Nguyen, B. Bose, and D. A. Tran, "Distributed client-server assignment for online social networks applications," *IEEE Trans. Emerging Topics Comput.*, 2014, vol. 2, no. 4, pp. 422–435, Dec. 2014.

Yichuan Jiang (SM'13) received the PhD degree in computer science from Fudan University, Shanghai, China, in 2005. He is currently a full professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His main research interests include distributed systems, multi-agent systems, and social networks. He has published more than 80 scientific articles in refereed journals and conference proceedings, such as the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, the *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, the *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, the *IEEE Transactions on Cybernetics*, the *ACM Transactions on Autonomous and Adaptive Systems*, the *Journal of Parallel and Distributed Computing*, the International Joint Conference on Artificial Intelligence (IJCAI), the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), and the IEEE International Conference on Tools with Artificial Intelligence (ICTAI). He received the Best Paper Award from PRIMA06 and Best Student Paper Awards twice from ICTAI13 and ICTAI14. He is a senior member of the IEEE, a member of editorial board of Advances in Internet of Things, an editor of *International Journal of Networked Computing and Advanced Information Management*, an editor of *Operations Research and Fuzziology*, and a member of the editorial board of *Chinese Journal of Computers*.