

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,000

Open access books available

125,000

International authors and editors

140M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Analysis of Effective Load Balancing Techniques in Distributed Environment

Anju Shukla, Shishir Kumar and Harikesh Singh

Abstract

Computational approaches contribute the significance role in various fields such as medical applications, astronomy, and weather science, to perform complex calculations in speedy manner. Today, personal computers are very powerful but underutilized. Most of the computer resources are idle; 75% of the time and server are often unproductive. This brings the sense of distributed computing, in which the idea is to use the geographically distributed resources to meet the demand of high-performance computing. The Internet facilitates users to access heterogeneous services and run applications over a distributed environment. Due to openness and heterogeneous nature of distributed computing, the developer must deal with several issues like load balancing, interoperability, fault occurrence, resource selection, and task scheduling. Load balancing is the mechanism to distribute the load among resources optimally. The objective of this chapter is to discuss need and issues of load balancing that evolves the research scope. Various load balancing algorithms and scheduling methods are analyzed that are used for performance optimization of web resources. A systematic literature with their solutions and limitations has been presented. The chapter provides a concise narrative of the problems encountered and dimensions for future extension.

Keywords: load balancing, resource management, resource scheduling, load measurement, fault tolerance

1. Introduction

The performance of any web server has been affected by the web traffic usually, and the web server makes a slow response because it gets overloaded. Due to the increased traffic over the Internet, a web server faces challenges to serve the large number of users with high-speed availability. Therefore, the concept of resource confederation comes in existence. The popular Google web server works on the same concept. It distributes the user's query in different web servers which are geographically distributed at various locations. Load balancing plays a vital role in the operation of distributed and parallel computing. It partitioned the incoming workload into smaller tasks that are assigned to computational resources for concurrent execution. The load may be CPU capacity, memory size, network load, delay, etc. The reason behind load balancing is to handle requests of multiple users without degrading the performance of web server. Load balancer receives requests

from user, determines the load on available resources, and sends request to the server which is lightly loaded. The major functions of load balancer are as follows:

- Distributes incoming traffic across multiple computational resources
- Determines resource availability and reliability for task execution
- Improves resource utilization
- Increases client satisfaction
- Provides fault tolerance and flexible framework by adding or subtracting resources as demand occurs

Load balancing significantly improves global system performance in terms of throughput and resource utilization. The several reasons to use LBAs are as follows: cost optimization, fault tolerance ability, system adaptability and extensibility, decreased response time, idle time of resources, increased throughput, reliability, and starvation prevention [1, 2].

To incorporate these benefits, it is important to select the suitable Load Balancing Algorithm (LBA) for web resources. LBAs can be categorized in three categories based on process origination as shown in **Figure 1**. Both sender-initiated and receiver-initiated algorithms use different transfer and location policies for implementing load balancing. Symmetric-initiated algorithms eliminate the preemption condition of receiver-initiated algorithm and offer two algorithms: above-average algorithm and adaptive algorithm. Above-average algorithm uses an acceptable range for deciding whether a node is sender-initiated or receiver-initiated.

The second classification of LBA exists based on the current state of the system as shown in **Figure 2**. Static algorithms require prior information about the system characteristics such as processing capability, memory, number of active connections, etc., while DLB algorithms use the current status of the system to make scheduling decisions.

1.1 Static load balancing

Static load balancing approaches use the prior information of tasks, computing resources or processing element, and network detail as shown in **Figure 3**. The task can be submitted to any processing element using two methods:

- Stateless method
- State-based method

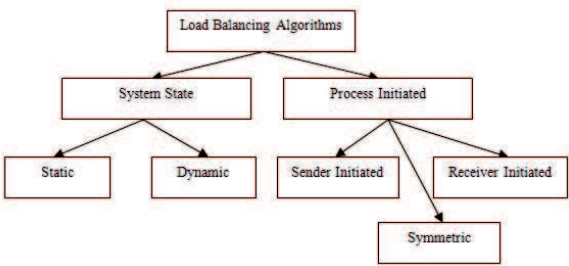


Figure 1.
Classification based on process origination.

In stateless method, selection of a processing element (PE) is done without having any awareness of the system environment, while in a state-based method, selecting a PE requires information of the system condition [3]. Stateless methods are simple to implement, but it provides one-to-one interaction between the client and server at a time as shown in **Figure 4**.

Figure 5 represents the stateful load balancing method; the load balancer keeps track for all the sessions, and decisions are taken based on server load. Various stateless techniques exist for selecting the processing element such as RR-LBA, weighted round robin (WRR)-LBA, and random allocation algorithm [3]. However, these algorithms have limited scope due to the dynamic nature of distributed environment.

1.2 Dynamic load balancing

It varies from the SLB algorithms in which clients' requests are distributed among available resources at run time. The LB assigns the request based on the dynamic information collected from all the resources as shown in **Figure 6**.

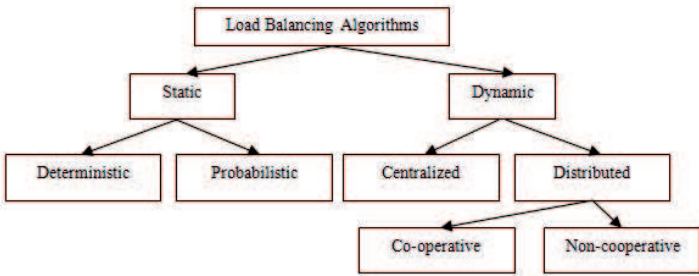


Figure 2.
Classification based on system state.

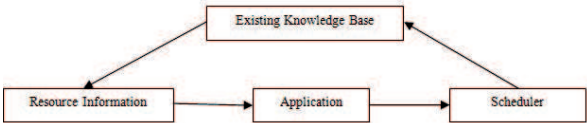


Figure 3.
Static load balancing.

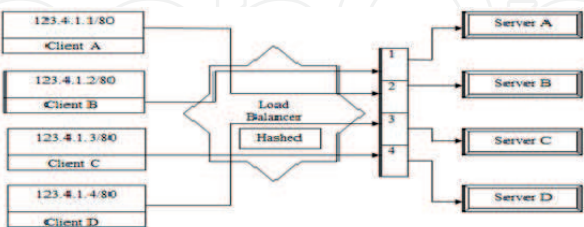


Figure 4.
Stateless static load balancing.

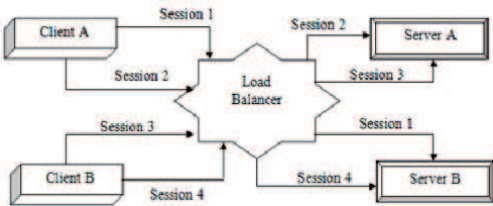


Figure 5.
Stateful static load balancing.

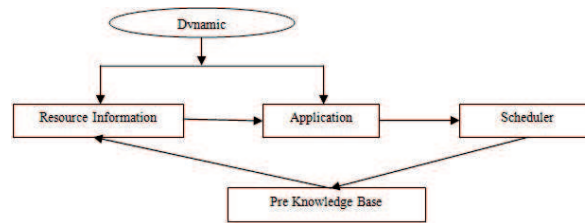


Figure 6.
Dynamic load balancing.

DLB algorithms can be classified in two categories—distributed and non-distributed. In distributed DLB, all computing resources are equally responsible for balancing the load. The responsibility of load balancing is shared among all the resources. But in non-distributed algorithms, each resource performs independently to accomplish the common goal. Generally, distributed DLB algorithms generated more message overhead than non-distributed DLB due to its interaction with all the resources.

Distributed algorithms perform better in fault conditions as it degrades only the sectional of the system instead of global system performance. Non-distributed algorithms are further classified into two categories—centralized and semi-centralized. In centralized algorithm, a central server is responsible for executing load balancing algorithm. In semi-centralized, servers are arranged in clusters, and load balancing within the cluster is managed centrally.

2. Challenges of load balancing

The load balancer implements several load balancing algorithms to determine the suitable resource. However it faces several issues while distributing the load across available resources. Several major issues with their respective solutions are presented in the next section.

2.1 Increased web traffic

Over the last few years, the web traffic is increased very rapidly due to numerous registered websites and online transactions. As the numbers of requests are increased, the response of server becomes slow due to the limited number of open connections. The requests are added to the overall processing capability of resources. When incoming requests go beyond the capability of the resource, a resource crashes or fault occurs. Several authors analyzed and suggested the solution to resolve the issue. The first solution is the server upgradation in which requests are handled by a more powerful server for a while. But, scalability, interruption, and maintenance issues are associated with this solution. Another solution is the outsourcing in which requests are sent to another suitable server for speedy response. But this approach is costly and has limited control over the QoS issues. Chen et al. [4] suggested that the web page size and number of users both affect the system response time.

The most favorable solution is to use the multiple servers with an efficient load balancer which balances the load among servers. The performances of these servers are analyzed through queueing models or waiting line models. Broadly, two types of load balancing models are used to analyze the web server performance. Each approach has its benefits, applications, and limitations.

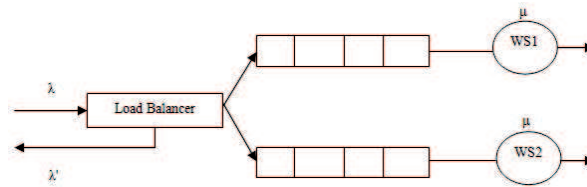


Figure 7.
 Centralized queueing model.

2.1.1 Centralized queueing model for load balancing

In this mechanism, homogeneous servers with finite buffer sizes are used as shown in **Figure 7**. The load balancer receives request from the user and redirects the request among servers using one of these routing policy:

- Random policy
- RR policy
- Shortest queue policy

Zhang and Fan [5] compared these policies in terms of rejection rate and system response time. They analyzed that these algorithms perform well when traffic is light. But when web traffic becomes high, shortest queue policy performs better than random and RR policy. The number of rejections in RR and random policy is increased as the traffic increases. Singh and Kumar [6] presented a queueing algorithm for measuring the overloading and serving capacity of server in distributed load balancing environment. The algorithm performs better in both homogeneous and heterogeneous environment than the remaining capacity (RC) and server content based queue (QSC) algorithms.

2.1.2 Distributed queueing model for load balancing

These mechanisms address the network latency issue also, which avoids network congestion. The queueing models follow certain arrival and distribution rules to distribute the requests. Zhang and Fan [5] suggested that distributed queueing models perform well in heavy traffic conditions. Routing decisions are taken on the basis of queue length differences of web servers. The collected information is used in traffic distribution for improving the performance of web servers. Singh and Kumar [7] suggested that task completion time directly affects the queue length of the web server. They presented a model based on the ratio factor of the task's average completion time. The model is compared with the model presented by Birdwell et al. [8], and it performs better for two performance metrics: average queue length and average waiting time of web servers.

Li et al. [9] analyzed network delay and presented a delay controlled load balancing approach for improving network performance. However, the approach has limited applicability and is suitable for stable path states.

2.2 Resource selection and task allocation

Many researchers have addressed the problem of resource selection and task allocation for the fair perspective of load balancing. It is the responsibility of load balancer to map resource and task before actual execution as shown in **Figure 8**.

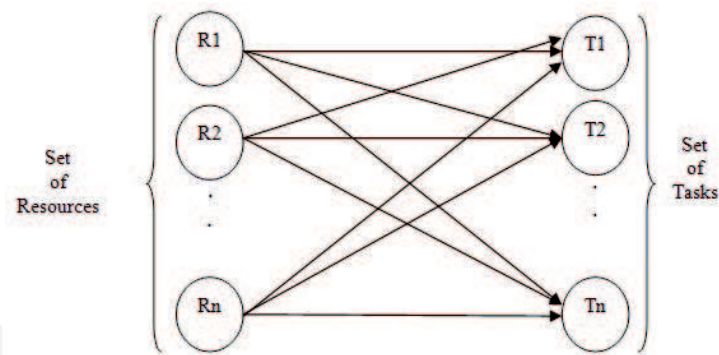


Figure 8.
Task and resource allocation model.

The resource management consists of two major functions: resource provisioning and resource monitoring. In resource provisioning, the user submits task to the broker with various predefined QoS constraints. The broker is responsible to find the suitable resource for task execution. The resource scheduling is all about mapping and execution of task on the appropriate resource as shown in **Figure 9**. Various types of resources that need to be managed are shown in **Figure 10**.

Hao et al. [10] categorized the resource in three categories—underloaded, normal loaded, and overloaded. The scheduler assigns the task to underloaded or normal-loaded resources only. Chang et al. [11] categorized the resources into L discrete levels for selecting the fittest resource for task execution. Arabnejad and Barbosa [12] presented a budget-based task scheduling and calculated the worthiness of all the resources for resource selection.

Naik et al. [13] presented a value function to select a resource for task execution. A value function is calculated using completion ratio and historic information of a resource. For minimizing the data transfer between the resources, Cheng et al. [14]

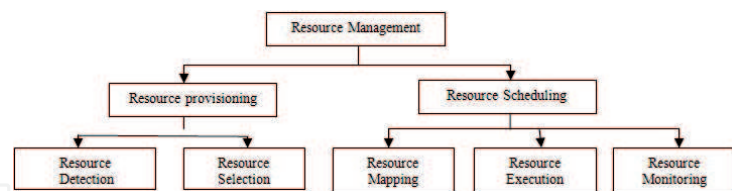


Figure 9.
Resource management classifications.

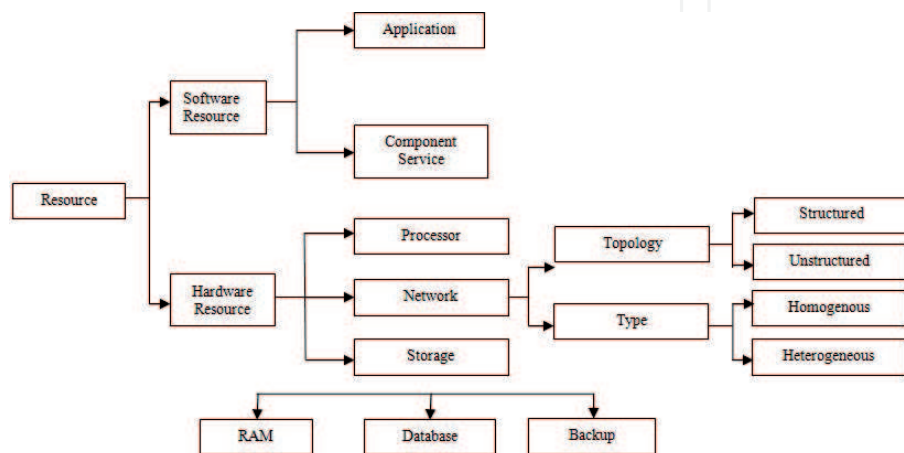


Figure 10.
Types of resources.

used a hypergraph which identifies task and data dependency. Tasks that use similar data are assigned to the same resource to decrease the cost indirectly. Abdelrouf et al. [15] used a genetic algorithm for producing chromosomes. A fitness function is used for generating chromosomes. Individuals who have higher fitness value will only proceed for further chromosome reproduction.

Murugesan and Chellappan [16] suggested deadline and budget-based resource selection method for divisible workloads. The method assigns the appropriate resource in terms of cost from the list of available resources. Shah et al. [17] also claimed a linear programming-based resource allocation method for divisible workloads. The job is categorized in appropriate sizes to allocate on available resources. Singh and Kumar [18] improved the resource selection method presented by Singhal et al. [19] by determining the task workload and resource availability, respectively. Ang et al. [20] introduced a resource allocation mechanism by considering the requirement of user as well as service provider.

Various researchers suggested numerous techniques for heterogeneous task allocation [21–24]. Adhikari et al. [25] presented a task assignment mechanism that provides reduced makespan and execution cost in cloud environment. They have used Bat algorithm for selecting the suitable cluster with faster convergence. Raman et al. [26] provide improvements of traditional round robin (RR) task scheduling which performs well when all the resources have equal serving capacity. In heterogeneous environment, it does not provide prominent results. Therefore, a weight is assigned to each server which represents the priority of selecting a server. The algorithm performs well in distributing the load more efficiently than RR scheduling algorithm.

Pham and Huh [27] analyzed the cloud-fog environment and presented a task scheduling algorithm. The suggestion behind the presented algorithm is the association between fog nodes and cloud nodes to decrease the makespan and price of cloud resources. If the computation is not feasible on fog node, then tasks are executed on cloud node. Several constraints like deadline and budget can enhance the algorithm efficiency and applicability.

Wu [28] presented a task scheduling for embedded systems to enhance the performance of real applications in CloudIoT paradigm. These approaches are used in real-time networks where time constraints are strictly followed. The algorithm increases the scheduling success rate of real-time task on heterogeneous web servers. Moschakis and Karatza [29] analyzed the workload generated by IoT devices and scheduled them on multi cloud-based system. The least loaded server is selected by global dispatcher for scheduling IoT jobs.

Grandinetti et al. [30] presented an offline mathematical formula to improve task scheduling and average waiting time. Xu et al. [31] presented a task scheduling algorithm which determines crossover and mutation operations for mapping between tasks and resources. Kamalinia and Ghaffari [32] addressed the task scheduling as an NP complete problem. They also used a genetic algorithm to design task scheduling problem to improve makespan and resource efficiency. The presented scheduling algorithm reduces the communication cost among processors by using meta-heuristic methods.

2.3 Load measurement

The load measurement is very important and crucial activity in distributed environment. Various load balancing algorithms determine resource load condition before real implementation of task. Various performance metrics like fault tolerance, waiting time, response time, etc. can effectively be optimized by measuring the current load of a resource. Many authors addressed this issue and presented various resource provisioning techniques for effective distribution of incoming load.

Patel and Tripathy [33] categorized the resources in three categories: under-loaded, normal-loaded, and overloaded to manage the load of available resources. Before assigning a task to a resource, the scheduler checks the current load of each resource and selects the underloaded or normal-loaded resource for task execution. Task length, processing element capacity, and deadline constraints are the factors that are considered to determine the current load of each resource. If a resource becomes overloaded, the unfinished tasks are shifted to another suitable resource for completing their execution. Checkpoint mechanism is used to save and resume the task state which greatly decreases the average response time and task resubmission time and improves the system throughput.

Liu et al. [34] advised that resource provisioning techniques may balance the resource load effectively. They presented peer load balance provision compares the demand and resource capacity by considering requirement of both customer and service provider. The presented mechanism reduced the cost and average response time than other existing methods.

Rathore and Chana [35] determined a dynamic threshold value based on standard deviation for load balancing and job migration. For job migration, the resources are categorized and the average load of each cluster is compared with processing element's threshold value. For load balancing, tasks are selected randomly either from underloaded or overloaded resource collection.

Kaushik and Vidyarthi [36] consider various parameters for effective job scheduling and resource allocation. The presented model selects the best cluster in terms of increased system reliability and reduced energy consumption and balances the system load efficiently. The customer can prioritize their choices to select the suitable cluster for task execution. An effective approach for determining job migration overhead can increase the model adaptability in real scenarios.

2.4 Cost optimization

Load balancing algorithm maps task to various heterogeneous resources based on predefined objectives. The major objective of load balancer is to optimize task completion time, resource cost, and its utilization. Several authors addressed the cost issue and provide possible solutions for its optimization.

Garg and Singh [37] suggested an adaptive workflow scheduling (AWS) by considering resource cost and communication cost between task and resources. Due to heterogeneous nature of resources, final cost is calculated periodically. Chaisiri et al. [38] analyzed the resource provisioning phases and suggested that reservation method provides reduced cost than on demand methods. Broadly, there are three stages in resource provisioning:

- Resource reservation
- Resource expanding
- Resource on demand

In the first stage, the cloud broker arranged the resources in advance without experiencing the customer requirement. In the second phase, the customer requirement and resource cost are comprehended, and the resource overutilization or underutilization is identified. If customer requirement is greater than reserved resources, the broker could request for additional resources on pay-per-use basis. Here, the on demand phase started. In on demand phase, the customer must know the appropriate future requirement which is difficult to estimate in cloud environment.

Singh and Kumar [39] presented a cost optimization method based on process activity. Processing cost and waiting time are determined by using activity time, resource utilization, and variability factor to check the method efficiency. Bittencourt and Madeira [40] presented a cost optimization method for hybrid cloud. The clouds can be categorized in three categories based on resource availability: public cloud, private cloud, and hybrid cloud. A user can use the services of public cloud by using pay-per-use method. Private clouds belong to individuals and offer free variety of services. In hybrid cloud, resources from public cloud are aggregated as per requirement. Bittencourt and Madeira [40] identified the method for appropriate resource.

Cao et al. [41] analyzed that each task is different from each other in cloud environment. They suggested an activity-based task scheduling approach for task reduction. The presented algorithm performs well than traditional task assignment approaches in terms of cost reduction.

Efficient resource provisioning plays a vital role in reducing the cost of task execution. Suresh and Varatharajan [42] presented a particle swarm optimization (PSO)-based resource provisioning algorithm. PSO is adopted to select the appropriate resource for cost optimization. Three performance metrics task execution time, memory usage, and cost are evaluated and compared with other existing methods. The simulation result shows that the presented PSO-based algorithm provides minimum execution time and memory usage with least cost than other state-of-the-art methods.

Salehan et al. [43] suggested auction-based resource allocation to meet the requirement of the customer and service provider. At the time of scheduling, resources are assigned to users that have highest bids. The algorithm provides highest profit and satisfies both the customers and service providers for multiple criteria than other existing methods. Nezarat and Dastghaibfard [44] map the resource allocation mechanism to economic-based supply and demand problem which provides better functionality with 17% profit with other existing methods.

Netjinda et al. [45] suggested a task scheduling for workflow applications. These workflow applications consist of dependent task with deadline constraints. The aim is to select the least cost cloud resource through PSO for workflow-based task execution. The effective task scheduling decreases the execution time which directly affects the final cost. By considering communication overhead, the model effectiveness and applicability can be increased in real cloud environment.

Chunlin and Layuan [46] presented a resource provisioning method for mobile clients. The mobile devices greatly depend on cloud resources for accessing data and performing operations. The aim is to select the optimal resource at least cost. The service provider executes the tasks on appropriate resources to get the maximum profit.

2.5 Fault tolerance

Fault tolerance is a mechanism that provides the estimated quality results even in the presence of faults. A system with its components and services can consider reliable only if it has fault tolerance capability. Therefore, fault tolerance issue has got a noticeable attention by the research community over the last decades [47].

Fault tolerance techniques can be categorized into two: proactive and reactive. Proactive techniques are prevention techniques that determine the controlled state for fault tolerance before they occur. The systems are continuously monitored for fault estimation. Proactive fault tolerance can be implemented in three ways: self-healing, preemption migration, and system rejuvenation. In self-healing, fault recovery procedures are periodically applied for autonomous recovery. In preemptive migration, the tasks are shifted from fault probable resource to another

resource. System rejuvenation is the mechanism in which periodic backups are taken for cleaning and removing errors from the system.

Another category is the reactive approaches that deal with faults after their occurrence. Reactive fault tolerance can also be implemented in three ways: job replication, job migration, and checkpoint. In job replication, several instances or copies of the same task make available on different resources. If one instance fails, task is executed on another instance. In job migration, tasks are migrated to another suitable resource for completing its execution. In checkpoint, task states are periodically saved and restarted from the last saved state instead of from the very beginning [47]. Several authors suggested fault tolerance mechanism and recovery solutions to resolve the issue.

Patel et al. [48] addressed resource failure issues and presented a checkpoint based recovery mechanism for task execution. If task does not complete its execution within deadline, then another suitable resource is selected for completing its execution. Before transferring it to another suitable resource, task state is saved and resumed for further execution through checkpoint. This results in reduced execution time, response time, and improved throughput than other existing methods.

Generally checkpoint increases the execution time that directly affects the execution cost. Egwutuoha et al. [49] use the process of redundant technique to reduce the task execution time. The presented technique is pretty good and reduces up to 40% checkpoint overhead. Choi et al. [50] identify the malicious users to provide fault tolerance scheduling in cloud environment. Any user which only use cloud services and reject other requests is treated as malicious user. The reputation is calculated to determine the malicious users. The work can be implemented to improve network reliability and task execution time in cloud paradigm.

Mei et al. [51] suggested that replication-based fault tolerance approaches waste lots of resources and also compromise with makespan. To resolve the issue, Mei et al. [51] presented fault tolerance scheduling mechanism that ensures successful completion of task execution. The limitation of replication is avoided by rescheduling the task for further execution. If scheduler identifies the failure, it reassigns task to another suitable resource and saves the wastage of resources. This mechanism reduces resource consumption and task execution time. However, costs are presumed for implementing scheduling, which limits the model applicability in real scenario.

Nazir et al. [52] use fault index for maintaining the history of resources. Fault index is determined based on successful and unsuccessful task completion on particular resource. Based on fault index value, grid broker replicates the task that can be used when fault occurs. Budget and time constraints are also considered at the time of task scheduling. The presented mechanism satisfies various QoS requirement, increases the reliability, and performs consistent in the existence of fault also.

Qureshi et al. [53] combined two fault tolerance techniques to inherit the favorable aspects. They perform hybridization of alternate task with retry and checkpoint mechanism and evaluate various performance metrics. The simulation result shows that alternate task with checkpoint mechanism performs better and improves system throughput than other existing methods.

Cloud facilitates the storage and access heterogeneous data in a distributed remote network. Due to dynamicity, network congestion and system faults are key factors for fault occurrence. Preventing the network from congestion and selection of suitable servers can avoid the fault conditions. Tamilvizhi and Parvathavarthini [54] suggested the concept of square matrix multiplication to manage the network traffic and avoid network congestion. The resource monitor predicts the fault

conditions and uses migration policies to avoid system failure. The presented fault tolerance mechanism provides reduced cost with less energy consumption.

Garg and Singh [55] observed various fault conditions and suggested a fault tolerance-based task scheduling algorithm in grid environment. A genetic algorithm is used to determine the resource capacity for task scheduling. The presented approach increased system reliability and reduced task execution time in grid environment.

2.6 Interoperability issue

Interoperability refers efficient migration and integration of heterogeneous applications and data to get the seamless services across domains. Various distributed applications exist to provide millions of services that differ in the services they offered:

- Distributed computing is a collection of various heterogeneous components that are located at remote locations, which coordinate with each other by message passing. Each component or processor has its own memory. It is a kind of parallel computing in which a task is split into subtasks to run on multiple components simultaneously.
- Grid computing is a network of computer resources that are connected to solve a complex problem. Each resource is loosely linked and runs independent task to achieve a common goal. Grid computing may be classified on the basis of scale and functionality. On the basis of scale, grid computing may be classified into two categories (**Table 1**), i.e., cluster grid and enterprise grid. Cluster means a group of similar kind of entities. So cluster grid provides services to the group or departmental level. Another type is enterprise grid that provides to share resources within the enterprise.
- Cloud computing provides on demand computer resources such as storage or computational resources without direct involvement of users. It has effective data management and computing framework for executing task in parallel to improve various QoS metrics.
- Fog computing is the extension of cloud computing which consists of multiple fog nodes that are directly connected to the physical devices. The difference between both technologies is that cloud is a centralized system while fog is distributed but decentralized system.

S. no	Classification criteria	Types of grid	Characteristics
1	Scale	Cluster	Computational services are limited to a group or a department
		Enterprise	Provides services within an enterprise
2	Functionality	Global	Comprises of collection of cluster grid
		Computational	Acts as an integrated processing resource
		Data	Coordinate and manage database information which is located at remote locations

Table 1.
Classification of grid.

- CloudIoT is an innovative trend which connects and manages millions of devices in very cost-effective manners that are dispersed globally. Cloud can profit by IoT to deal with real-world things by sharing the pool of highly computational resources rather than having local servers or personal devices to handle applications [56, 57].

Various authors analyzed the interoperability issues that are briefly presented with their respective solutions. Aazam et al. [58] focused on analyzed two complementary technologies: cloud computing and IoT. Various challenges and integration issues of CloudIoT framework are discussed. Data analysis, service provisioning, and storage are the future dimensions to improve the performance of CloudIoT model.

Botta et al. [59] also analyzed the integration issues of cloud and IoT. Both the technologies are analyzed separately based on applications, technology, issues, and challenges. The details of existing platforms and projects are presented that are currently implementing CloudIoT. Standardization, address resolution, multi-networking, and developments of APIs are some future directions to provide full potential to CloudIoT framework. Khodkari et al. [60] present the significance and requirement of CloudIoT paradigm. They presented complementary aspects of cloud computing and IoT and assure the QoS by evaluating the integrity requirement of both the technologies.

Bonomi et al. [61] analyzed characteristics, services, and applications of fog computing. They determined the importance of collaboration of fog and cloud and address that some applications need both cloud globalization and fog localization like big data and analytics.

2.7 QoS issues

The user submits the tasks with various QoS constraints (cost execution time, energy consumption, delay, etc.) to improve the performance in distributed environment. Researchers addressed several QoS issues and provide the solutions for meeting the objective. Aron and Chana [62] observed various QoS issues and identified four issues, i.e., cost, reliability, security, and time, for resource provisioning in grid environment. Service-level agreement (SLA) reduced the complexity of resource provisioning by maintaining up-to-date information of all the resources. The presented approach performs better in terms of resource utilization, cost, and customer satisfaction.

Popularity of cloud computing increased burden on distributed data centers. These data centers consumes excessive amount of energy to provide services and fulfill consumer satisfaction. Horri et al. [63] identified overloaded and underloaded servers and shift load from overloaded to underloaded resources. This makes a trade-off between energy consumption and SLA. HoseinyFarahabady et al. [64] suggested an objective function to reduce cost and performance improvement for resource allocation mechanism. Two test cases are considered: tasks with known running time and tasks with unknown running time. They listed Monte Carlo method to determine the task's unknown values.

3. Summary

A web server system used several load balancing techniques for distributing its load among available web resources. In this chapter, several load balancing issues have been identified for managing the web resources in distributed environment.

Detailed description of existing approaches, strength, limitation, and future scope has been analyzed and an adequate radiance has been thrown to these techniques. On the basis of abovementioned issues, several future dimensions have been identified that will be beneficial for the research community to achieve various objectives:

- Development of a resource allocation model which considers resource as well as task characteristics to optimize various QoS metrics
- Development of a fault tolerance load balancing model for partial executed tasks due to resource failure and construction of a resource selection policy for task execution
- Analysis of contextual relationship among CloudIoT issues and optimization through effective scheduling
- Development of an execution time prediction model for efficient resource provisioning, selection, and scheduling


IntechOpen

Author details

Anju Shukla, Shishir Kumar* and Harikesh Singh
Department of Computer Science and Engineering, Jaypee University of
Engineering and Technology, Guna, MP, India

*Address all correspondence to: dr.shishir@yahoo.com

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Alakeel AM. A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Information Security*. 2010;**10**(6):153-160
- [2] Khan RZ, Ali MF. An efficient diffusion load balancing algorithm in distributed system. *International Journal of Information Technology and Computer Science*. 2014;**6**(8):65-71
- [3] Kumar B, Richhariya V. Load Balancing of Web Server System Using Service Queue Length. *M.tech Scholar (CSE) Bhopal*. Vol. 5(5). 2014. Available from: http://www.ijetae.com/files/Volume4Issue5/IJETA_0514_14.pdf
- [4] Chen C, Bai Y, Chung C, Peng H. Performance measurement and queueing analysis of web servers with a variation of webpage size. In: *Proceedings of the International Conference on Computer Applications and Network Security*. 2011. pp. 170-174
- [5] Zhang Z, Fan W. Web server load balancing: A queueing analysis. *European Journal of Operational Research*. 2008;**186**(2):681-693
- [6] Singh H, Kumar S. WSQ: Web server queueing algorithm for dynamic load balancing. *Wireless Personal Communications*. 2015a;**80**(1):229-245
- [7] Singh H, Kumar S. Analysis & minimization of the effect of delay on load balancing for efficient web server queueing model. *International Journal of System Dynamics Applications*. 2014;**3**(4):1-16
- [8] Birdwell JD, Chiasson J, Tang Z, Abdallah C, Hayat MM, Wang T. Dynamic time delay models for load balancing. Part I: Deterministic models. In: *Advances in Time-Delay Systems*. Berlin, Heidelberg: Springer; 2004. pp. 355-370
- [9] Li M, Nishiyama H, Kato N, Mizutani K, Akashi O, Takahara A. On the fast-convergence of delay-based load balancing over multipaths for dynamic traffic environments. In: *International Conference on Wireless Communications and Signal Processing*. 2013. pp. 1-6
- [10] Hao Y, Liu G, Wen N. An enhanced load balancing mechanism based on deadline control on GridSim. *Future Generation Computer Systems*. 2012;**28**(4):657-665
- [11] Chang RS, Lin CF, Chen JJ. Selecting the most fitting resource for task execution. *Future Generation Computer Systems*. 2011;**27**(2):227-231
- [12] Arabnejad H, Barbosa JG. A budget constrained scheduling algorithm for workflow applications. *Journal of Grid Computing*. 2014;**12**(4):665-679
- [13] Naik KJ, Jagan A, Narayana NS. A novel algorithm for fault tolerant job scheduling and load balancing in grid computing environment. In: *International Conference on Green Computing and Internet of Things*. 2015. pp. 1113-1118
- [14] Cheng B, Guan X, Wu H. A hypergraph based task scheduling strategy for massive parallel spatial data processing on master-slave platforms. In: *23rd International Conference on Geoinformatics*. 2015. pp. 1-5
- [15] Abdelrouf W, Yousif A, Bashir MB. High exploitation genetic algorithm for job scheduling on grid computing. *International Journal of Grid and Distributed Computing*. 2016;**9**(3):221-228
- [16] Murugesan G, Chellappan C. An economic allocation of resources for divisible workloads in grid computing paradigm. *European Journal of Scientific Research*. 2011;**65**(3):434-443

- [17] Shah SNM, Mahmood AKB, Oxley A. Modified least cost method for grid resource allocation. In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. 2010. pp. 218-225
- [18] Singh H, Kumar S. Optimized resource allocation mechanism for web server grid. In: IEEE UP Section Conference on Electrical Computer and Electronics. 2015. pp. 1-6
- [19] Singhal S, Kumar M, Kant K. An economic allocation of resources in grid environment. In: International Conference on Information Systems and Computer Networks. 2013. pp. 185-190
- [20] Ang TF, Por LY, Liew CS. Dynamic pricing scheme for resource allocation in multi-cloud environment. *Malaysian Journal of Computer Science*. 2017;**30**(1): 1-17
- [21] Arunarani AR, Manjula D, Sugumaran V. Task scheduling techniques in cloud computing: A literature survey. *Future Generation Computer Systems*. 2019;**91**:407-415
- [22] Chinnathambi S, Santhanam A, Rajarathinam J, Senthilkumar M. Scheduling and checkpointing optimization algorithm for Byzantine fault tolerance in cloud clusters. *Cluster Computing*. 2019;**22**(6):14637-14650
- [23] Zhang Z. Resource selection method based on service capability in cloud manufacturing. *International Journal of Internet Manufacturing and Services*. 2018;**5**(2-3):169-183
- [24] Zhu LN, Li PH, Yang X, Shen GJ, Zhao YW. EE-RJMTFN: A novel manufacturing risk evaluation method for alternative resource selection in cloud manufacturing. *Concurrent Engineering*. 2019;**27**(1):3-13
- [25] Adhikari M, Nandy S, Amgoth T. Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. *Journal of Network and Computer Applications*. 2019;**128**:64-77
- [26] Raman K, Subramanyam A, Rao AA. Comparative analysis of distributed web server system load balancing algorithms using qualitative parameters. *VSRD International Journal of Computer Science and Information Technology*. 2011;**8**:592-600
- [27] Pham XQ, Huh EN. Towards task scheduling in a cloud-fog computing system. In: 18th Asia-Pacific Network Operations and Management Symposium. 2016. pp. 1-4
- [28] Wu DH. Task optimization scheduling algorithm in embedded system based on internet of things. *Applied Mechanics and Materials*. 2014;**513**:2398-2402
- [29] Moschakis IA, Karatza HD. Towards scheduling for Internet of Things applications on clouds: A simulated annealing approach. *Concurrency and Computation: Practice and Experience*. 2015;**27**(8):1886-1899
- [30] Grandinetti L, Pisacane O, Sheikhalishahi M. An approximate—Constraint method for a multi-objective job scheduling in the cloud. *Future Generation Computer Systems*. 2013;**29**(8):1901-1908
- [31] Xu Y, Li K, Hu J, Li K. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Information Sciences*. 2014;**270**:255-287
- [32] Kamalinia A, Ghaffari A. Hybrid task scheduling method for cloud computing by genetic and DE algorithms. *Wireless Personal Communications*. 2017;**97**(4):6301-6323
- [33] Patel DK, Tripathy C. An improved approach for load balancing among heterogeneous resources in

computational grids. *Engineering with Computers*. 2015;**31**(4):825-839

[34] Liu L, Mei H, Xie B. Towards a multi-QoS human-centric cloud computing load balance resource allocation method. *The Journal of Supercomputing*. 2016;**72**(7):2488-2501

[35] Rathore N, Chana I. Variable threshold-based hierarchical load balancing technique in grid. *Engineering with Computers*. 2015;**31**(3):597-615

[36] Kaushik A, Vidyarthi DP. An energy-efficient reliable grid scheduling model using NSGA-II. *Engineering with Computers*. 2016;**32**(3):355-376

[37] Garg R, Singh AK. Adaptive workflow scheduling in grid computing based on dynamic resource availability. *Engineering Science and Technology an International Journal*. 2015;**18**(2):256-269

[38] Chaisiri S, Lee BS, Niyato D. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*. 2012;**5**(2):164-177

[39] Singh H, Kumar S. Resource cost optimization for dynamic load balancing on web server system. *International Journal of Distributed and Cloud Computing*. 2014;**2**(1):7-18

[40] Bittencourt LF, Madeira ERM. HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*. 2011;**2**(3):207-227

[41] Cao Q, Wei ZB, Gong WM. An optimized algorithm for task scheduling based on activity based costing in cloud computing. In: *3rd International Conference on Bioinformatics and Biomedical Engineering*. 2009. pp. 1-3

[42] Suresh A, Varatharajan R. Competent resource provisioning

and distribution techniques for cloud computing environment. *Cluster Computing*. 2017:1-8

[43] Salehan A, Deldari H, Abrishami S. An online valuation-based sealed winner-bid auction game for resource allocation and pricing in clouds. *The Journal of Supercomputing*. 2017;**73**(11):4868-4905

[44] Nezarat A, Dastghaibfard G. A game theoretical model for profit maximization resource allocation in cloud environment with budget and deadline constraints. *The Journal of Supercomputing*. 2016;**72**(12):4737-4770

[45] Netjinda N, Sirinaovakul B, Achalakul T. Cost optimal scheduling in IaaS for dependent workload with particle swarm optimization. *The Journal of Supercomputing*. 2014;**68**(3):1579-1603

[46] Chunlin L, Layuan L. Cost and energy aware service provisioning for mobile client in cloud computing environment. *The Journal of Supercomputing*. 2015;**71**(4):1196-1223

[47] Hasan M, Goraya MS. Fault tolerance in cloud computing environment: A systematic survey. *Computers in Industry*. 2018;**99**:156-172

[48] Patel DK, Tripathy D, Tripathy C. An improved load-balancing mechanism based on deadline failure recovery on GridSim. *Engineering with Computers*. 2016;**32**(2):173-188

[49] Egwutuoha IP, Chen S, Levy D, Selic B. A fault tolerance framework for high performance computing in cloud. In: *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. 2012. pp. 709-710

[50] Choi S, Chung K, Yu H. Fault tolerance and QoS scheduling using CAN in mobile social cloud computing. *Cluster Computing*. 2014;**17**(3):911-926

- [51] Mei J, Li K, Zhou X, Li K. Fault-tolerant dynamic rescheduling for heterogeneous computing systems. *Journal of Grid Computing*. 2015;**13**(4):507-525
- [52] Nazir B, Qureshi K, Manuel P. Replication based fault tolerant job scheduling strategy for economy driven grid. *The Journal of Supercomputing*. 2015;**62**(2):855-873
- [53] Qureshi K, Khan FG, Manuel P, Nazir B. A hybrid fault tolerance technique in grid computing system. *The Journal of Supercomputing*. 2011;**56**(1):106-128
- [54] Tamilvizhi T, Parvathavarthini B. A novel method for adaptive fault tolerance during load balancing in cloud computing. *Cluster Computing*. 2017;1-14
- [55] Garg R, Singh AK. Fault tolerant task scheduling on computational grid using checkpointing under transient faults. *Arabian Journal for Science and Engineering*. 2014;**39**(12):8775-8791
- [56] Singh S, Chana I. A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of Grid Computing*. 2016;**14**(2):217-264
- [57] Yassa S, Chelouah R, Kadima H, Granado B. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *The Scientific World Journal*. 2013:1-13
- [58] Aazam M, Khan I, Alsaffar AA, Huh EN. Cloud of things: Integrating Internet of Things and cloud computing and the issues involved. In: *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST)* Islamabad, Pakistan. 2014. pp. 414-419
- [59] Botta A, De Donato W, Persico V, Pescapé A. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*. 2016;**56**:684-700
- [60] Khodkari H, Maghrebi SG, Branch R. Necessity of the integration Internet of Things and cloud services with quality of service assurance approach. *Bulletin de la Société Royale des Sciences de Liège*. 2016;**85**(1):434-445
- [61] Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the internet of things. In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. 2012. pp. 13-16
- [62] Aron R, Chana I. Formal QoS policy based grid resource provisioning framework. *Journal of Grid Computing*. 2012;**10**(2):249-264
- [63] Horri A, Mozafari MS, Dastghaibifard G. Novel resource allocation algorithms to performance and energy efficiency in cloud computing. *The Journal of Supercomputing*. 2014;**69**(3):1445-1461
- [64] HoseinyFarahabady M, Lee YC, Zomaya AY. Randomized approximation scheme for resource allocation in hybrid-cloud environment. *The Journal of Supercomputing*. 2014;**69**(2):576-592