Journal of Network and Computer Applications 88 (2017) 50–71

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Load-balancing algorithms in cloud computing: A survey

Einollah Jafarnejad Ghomi[a], Amir Masoud Rahmani[a,*], Nooruldeen Nasih Qader[b]

[a] Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Computer Science, University of Human Development, Sulaimanyah, Iraq

ARTICLE INFO

Keywords:
Cloud computing
Load balancing
Task scheduling
Hadoop MapReduce

ABSTRACT

Cloud computing is a modern paradigm to provide services through the Internet. Load balancing is a key aspect of cloud computing and avoids the situation in which some nodes become overloaded while the others are idle or have little work to do. Load balancing can improve the Quality of Service (QoS) metrics, including response time, cost, throughput, performance and resource utilization.

In this paper, we study the literature on the task scheduling and load-balancing algorithms and present a new classification of such algorithms, for example, Hadoop MapReduce load balancing category, Natural Phenomena-based load balancing category, Agent-based load balancing category, General load balancing category, application-oriented category, network-aware category, and workflow specific category. Furthermore, we provide a review in each of these seven categories. Also. We provide insights into the identification of open issues and guidelines for future research.

1. Introduction

Cloud computing is a modern technology in the computer field to provide services to clients at any time. In a cloud computing system, resources are distributed all around the world for faster servicing to clients (Dasgupta et al., 2013; Apostu et al., 2013). The clients can easily access information via various devices such as laptops, cell phones, PDAs, and tablets. Cloud computing has faced many challenges, including security, efficient load balancing, resource scheduling, scaling, QoS management, data center energy consumption, data lock-in and service availability, and performance monitoring (Kaur et al., 2014; Malladi et al., 2015). Load balancing is one of the main challenges and concerns in cloud environments;(Jadeja and Modi, 2012) it is the process of assigning and reassigning the load among available resources in order to maximize throughput, while minimizing the cost and response time, improving performance and resource utilization as well as energy saving (Singh et al., 2016; Goyal et al., 2016). Service Level Agreement (SLA) and user satisfaction could be provided by excellent load balancing techniques. Therefore, providing the efficient load-balancing algorithms and mechanisms is a key to the success of cloud computing environments. Several researches have been done in the field of load balancing and task scheduling in cloud environments. However, our studies showed that despite the key role of load-balancing algorithms in cloud computing, especially in the advent of big data, there are a few comprehensive reviews of these algorithms. First, we mention a few recent papers that have reviewed the load-balancing algorithms and mechanisms in cloud environments:

- Milani and Navimipour (2016) have presented a systematic review of the existing load balancing techniques. They classified the existing techniques based on different parameters. The authors compared some popular load-balancing algorithms and presented their main properties, including their advantages and disadvantages. They also addressed the challenges of these algorithms and mentioned the open issues. However, their work lacks a discussion regarding the load balancing and task scheduling techniques in Hadoop MapReduce that is an issue nowadays.
- Mesbahi and Rahmani (2016) have studied state of the art load balancing techniques and the necessary requirements and considerations for designing and implementing suitable load-balancing algorithms for cloud environments. They presented a new classification of load balancing techniques, evaluated them based on suitable metrics and discussed their pros and cons. They also found that the recent load balancing techniques are focusing on energy saving. However, their work suffers from the lack of simulating the load balancing techniques by simulator tools; in addition, a discussion of open issues and future topics that researchers should focus on is also missing.
- Kanakala et al. (2015a, 2015b) have analyzed the performance of load balancing techniques in cloud computing environments. They studied several popular load-balancing algorithms and compared

* Corresponding author.
E-mail addresses: e.jafarnejad@srbiau.ac.ir (E. Jafarnejad Ghomi), rahmani@srbiau.ac.ir (A. Masoud Rahmani), nooruldeen.qader@uhd.edu.iq (N. Nasih Qader).

http://dx.doi.org/10.1016/j.jnca.2017.04.007
Received 31 December 2016; Received in revised form 6 March 2017; Accepted 7 April 2017
Available online 08 April 2017

them based on metrics such as throughput, speed, complexity, etc. They concluded that none of the reviewed algorithms were able to perform well in all the required areas of load balancing. However, they did not mention the current trend, future works, and open issues in the field of load balancing in cloud environments.

- Ivanisenko and Radivilova (2015) have studied major load-balancing algorithms in distributed systems. They classified the most used load-balancing algorithms in distributed systems, including cloud technology, cluster systems, and grid systems. They also presented a comparative analysis of different load-balancing algorithms on various efficiency indicators such as throughput, migration time, response time, etc. In their work, a description of the main features of load-balancing algorithms, analysis of their advantages, and defaults of each type of algorithms is also presented. Nevertheless, a discussion of challenges, open issues, and future trends is similarly missing.
- Farrag and Mahmoud (2015) have reviewed intelligent cloud algorithms for load balancing problems, including Genetic Algorithms (GA), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO). They also proposed an implementation of Ant Lion Optimizer (ALO) based cloud computing environment as an efficient algorithm, which was expected to supply the outcomes in load balancing. The authors found that these algorithms showed a better performance than traditional ones in terms of QoS, response time, and makespan. However, they did not evaluate their proposed algorithm in different scales of cloud systems by comparing its results.

To help the future researchers in the field of load balancing in designing novel algorithms and mechanisms, we surveyed the literature and analyzed state of the art mechanisms. Therefore, the purpose of this paper is to survey the existing techniques, describe their properties, and clarify their pros and cons. The main goals of this paper are as follows:

- Studying the existing load balancing mechanisms
- Providing a new classification of load balancing mechanisms
- Clarifying the advantages and disadvantage of the load-balancing algorithms in each class
- Outlining the key areas where new researches could be done to improve the load-balancing algorithms

The rest of the paper is organized as follows. Section 2 provides a literature review for the model, metrics, policies, and taxonomy of load-balancing algorithms. Challenges in cloud-based load-balancing algorithms are explained in Section 3. Section 4 provides a relatively comprehensive review of literature on the existing load balancing techniques and presents a new classification. Section 5 provides a discussion of the mentioned techniques and some useful statistics. Open issues are outlined in Section 6. Finally, in Section 7, we conclude our survey and provide future topics.

2. The load balancing model, metrics, and policies in literature

The model of load balancing is shown in Fig. 1 (Gupta et al., 2014), where we can see the load balancer receives users' requests and runs load-balancing algorithms to distribute the requests among the Virtual Machines (VMs). The load balancer decides which VM should be assigned to the next request. The data center controller is in charge of task management. Tasks are submitted to the load balancer, which performs load-balancing algorithm to assign tasks to a suitable VM. VM manager is in charge of VMs. Virtualization is a dominant technology in cloud computing. The main objective of virtualization is sharing expensive hardware among VMs. VM is a software implementation of a computer that operating systems and applications can

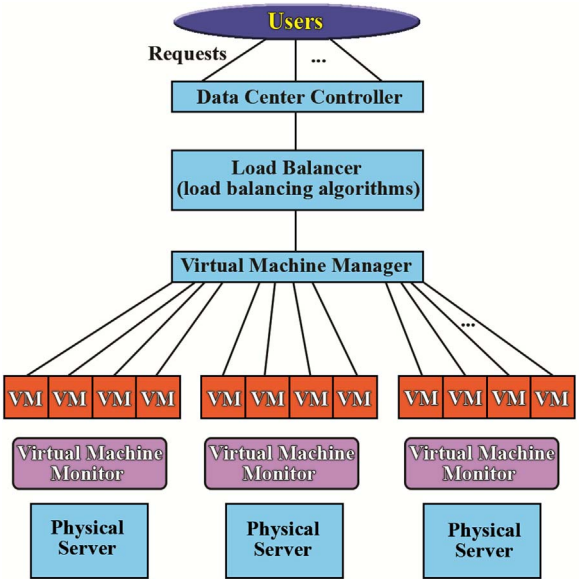

Fig. 1. The model of load balancing (Gupta et al., 2014).

run on. VMs process the requests of the users. Users are located all around the world and their requests are submitted randomly. Requests have to be assigned to VMs for processing. Therefore, the task assignment is a significant issue in cloud computing. If some VMs are overloaded while others are idle or have a little work to do, QoS will decrease. With the decreasing of QoS, users become unsatisfied and may leave the system and never return. A hypervisor or Virtual Machine Monitor (VMM) is used to create and manage the VMs. VMM provides four operations: multiplexing, suspension (storage), provision (resume), and life migration (Hwang et al., 2013). These operations are necessary for load balancing. In Ivanisenko and Radivilova (2015) it has been mentioned that load balancing has to consider two tasks: resource allocation and task scheduling. The result of these two tasks is the high availability of resources, energy saving, increasing the utilization of resources, reduction of cost of using resources, preserving the elasticity of cloud computing, and reduction of carbon emission.

2.1. Load balancing metrics

In this subsection, we review the metrics for load balancing in cloud computing. As mentioned before, researchers have proposed several load-balancing algorithms. Literature in load balancing (e.g., Daraghmi et al., 2015; Rastogi et al., 2015; Lua et al., 2011; Randles et al., 2010; Abdolhamid et al., 2014; Abdullahi et al., 2015;, Kansal et al., 2012;Milani and Navimipour, 2016) proposed metrics for applying load-balancing algorithms and we summarize them as follows:

- *Throughput*: This metric is used to calculate the number of processes completed per unit time.
- *Response time*: It measures the total time that the system takes to serve a submitted task.
- *Makespan:* This metric is used to calculate the maximum completion time or the time when the resources are allocated to a user.
- *Scalability*: It is the ability of an algorithm to perform uniform load balancing in the system according to the requirements upon increasing the number of nodes. The preferred algorithm is highly scalable.
- *Fault tolerance*: It determines the capability of the algorithm to perform load balancing in the event of some failures in some nodes or links.
- *Migration time*: The amount of time required to transfer a task from

an overloaded node to an under-loaded one.
- *Degree of imbalance*: This metric measures the imbalance among VMs.
- *Performance:* It measures the system efficiency after performing a load-balancing algorithm.
- *Energy consumption*: It calculates the amount of energy consumed by all nodes. Load balancing helps to avoid overheating and therefore reducing energy usage by balancing the load across all the nodes.
- *Carbon emission*: It calculates the amount of carbon produced by all resources. Load balancing has a key role in minimizing this metric by moving loads from underloaded nodes and shutting them down.

2.2. Taxonomy of load-balancing algorithms

In this subsection, we present the existing classification of load-balancing algorithms. In some studies (Rastogi, 2015; Mishra et al., 2015; Bhatia et al., 2012) load-balancing algorithms were classified based on two factors: the state of the system and person who initiated the process. Algorithms based on the state of the system are classified as static and dynamic. Some static algorithms are Round Robin, Min-Min and Max-Min Algorithms, and Opportunistic Load Balancing (OLB) (Aditya et al., 2015). Some of the dynamic algorithms include examples such as Ant Colony Optimization (ACO) (Nishant et al., 2012), Honey Bee Foraging (Babu et al., 2013), and Throttled (Bhatia et al., 2012). Nearly all dynamic algorithms follow four steps (Neeraj et al., 2014; Rathore and Chana, 2013; Rathore et al., 2013):

- ***Load monitoring***: In this step, the load and the state of the resources are monitored
- ***Synchronization*:** In this step, the load and state information is exchanged.
- ***Rebalancing Criteria***: It is necessary to calculate a new work distribution and then make load-balancing decisions based on this new calculation.
- ***Task Migration***: In this step, the actual movement of the data occurs. When system decides to transfer a task or process, this step will run.

The characteristics of static algorithms are:

1. They decide based on a fixed rule, for example, input load
2. They are not flexible
3. They need prior knowledge about the system.

The characteristics of dynamic algorithms are:

1. They decide based on the current state of the system
2. They are flexible
3. They improve the performance of the system

Dynamic algorithms are divided into two classes: distributed and non-distributed. In the distributed approach, all nodes execute the dynamic load-balancing algorithm in the system and the task of load balancing is shared among them (Rastogi et al., 2015). The interactions of the system nodes take two forms: cooperative and non-cooperative. In the cooperative form, the nodes work together to achieve a common objective, for example, to decrease the response time of all tasks. In the non-cooperative form, each node works independently to achieve a local goal, for example, to decrease the response time of a local task. Non-distributed algorithms are divided into two classes: centralized and semi-distributed. In the centralized form, a single node called the central node executes the load-balancing algorithms and it is completely responsible for load balancing. The other nodes interact with the central node. In the semi-distributed approach, nodes in the system are divided into clusters and each cluster is of centralized form. The central

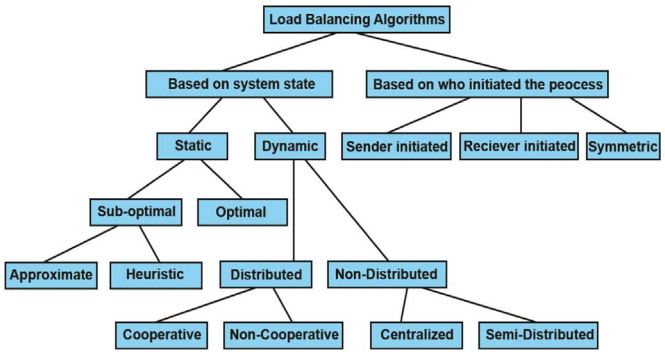

Fig. 2. State of the art classification of load balancing strategies.

nodes of the clusters achieve load balancing of the system. Static algorithms are divided into two categories: optimal, and sub-optimal (Neeraj et al., 2014). In optimal algorithms, the data center controller determines information about the tasks and resources and the load balancer can make an optimal allocation in a reasonable time. If the load balancer could not calculate an optimal decision for any reason, a sub-optimal allocation is calculated. In an approximate mechanism, the load-balancing algorithm terminates after finding a good solution, namely, it does not search the whole solution space. After that, the solution is evaluated by an objective function. In a heuristic manner, load-balancing algorithms make reasonable assumptions about tasks and resources. In this way, these algorithms make more adaptive decisions that are not limited by the assumptions. Algorithms in a sender-initiated strategy make decisions on arrival or creation of tasks, while algorithms in a receiver-initiated strategy make load-balancing decisions on the departure of finished tasks. In a symmetric strategy, either sender or receiver makes load-balancing decisions (Daraghmi et al., 2015; Alakeel et al., 2010; Rathore and Channa, 2011). A state of the art classification schema is shown in Fig. 2.

2.3. Policies in dynamic load-balancing algorithms

As mentioned before, dynamic load-balancing algorithms use the current state of the system. For this purpose, they apply some policies (Daraghmi et al., 2015; Kanakala et al., 2014; Alakeel et al., 2010; Yahaya et al., 2011; Mukhopadhyay et al., 2010; Babu et al., 2013; Kumar and Rana, 2015). These policies are:

Transfer Policy: This policy determines the conditions under which a task should be transferred from one node to another. Incoming tasks enter the transfer policy, which based on a rule determines the transfer of the task or processes it locally. This rule relies on the workload of each of the nodes. This policy includes task re-scheduling and task migration.
Selection policy: This policy determines which task should be transferred. It considers some factors for task selection, including the amount of overhead required for migration, the number of non-local system calls, and the execution time of the task.
Location Policy: This policy determines which nodes are under-loaded, and transfers tasks to them. It checks the availability of necessary services for task migration or task rescheduling in the targeted node.
Information Policy: This policy collects all information regarding the nodes in the system and the other policies use it for making their decision. It also determines the time when the information should be gathered. The relationships among different policies are as follows. Incoming tasks are intercepted by the transfer policy, which decides if they should be transferred to a remote node for the purpose of load balancing. If the task is not eligible for transferring, it will be processed locally. If the transfer policy decides that a task should be transferred, the location policy is triggered in order to find a remote

Table 1
Summary of load balancing policies.

Policy	Transfer policy	Selection policy	Location policy	Information policy
Description	Includes: • task re-scheduling • task migration • Based on thresholds in terms of load units.	Factors for selection a task to transfer: • Overhead of migration. • A number of the remote-system calls. • The execution time of the task.	• Find suitable partner for transfer task. • Checks the availability of the services necessary for migration within the Partner.	• Determine the time when the information about nodes has to gather. • There of three types of information policy: 1. Demand-driven policy. 2. Periodic policies. 3. State-change driven policy.

node for processing the task. If a remote partner is not found, the task will be processed locally, otherwise, the task will be transferred to the remote node. Information policy provides the necessary information for both transfer and location policies to assist them in making their decisions. These descriptions are summarized in Table 1.

3. Challenges in cloud-based load balancing

Review of the literature shows that load balancing in cloud computing has faced some challenges. Although the topic of load balancing has been broadly studied, based on the load balancing metrics, the current situation is far from an ideal one. In this section, we review the challenges in load balancing with the aim of designing typical load balancing strategies in the future. Some studies have mentioned challenges for the cloud-based load balancing (Palta and Jeet, 2014; Nuaimi et al., 2012; Kanakala and Reddy, 2015a, 2015b; Khiyaita et al., 2012; Ray and Sarkar, 2012; Sidhu and Kinger, 2013), including:

3.1. Virtual machine migration (time and security)

The service-on-demand nature of cloud computing implies that when there is a service request, the resources should be provided. Sometimes resources (often VMs) should be migrated from one physical server to another, possibly on a far location. Designers of load-balancing algorithms have to consider two issues in such cases: Time of migration that affects the performance and the probability of attacks (security issue).

3.2. Spatially distributed nodes in a cloud

Nodes in cloud computing are distributed geographically. The challenge in this case is that the load balancing algorithms should be designed so that they consider parameters such as the network bandwidth, communication speeds, the distances among nodes, and the distance between the client and resources.

3.3. Single point of failure

As mentioned in Section 2, some of the load-balancing algorithms are centralized. In such cases, if the node executing the algorithm (controller) fails, the whole system will crash because of that single point of failure. The challenge here is to design distributed or decentralized algorithms.

3.4. Algorithm complexity

The load-balancing algorithms should be simple in terms of implementation and operation. Complex algorithms have negative effects on the whole performance.

3.5. Emergence of small data centers in cloud computing

Small data centers are cheaper and consume less energy with respect to large data centers. Therefore, computing resources are distributed all around the world. The challenge here is to design load-balancing algorithms for an adequate response time.

3.6. Energy management

Load-balancing algorithms should be designed to minimize the amount of energy consumption. Therefore, they should follow the energy-aware task scheduling methodology (Vasic et al., 2009). Nowadays, the electricity used by Information Technology (IT) equipment is a great concern. In 2005, the total energy consumed by IT equipment was 1% of total power usage in the world (Koomey et al., 2008). Google data centers have consumed 260 million Watts of energy that is equal to 0.01% of the world's energy [37]. Research has shown that on an average, 30% of cloud servers exploit 10–15% of the resource capacity. Limited resource utilization increases the cost of cloud center operations and power usage (Vasic et al., 2009; Koomey et al., 2008). Due to the tendency of organizations and users to use cloud services, in the future, the installations of cloud providers will expand and thus the energy usage in this industry will increase rapidly. This increase in energy usage not only increases the cost of energy but also increases carbon-emission. If the number of servers in data centers reaches a threshold, their power usage can be as much as that of a city. High energy consumption has become a major concern for industry and society (Kansal et al., 2012).

What is the role of load balancing mechanisms in energy efficiency? In this section, we answer this question. Our survey of the literature [Ahmad et al., 2015; Vasic et al., 2009; Koomey et al., 2008) clarified that developing energy-saving approaches in load balancing is on the way. Load-balancing algorithms can be designed in ways that maximize the utilization of a physical server. For this purpose, they monitor the permanent workload of servers and migrate VMs from under-loaded physical servers to other servers and force some of the servers to enter a sleep state (shrinking the set of active machines). In Vasic and Barisits (2009) it has been shown that energy efficiency reaches a peak in full utilization of a machine. Energy efficient load balancing mechanisms have to make a certain contribution to power management too. In this way, load-balancing mechanisms are necessary for achieving green computing in a cloud. In green computing, two factors are important: Energy usage reduction and carbon emission reduction.

4. Survey on existing load balancing mechanisms

In this section, we survey the literature on the existing mechanisms for load balancing in cloud environments. For this purpose, we studied a number of journals and conference proceedings to present a new classification of them. We have classified the existing mechanisms into seven categories:

- Hadoop MapReduce load balancing category (HMR-category in this

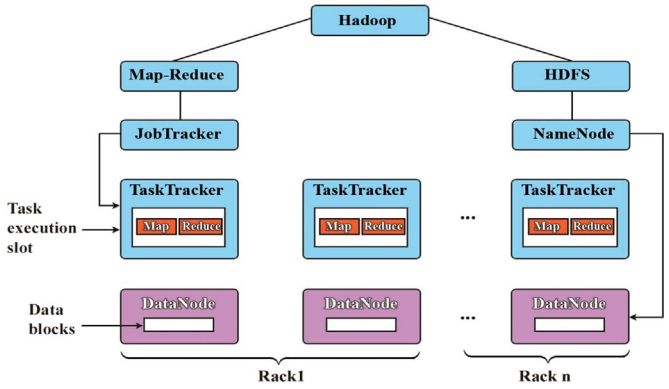

Fig. 3. The architecture of Hadoop.

paper)
- Natural Phenomena-based load balancing category (NPH-based in this paper)
- Agent-based load balancing category (Agent-based in this paper)
- General load balancing category (GLB-category in this paper)
- Application oriented load balancing (AOLB-category in the paper)
- Network-aware task scheduling and load balancing (NATSLB-category in the paper)
- Workflow specific scheduling algorithms (WFSA-category in the paper)

In the next subsections, we will address each category.

4.1. An Introduction to Hadoop MapReduce

A large volume of data is produced daily, for example from, Facebook, Twitter, Telegram, and WEB. These data sources together form big data. Hadoop is an open source framework for the storage and processing of big data on clusters of commodity machines (Hefny et al., 2014; Chethana et al., 2016; Dsouza et al., 2015). We have summarized the architecture of Hadoop in Fig. 3. Hadoop consists of two core components, namely Hadoop Distributed File System (HDFS) for data storage and MapReduce for data processing. HDFS and MapReduce follow master/slave architecture. A master node in HDFS is called NameNode and slaves or workers are called DataNodes. For storing a file, HDFS splits it into fixed-size blocks (i.e., 64 MB per block) and sends them to DataNodes. NameNode does mapping of blocks to workers. In MapReduce, the master node is called a JobTracker and slaves are called TaskTrakers. User's jobs are delivered to the JobTracker that is responsible for managing the jobs over a cluster and assigning tasks to TaskTrackers. MapReduce provides two interfaces called Map and Reduce for parallel processing. In general, the Map and Reduce functions divide the data that they operate on for load balancing purposes (Sui et al., 2011). TaskTracker executes each map and reduce task in a corresponding slot. Nodes in Hadoop spread over racks contained in one or several servers.

4.1.1. Load balancing schedulers in Hadoop

Hadoop simplifies cluster programming as it takes care of load balancing, parallelization, task scheduling, and fault tolerance automatically (Chethana et al., 2016; Vaidya et al., 2012; Rao et al., 2011). In other words, MapReduce, as the Google privacy strategy, hides the details of parallelization and distribution. Scheduling in Hadoop MapReduce is achieved at two levels: job level and task level (Dsouza et al., 2015). In job level scheduling, jobs are selected from a job queue (based on a scheduling strategy); in task-level scheduling, tasks of the job are scheduled. Scheduling strategies decide when and which machine a task is to be transferred for processing (load balancing). Hadoop uses First-In-First-Out (FIFO) strategy as its default scheduling, but it is pluggable for new scheduling algorithms. The scheduler is a pluggable module in Hadoop, and users can design their own dispatchers according to their actual application requirements (Khalil et al., 2013). Researchers have developed several scheduling algorithms for the MapReduce environment that contribute to the load balancing (Manjaly et al., 2013; Patel et al., 2015; Dagli et al., 2014; Selv et al., 2016). In addition, several load-balancing algorithms are developed as a plugin to standard MapReduce component of Hadoop. As mentioned before, any strategy used for an even load distribution among processing nodes is called load balancing. The main purpose of load balancing is to keep all processing nodes in use as much as possible, and not to leave any resources in an idle state while some other resources are being overloaded. Conceptually, a load-balancing algorithm implements a mapping function between the tasks and processing nodes (Destanoğlu et al., 2008). According to this definition of load balancing, scheduling algorithms do the task of load balancing. For this reason, we first surveyed and analyzed the load balancing schedulers in Hadoop.

4.1.1.1. FIFO scheduling. FIFO is the default scheduler in Hadoop that operates on a queue of jobs. In this scheduler, each job is divided into individual tasks that are assigned to a free slot for processing (Shaikh et al., 2017; Li et al., 2015). A job dominates the whole cluster and only after finishing a job, the next job can be processed. Therefore, in this scheduler job wait time, especially for short jobs, increases and no jobs could be preempted. The default FIFO job scheduler in Hadoop assumes that the submitted jobs are executed sequentially under a homogeneous cluster. However, it is very common that MapReduce is being deployed in a heterogeneous environment; the computing and data resources are shared for multiple users and applications.

4.1.1.2. Fair scheduler. Facebook developed the fair scheduler (Zaharia et al., 2009). In this algorithm, jobs are entered into pools (multiple queues) and in the case of multiple users; one pool is assigned to each user. Fair scheduler distributes the available resources among the pools and tries to give each user a fair share of the cluster over time, with each pool allocated a minimum number of Map and Reduce slots. If there are free slots in an idle pool, they may be allocated to other pools, while extra capacity in a pool is shared among the jobs. In contrast to FIFO, the fair scheduler supports preemption, therefore if a pool has not received its fair share for a long time, then the scheduler will preempt tasks in pools running over capacity in order to give the slots to the pool running under capacity. In this way, a long batch job cannot block short jobs for a long time (Polato et al., 2014; Xia et al., 2011; Zaharia et al., 2008).

4.1.1.3. Capacity scheduler. Yahoo! developed the Capacity scheduler to guarantee a fair allocation of resources among a large number of cluster users (Zaharia et al., 2009). For this purpose, it uses queues with a configurable number of task slots (Map or Reduce). Available resources are assigned to queues according to the priorities. If there are free resources in some queues, they are allocated to other queues (Hefny et al., 2014; Chethana et al., 2016; Polato et al., 2014). Within a queue, the priority of jobs is determined based on the job arrival time, class of the job, and priority settings for users according to the Service Level Agreement (SLA). When a slot in a TaskTracker becomes free, the scheduler chooses a job with the longest waiting time from a queue with the lowest load. Therefore, the capacity scheduler enforces cluster sharing among users, rather than among jobs, as is the case in the fair scheduler (Dsouza et al., 2015; Gautam et al., 2015).

4.1.1.4. Delay scheduler. The delay scheduler is an optimization of the fair scheduler, which eliminates the locality issues of the latter (Zaharia

et al., 2010). We consider a scenario in which a slot becomes free and we have to select a task of the job in front of a queue to process. It is possible that the data needed by this task does not exist on the node with a free slot. This is a locality problem. In the delay scheduler, this task is temporarily delayed until a slot in a node with the needed data becomes free. If the delayed time becomes long enough, to avoid starvation, the non-local task is allowed to schedule (Manjaly et al., 2013).

4.1.1.5. Longest Approximate Time to End (LATE). The LATE scheduler was developed to improve the job response time on Hadoop in heterogeneous environments (Lee et al., 2011). Some tasks may progress slowly due to CPU high load, race condition, temporary slowdown due to background processes, or slow background processes. These tasks are called speculative tasks. The LATE scheduler tries to find a slow task and execute an equivalent backup task on another node. This execution is called speculative execution. If the new copy of the task executes faster, the whole job performance will improve. The LATE Scheduler assigns priorities to slow or failed tasks for speculative execution and then selects the fastest nodes for that speculative execution. LATE scheduling improves the response time of Hadoop in heterogeneous environments.

4.1.1.6. Deadline constraint scheduler. The deadline constraint scheduler was designed to satisfy the user constraints (Kc et al., 2010). The goals of this scheduler are: (1) to be able to give users immediate feedback on whether the job can be completed within the given deadline or not and proceed with the execution if the deadline can be met. Otherwise, users have the option to resubmit with modified deadline requirements, (2) to maximize the number of jobs that can be run in a cluster while satisfying the time requirements of all jobs [Dsouza et al., 2015, 2015). Experiment results showed that when deadlines for the job is different, then the scheduler assigns a different number of tasks to Tasktracker and makes sure that the specified deadline is met.

We have thoroughly investigated and analyzed the scheduling algorithms in Hadoop. Our observations are summarized in Table 2. The analysis table contains the names of the algorithms proposed by researchers, parameters that they have tried to improve, advantages and disadvantages, and the tools through which they have simulated their experiments.

4.1.2. MapReduce optimization for load balancing

In this subsection, we review some of the algorithms proposed for MapReduce load balancing. In the standard Hadoop MapReduce, each data file is divided into fixed-sized blocks and each block has three replicas on three different DataNodes with two rules: (1) no two copies are on the same DataNode, (2) no two copies are on the same rack, provided that there are enough racks. However, in replica placement, the current load of DataNodes is irrelevant. A built-in tool called *the balancer* executes repeatedly, *the balancer* moving data blocks from the overloaded DataNodes to under-loaded ones (Lin et al., 2015). The balancer tool is used to balance an imbalanced cluster, but it would be better if we could keep the cluster as balanced as possible from scratch. Furthermore, using the balancer tool to load migration consumes a lot of system resources. Therefore, several researches have tried to provide load-balancing techniques in the Hadoop environment; we have reviewed some of them here.

- Valvåg et al. (2011, 2009)) proposed Cogset, a unified engine, for static load balancing. The authors have found that the loose coupling between HDFS and MapReduce engine is the cause of poor data

Table 2
An overview of the current load balancing scheduler in Hadoop MapReduce.

Algorithm	Load balancing	Utilization	Starvation	Adaptive	Job allocation	Throughput	Preemptive	Advantages	Disadvantages
FIFO	No	Low	Yes	No	Static	Low	No	• Simple implementation • Efficient	• Only one job at a time uses cluster resources • Low data locality when running a small job on cluster • No considering priority or job size • No-support multiuser execution
Fair	Relatively	High	No	Yes	Static	High	Yes	• Distributes resources among jobs fairly • Fast response time for both short and long jobs • Improves the QoS through job classification	• Does not consider the actual workload of nodes for task scheduling which may result imbalance
LATE	Yes	High	No	Yes	Static	High	Yes	• Suitable for heterogeneous environment • Improves the performance by reducing response time	• It use the past information • Is not suitable for environment with dynamic loading • It does not ensure reliability
Capacity	Yes	High	No	Yes	Static	High	No when job fails	• Improves resources utilization • It is flexible and efficient • It can be used in large clusters	• Users should acquire large amount of information about system to set a queue
Delay	Yes	High	May	Yes	Static	High	Yes	• Job response time may decrease	• Killing speculative tasks is instantaneous • Killing speculative tasks wastes the work performed by them • Does not ensure reliability
Deadline	Yes	High	No	Yes	Dynamic	High	Yes	• Focus on increasing system • Works better for large clusters utilization	• Homogeneous nodes are assumed • Dynamic job deadline can affect the response time • the job may not execute in the system due to its deadline constraints

locality for many applications. Rather than viewing the file system and execution engine as separate and loosely coupled components, Cogset combines them closely into a distributed storage system that supports parallel processing of data at the actual storage nodes. Cogset consists of two stages: (1) data storage is distributed over the cluster through partitioning and replications stage, (2) data access is achieved through a traversal stage. Due to the importance of load balancing and fault tolerance, the replication mechanism is an integral part of Cogset. The work provided a system with significantly better performance than Hadoop, in particular for small and moderate data volumes; it is not fully scalable.

- Ahmad et al. (2012) proposed Tarazu, a suite of optimizations of MapReduce, to address the problem of poor performance of MapReduce in heterogeneous clusters. The authors believe that the poor performance of MapReduce is due to two factors: (1) MapReduce causes excessive and burst network communication, (2) heterogeneity amplifies the Reduce load imbalance (Fadika et al., 2011). Tarazu consists of (1) *Communication-Aware Load Balancing of Map computation (CALB)* across the nodes, (2) *Communication-Aware Scheduling of Map computation (CAS)* to avoid burst network traffic, and (3) *Predictive Load Balancing of Reduce computation (PLB)* across the nodes. Authors showed by simulation that using Tarazu significantly improves the performance over a traditional Hadoop MapReduce in heterogeneous clusters.
- Kolb et al. (2011) proposed a block-based load-balancing algorithm, BlockSplit, to reduce search space of Entity Resolution (ER). ER is the task of identifying entities referring to the same real-world object. ER techniques usually compare pairs of entities by evaluating multiple similarity measures. They utilize a blocking key based on the values of one or several entity attributes to divide the input data into multiple partitions (blocks) and restrict the subsequent matching to entities of the same block. For example, it is sufficient to compare entities of the same manufacturer when matching product offers. The BlockSplit approach takes the size of the blocks into account and assigns entire blocks to reduce tasks if this does not violate the load balancing constraints. Larger blocks are split into smaller chunks based on the input partitions to enable their parallel matching within multiple Reduce tasks (Kolb et al., 2012). The evaluation in a real cloud environment demonstrated that the proposed algorithm was robust against data skew and scaled with the number of available nodes.
- Hsueh et al. (2014) proposed a block-based load-balancing algorithm for Entity Resolution with multiple keys in MapReduce. Actually, the authors extended the BlockSplit algorithm presented in Kolb et al. (2011) by considering more than one blocking key. In their algorithm, the load distribution in the Reduce phase is more precise because an entity pair may exist in a block only when the number of common blocking keys between the pair exceeds a certain threshold (i.e., *kc*). Since an entity may have more than one *kc* key, it needs to generate all the combinations of *kc* keys for potential key comparisons. The proposed algorithm features in the combination-based blocking and load-balanced matching. Experiments using the well-known CiteSeerX digital library showed that the proposed algorithm was both scalable and efficient.
- Hou et al. (2014) proposed a dynamic load-balancing algorithm for Hadoop MapReduce. Their algorithm balances the workload on a rack, while previous works tried to load balance between individual DataNodes. In the standard MapReduce and its optimizations, there was no way for Hadoop to guarantee that higher capability racks have more workload than lower capability racks. In other words, when assigning workload to DataNodes, the processing capacity was irrelevant. Their work has two novelties: (1) They concentrate on load balancing between racks; (2) They use Software Defined Network (SDN) to improve the data transfer. The results of simulation experiments showed that by moving the tasks from the busiest rack to a less busy one, the finished time of these tasks decreased substantially by adopting the algorithm.
- Vernica et al. (2012) proposed a suite of adaptive techniques to improve the MapReduce performance. The authors have ignored the key assumption of MapReduce that mappers run in isolation. They used an asynchronous channel called the Distributed Meta Data Store (DMDS) to share the situation information between mappers. They used these mappers, called Situation-Aware-Mappers (SAMs), to make traditional MapReduce more dynamic: (1) Adaptive Mappers, (2) Adaptive Combiners, (3) Adaptive Sampling and Partitioning. Adaptive Mappers merge small partitions into a virtual split thus making more splits that avoid frequent check pointing and load imbalance (Doulkeridis et al., 2013). Adaptive Combiners perform a hash-based aggregation instead of sort-based ones. In contrast to standard MapReduce, Adaptive Sampling creates local sampling dynamically, aggregates them, and produces a histogram. Adaptive Partitioning can exploit the global histogram to produce partitions of the same size for better load balancing. Although SAMs can solve the data skew problem, they cannot solve the computational skew in reducers (Shadkam et al., 2014). Experimental evaluation showed that the adaptive techniques dramatically improve the MapReduce performance and especially performance stability.
- Yang and Chen (2015) proposed an adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds. The paper makes improvements on the original speculative execution method of Hadoop (called Hadoop Speculative) and LATE Scheduler by proposing a new scheduling scheme known as Adaptive Task Allocation Scheduler (ATAS). The ATAS adopts more accurate methods to determine the response time and backup tasks that affect the system, which is expected to enhance the success ratio of backup tasks and thereby effectively increase the system's ability to respond. Simulation experiments showed that the proposed ATAS scheme could effectively enhance the processing performance of MapReduce.
- Bok et al. (2016) proposed a scheduling scheme to minimize the deadline miss of jobs to which deadlines are assigned when processing large multimedia data such as video and image in MapReduce frameworks. The proposed scheme improves job task processing speed by utilizing a replica node of the same data required to process jobs if a node where I/O load is excessive is about to process the jobs. A replica node refers to another node that has the data block required to process jobs at available nodes. If available nodes are not found despite the expected job completion time exceeding the deadline, the most non-urgent job is searched and the corresponding job task is temporarily suspended to fasten the job completion time. The performance evaluation result showed that the proposed scheme reduced completion time and improved the deadline success ratio.
- Ghoneem and Kulkarni (2016) introduced an adaptive scheduling technique for MapReduce scheduler to increase efficiency and performance when it is used in the heterogeneous environment. In this model, we make the scheduler aware of cluster resources and job requirement by providing the scheduler with a classification algorithm. This algorithm classifies jobs into two categories executable and non-executable. Then the executable jobs are assigned to the proper nodes to be executed successfully without failures, which increase the execution time of the job. This scheduler overcomes the problems of previous schedulers such as small job starvation, a sticky node in fair scheduler, and the mismatch between resource and job. The adaptive scheduler increase performance of MapReduce model in the heterogeneous environment while minimizing master node overhead and network traffic.
- Benifa and Dejey (2017) proposed a scheduling strategy named efficient locality and replica-aware scheduling (ELRAS) integrated with an autonomous replication scheme (ARS) to enhance the data locality and performs consistently in the heterogeneous environ-

Table 3
An overview of the current load balancing strategies for Hadoop MapReduce.

Year	Authors	Static/ Dynamic	Key Idea	Main Objective	Advantages	Disadvantages	Evaluation techniques	Journal/ Conference
2017	Ghoneem and Kulkarni (2016)	Dynamic	• Handling heterogeneity and scalability of Hadoop MapReduce	• Increasing the performance of MapReduce using an efficient scheduling	• Considering job requirements and node capabilities • Reducing makespan • No starvation • Scalable	• Finding content information is computationally expensive and time-consuming	• Implementing the scheduler on a cluster consisted of three nodes	Proceedings of the International Conference on Data Engineering and Communication Technology (Springer)
2017	Benfia et al. (2017)	Dynamic	• Using data locality in scheduler	• Improving throughput and reducing cross-rack communication	• Effective utilization of resources • Providing nearly optimal data locality • Reducing execution time • Adaptable for a wide range of applications	• No considering auto-scaling applications for commercial cloud environment	• A heterogeneous cluster built in Amazon EC2 Environment as a testbed	Wireless personal communication (Springer)
2016	Bok et al. (2016)	Dynamic	• Employing speculative tasks and block replication to avoid deadline miss	• Minimizing deadline miss of jobs • Providing MapReduce scheduling schema foe multimedia data	• Reducing completion time • Improving deadline success ratio • Considering both I/O loads in nodes and data locality	• No implementation in a real MapReduce environment	• Performance evaluation was conducted with personal computers whose OS was Windows 7	Multimedia tools and Applications (Springer)
2015	Yang and Chen (2015)	Dynamic	• Improving original LATE scheduler	• Enhancing MapReduce model by a task allocation scheduler	• Increasing throughput • Reducing mean tasks latency • Promote performance • Considering heterogeneity • Data locality, job types, and job importance are considered • Backup tasks quickly	• No, collect data of run-time tasks	• Installing heterogeneous cloud environment by physical and virtual machines • Using VMWare for managing nodes	Journal of Network and Computer applications (Elsevier)
2014	Hsueh et al. (2014)]	Dynamic	• Solving the Entity Resolution problem for a huge collection of entities with multiple blocking keys	• Load balancing among reducers • Reducing the response time of a job	• Using multiple keys is used to sort the entities, so the matching step can be accelerated • Efficiently solves the ER problem	• It may lead to duplicated comparison	• Experiments performed in a 30-nodes cluster which contains three types of nodes,	Twelfth Australian symposium
2014	Hou et al. (2014)	Dynamic	• Balancing the workload between different racks on a Hadoop cluster by considering the capability of DataNodes	• Decreasing the completion time of job tasks. • Maintain load balancing of clusters • Increasing the Map-Reduce performance	• Increasing the entire performance of Hadoop • Decreasing task completion time • Do load balancing between racks rather than Data Nodes	• Moving data between racks make communication overhead and consumes network bandwidth	• Using Mumak which is Apache's Hadoop Map-Reduce simulator	IEEE, Fourth International Conference on Big Data and Cloud Computing
2012	Ahmad et al. (2012)	Dynamic	• Proposing a suite of optimization for Map-Reduce	• Eliminating the load balancing problem of traditional Map-Reduce • Eliminating the communication overhead of traditional MapReduce in heterogamous clusters	• Eliminate the bottleneck due to shuffle or Map phase • Load balancing in Map-Side • Load balancing in Reduce-Side	• Tarazu considers clusters with two classes of hardware: Atom, Xeon, while literature showed that hardware in clusters has closely-related performance	• Using a heterogeneous cluster of 90 servers comprising 10 Xeon-based and 80 Atom-based server nodes	Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ACM)
2012	Vernica et al. (2012)	Dynamic	• Breaking the key assumption of isolation	• Improving cluster performance and simplify job	• Adding new runtime options to Hadoop and	• Situation-Aware Mappers continuously monitor the	• Running experiments on a 42-node IBM system x iDataPlex	ACM, International Conference on Extending

(continued on next page)

Table 3 *(continued)*

Year	Authors	Static/ Dynamic	Key Idea	Main Objective	Advantages	Disadvantages	Evaluation techniques	Journal/ Conference
			execution of Mappers in standard Map-Reduce	tuning ● Make Map-Reduce more dynamics	made them Situation Aware ● Flexible Map-Reduce ● Tasks in Situation-Aware mappers can alter their execution at runtime	execution of mappers. However, Situation-Aware Mappers cannot handle computational skew at the reducers.	dx340. Each node had two quad-core Intel Xeon E5540 64 bit 2.83 GHz processors, 32 GB RAM, and four SATA disks. The cluster consisted of 336 cores and 168 disks.	Database Technology
2011	Kolb et al. (2011)	Dynamic	● Even redistribution of data between map and reduce tasks	● Increasing the effectiveness and scalability of Map-Reduce ● Using blocking-techniques to facilitate entity resolution	● Handling data skew in Map-Reduce ● Suitable for heterogeneous cluster ● Is used for all kind of paired-wise similarity computation such as article comparison	● Considering one block key for any entity ● It may lead to imbalance in reduce phase due to using different-sized sub-block ● It consider multiple blocking key as many individual blocking key that is time-consuming	● Running experiments with real-world datasets on the Amazon EC2 cloud computing using Hadoop	Proceedings of the 20th ACM international conference on Information and knowledge management
2009	Valvåg et al. (2009)	Static	● Deterministic split of input dataset to some partitions	● Increasing the system efficiencyDecreasing the request response time	● Ease of implementation ● Supports appending to, readings and deleting files in a name space ● Avoid bottleneck ● Reducing the layering overhead of software running on top of the Map-Reduce	● Reconfiguring the placement of partitions in the presence of failure, entail copying a large amount of data between nodes ● Does not consider the heterogeneity	● Using a cluster of 12 Dell Power Edge 1995 machines interconnected by an HP ProCurve 4208VL with a 24-port 1 Gbps switched Ethernet module	Sixth IFIP International Conference on Network and Parallel Computing (IEEE)

Table 4
An overview of the current NPH-based category load balancing techniques.

Year	Authors	Key Idea	Main objectives	Advantages	Disadvantages	Evaluation techniques	Publication/ Presentation
2015	Yakhchi et al. (2015)	● Using Cuckoo Optimization Algorithm for load balancing in cloud computing	● Reducing energy consumption ● Maximizing resource utilization	● Simple policy for detecting over-/ under-utilized hosts ● Is suitable for green computing ● Using MMT Minimum Migration Time for VM migration	● Does not provide any security policy for VM migration ● It does not clear live/dead VM migration ● VM migration can increase response time ● SLA Avoidance ● No guaranty for QoS	● Simulation on CloudSim toolkit	IEEE 6th International Conference on Modeling, Simulation, and Applied Optimization
2013	Dasgupta et al. (2013)	● Using Genetic Algorithm (GA) for load balancing in cloud computing	● Load balancing of cloud infrastructure ● Minimizing the completion time of a given tasks set	● Improving system performance ● Reducing job time span ● Improve resource utilization	● Does not consider job priorities ● Low throughput ● No power saving ● Lack of scalability	● Simulation on CloudAnalyst toolkit	International Conference on Intelligence: Modeling Techniques and Applications(Elsevier)
2013	Babu et al. (2013)	● Inspiring the foraging behavior of honey bee for load balancing VMS	● Load balancing VMS for maximizing throughput ● Minimizing waiting time of tasks in queue	● Maximizing the throughput ● Waiting time of tasks become minimum	● Starvation for lower priority load ● Lack of scalability ● Single point of failure due to producing bees from single source	● Simulation on CloudSim toolkit	Applied Soft Computing (Elsevier)
2012	Nishant et al. (2012)	Using Ant Colony Optimization (ACO) for load balancing in cloud computing	Load balancing of nodes in cloud or grid systemsFinding optimal resources to process the submitted jobs	There is a single result set The task of each ant is specialized To avoid overloads due to ant creation, it uses a timer to suicide. Detection of over-/ under-loaded nodes and doing operations accordingly	Lack of scalabilityLack of throughputHead node make a bottleneck Lake of a mechanism for head node selectionIt is not clear the evaluation method It might cause data transmission break	No simulation or implementation	IEEE 14st International Conference on Modeling and Simulation

ment. ARS autonomously decides the data object be replicated by considering its popularity and removes the replica as it is idle. The results proved the efficiency of the algorithm for heterogeneous clusters and workloads.

Now that we have reviewed some approaches to load balancing in MapReduce, it is time to investigate and analyze them. In Table 3, we have summarized our analysis. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher.

4.2. Natural phenomena-based load balancing category

In this section, we have surveyed several load balancing strategies that are inspired by natural phenomena or biological behavior, for example, Ant-Colony, Honey-Bee, and Genetic algorithms.

- Yakhchi et al. (2015) proposed a load balancing method in cloud computing for energy saving by simulating the life of a family of birds called cuckoos. They have used Cuckoo Optimization Algorithm (COA). The cuckoos are species of birds that do not make nests for themselves. Cuckoos lay eggs in the nests of other birds with similar eggs to raise their young. For this, cuckoos search for the most suitable nests to lay eggs in order to maximize their eggs survival rate (Rajabioun et al., 2011). The load balancing method proposed in the paper consists of three different steps. In the first step, the COA is applied to detect over-utilized hosts. In the second step, one or more VMs are selected to migrate from the over-utilized host to other hosts. For this, they considered all the hosts except the over-utilized ones as under-utilized hosts and attempted to migrate all their VMs to the other host and switch them to sleep mode. It must be noted that if this process could not be completed, the under-utilized host is kept active. Finally, Minimum Migration Time (MMT) policy is used for selecting VMs from over-utilized and under-utilized hosts. The Simulation results demonstrated that the proposed approach reduced energy consumption. However, the method may cause SLA violation.
- Dasgupta et al. (2013) proposed a novel load-balancing strategy using a genetic algorithm (GA). The algorithm tries to balance the load of the cloud infrastructure while trying to minimize the completion time of a given task set. In the paper, a GA has been used as a soft computing approach, which uses the mechanism of natural selection strategy. It is a stochastic searching algorithm based on the mechanisms of natural selection and genetics. A simple GA is composed of three operations: (1) selection, (2) genetic operation, and (3) replacement. The algorithm creates a “population” of possible solutions to the problem and lets them “evolve” over multiple generations to find better and better solutions. The authors have tried to eliminate the challenge of the inappropriate distribution of the execution time, which is used to create the traffic on the server. Simulation results showed that the proposed algorithm outperformed the existing approaches like First Come First Serve (FCFS).
- Nishant et al. (2012) proposed a load-balancing algorithm using the Ant Colony Optimization (ACO). ACO is inspired from the ant colonies that work together in a foraging behavior. Inspired by this behavior, authors of Kabir et al. (2015) have used ACO for load balancing. In this algorithm, there is a head node that is chosen in such a way that it has the highest number of neighbor nodes. Ants move in two directions: (1) Forward movement; where ants move forward in a cloud to gather information about the nodes’ loads, (2) Backward movement; if an ant finds an under-loaded node (over-loaded node) on its path, it goes backward and redistributes the load among the cloud nodes. The main benefit of this approach lies in its detections of over-loaded and under-loaded nodes and thereby

performing operations based on the identified nodes.

- Babu et al. (2013) proposed a honeybee-based load balancing technique called HBB-LB that is nature-inspired; it is inspired by the honeybee foraging behavior. This technique takes into account the priorities of tasks to minimize the waiting time of tasks in the queue. This algorithm has modeled the behavior of honeybees in finding and reaping food. In cloud computing environments, whenever a VM is overloaded with multiple tasks, these tasks have to be removed and submitted to the under-loaded VMs of the same data center. Inspired by this natural phenomenon, the authors considered the removal of tasks from overloaded nodes as the honeybees do. When a task is submitted to a VM, it updates the number of priority tasks and the load of that VM and informs other tasks to help them in choosing a VM. Actually, in this scenario, the tasks are the honeybees and the VMs are the food sources. The experimental results showed that the algorithm improved the execution time and reduced the waiting time of tasks on the queue.

We investigated and analyzed the NPH-based category of load-balancing algorithms. The results are presented in Table 4. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher.

4.3. Agent-based load balancing techniques

In this section, we have reviewed the literature that proposed agent-based techniques for load balancing in cloud nodes. The dynamic nature of cloud computing is suitable for agent-based techniques. An agent is a piece of software that functions automatically and continuously decides for itself and figures out what needs to be done to satisfy its design objectives. A multi-agent system comprises a number of agents, which interact with each other. To be successful, the agents have to able to cooperate, coordinate and negotiate with each other. Cooperation is the process of working together, coordination is the process of reaching a state in which their actions are well suited, and in negotiation process, they agree on some parameters (Singha et al., 2015; Sim et al., 2011).

- Singh et al. (2015) proposed a novel autonomous agent-based load-balancing algorithm called A2LB for cloud environments. Their algorithm tries to balance the load among VMs through three agents: load agent, channel agent, and migration agent. Load and channel agents are static agents whereas migration agent is an ant, which is a special category of mobile agents. Load agent controls the information policy and calculates a load of VMs after allocating a job. A VM Load Fitness table supports the load agent. The fitness table maintains the list of all details of the VM properties in a data center such as id, memory, a fitness value, and load status of all VMs. Channel agent controls the transfer policy, selection policy, and location policy. Finally, the channel agent initiates the migration agents. They move to other data centers and communicate with the load agent of that data center to acquire the status of VMs present there, looking for the desired configuration. Result obtained through implementation proved that this algorithm works satisfactorily.
- Gutierrez-Garcia and Ramirez-Nafarrate (2015) proposed an agent-based load balancing technique for cloud data centers. The authors proposed a collaborative agent-based problem-solving technique capable of balancing workloads across commodity and heterogeneous servers by making use of VM live migration. They proposed an agent-based load balancing architecture composed of VM agents, server manager agents, and a front-end agent. They also proposed an agent-based load balancing mechanism for cloud environments composed of (1) migration heuristics that determines which VM should be migrated and its destination, (2) migration policies to select an AM for migration, (3) acceptance policies which determine which VMs should be accepted, and (4) a set of load balancing heuristics of the front-end to select the initial hosts of VMs. Simulation experiments showed that agents, through autonomous and dynamic collaboration, could efficiently balance loads in a distributed manner outperforming centralized approaches.
- Keshvadi and Faghih (2016) proposed a multi-agent load balancing system in an IaaS cloud environment. Their mechanism performs both receiver-initiated and sender-initiated approach to balance the IaaS load to minimize the waiting time of the tasks and guarantee the Service Level Agreement (SLA). The mechanism presented in the paper comprises of three agents: (1) VMM Agent, (2) Datacenter Monitor (DM), and (3) Negotiator Ant (NA). The VMM agent collects the CPU, memory and bandwidth utilization of the individual VM hosted by different types of tasks to monitor the load. A table for storing the state of the VMs supports this agent. The DM agent performs information policy in a datacenter by monitoring the VMM's information. This agent is supported by a table that maintains all information about the status and characteristics of all VMs in a datacenter. It categorizes the VMs based on their characteristics. DCM agents initiate NA agents. They move to other datacenters and communicate with the DCM agent of those datacenters to acquire the status of VMs there, searching for the desired configuration. Simulation results showed that the proposed algorithm was more efficient and there was a good improvement in the load-balance, response time, and makespan.
- Tasquier (2015) proposed an agent-based load balancer for multi-cloud environments. The author proposed an application-aware, multi-cloud, and load-balancer based on a mobile agent paradigm. The proposed architecture uses agents to monitor the status of the cloud infrastructure and detects the overload and/or under-utilization conditions. The multi-agent framework provides provisioning facilities to scale the application automatically to the under-loaded resources and/or to new resources acquired from other cloud providers. Furthermore, the agents are able to deallocate unused resources, thus leading to cost saving. The proposed architecture consists of three agents: (1) an executor agent, which represents the application running in multi-cloud environments, (2) a provisioner agent, which is responsible for managing the cloud infrastructure through adding and removing resources, (3) a monitor agent, which is responsible for monitoring the overload and/or under-utilization conditions. Users can overview the current state of the cloud environment through an additional agent called controllers. Moreover, each agent has mobility capabilities in order to migrate themselves autonomously on the multi-cloud infrastructure. The proposed algorithm overcame the provider lock-in challenge in the cloud and it was flexible to exploit the extreme elasticity.

We investigated and analyzed the agent-based load balancing techniques. The results are presented in Table 5. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher.

4.4. General load balancing techniques

In this section, we have surveyed and overviewed the literature in the field of general load balancing techniques. Although several algorithms are provided in this category, we have focused on new ones. For example, techniques such as First-In-First-Out (FIFO), Min-Min, Max-Min, Throttled, and Equally Spread Current Execution Load (ESCEL) are all belong to this category.

- Komarasamy and Muthuswamy (2016) proposed a novel approach for dynamic load balancing in a cloud environment. They called it

Table 5
An overview of agent-based load balancing techniques.

Year	Authors	Key Idea	Main objectives	Advantages	Disadvantages	Evaluation techniques	Journal/ Conference
2016	Keshvadi et al. (2015)	• Using Multiagent paradigm for dynamic load balancing across virtual machines • Using both senders- initiated and receiver-initiated approaches	• Maximizing resource utilization • Load balancing across virtual machines • Reducing the response time • Guarantee the SLA	• Increase the resource utilization • Avoid or reduce dynamic migration • Reducing the migration cost • Reducing the waiting time of tasks in queue	• Datacenter management ants do not have a timer for self-destroying and wait for message from parent	• Simulation using CloudSim toolkit • Agents are programmed using Java language	International Robotics and Automation Journal (MedCrave)
2015	Singh et al. (2015)	• Using software agents for load balancing in cloud computing	• Load balancing VMs • Reducing service time	• Improves resource utilization within a datacenter and multiple datacenters • Reduces response time	• Includes heavy computations • Migration agent does search for available VMs and is time-consuming	• Implementation using Java technology	International Conference on Advanced Computing Technologies and Applications (Elsevier)
2015	Gutierrez- Garcica and Ramirez-Nafarrate (2015)	• Using agent- based problem-solving technique for load balancing in a heterogeneous environment, live VM migration	• Efficient load balancing in a distributed manner	• Agents do load balancing using partial information about cloud datacenters • Considering the heterogeneity of servers and VMs	• does not estimate VM migration overhead • Provide no usage prediction mechanism • high migration overhead • It is a central approach and is not fully scalable	• Experiments performed using agent-based test-bed such as MapLoad and Red Hat • Test-bed was implemented in Java and JADE agent platform	Cluster Computing (Springer)
2015	Tasquier ([(2015)	• Using agent-based paradigm for developing an application aware multi-cloud load balancer	• Multi-cloud load balancing • Using full elasticity of cloud environments	• Using multi-cloud resources for load balancing • Overcoming the provider lock-in challenge in cloud • Is flexible to exploit the extreme elasticity	• Not implemented or simulated • Does not consider Quality of Service (QoS)	• Does not implemented or simulated	COLUMBIA International Publishing Journal of Cloud computing Research

Table 6
An overview of current GLB-category load balancing techniques.

Year	Authors	Key Idea	Main objectives	Advantages	Disadvantages	Evaluation techniques	Journal/Conference
2016	Komarasamy and Muthuswamy (2016)	• Using Bin Packing algorithm and VM reconfiguration for load balancing in cloud environment	• Load balancing virtual machines • Reducing job waiting time	• Handles the user requests during peak situation • Improves throughput • Increases resource utilization • Reduces job waiting time	• Using the second table, reservation table is space consuming • Does not consider energy saving	• Simulation in CloudSim toolkit	Indian journal of Science and Technology
2016	Chien et al. (2016)	• Using the estimating method of end of service time to load balancing VMs	• Reducing the response time • Load balancing VMs • Reducing the processing time	• Considers the actual instant processing power of VMs and job size to assign jobs to VMs • Is more effective • Improving response time and processing time	• Determining the actual instant processing power is complicated • Causes the problem of energy consumption and carbon emissions	• Simulation using CloudSim toolkit	IEEE 18th International Conference on Advance Communication Technology
2015	Domanal and Reddy (2015)	• Combining the Divide-and- Conquer methodology and throttled algorithm to load balancing in cloud environment	• Maximizing resource utilization • Intelligently assign jobs to VM for load balancing • Reducing the total execution time of tasks	• Maximize resource utilization • Reduces total execution time considerably	• Did not simulate in different workload situation • Does not consider deadline constraints	• Simulation in CloudSim toolkit	IEEE International Conference on cloud computing in emerging markets
2015	Kulkarni and BA (2015)	• Modifying Active VM Algorithm implemented in CloudAnalyst to load balancing VMs • Using reservation table between selection and allocation phases	• Uniform allocation of requests to VMs even during peak hours • Reduce the response time	• Load balancing VMs • It works well during the peak hours • Improve the elasticity	• Does not allocate the load uniformly to VMs across datacenters deployed at different geographical locations	• Simulation using CloudAnalyst toolkit	IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems

dynamic load balancing with effective bin packing and VM reconfiguration (DLBPR). DLBPR maps jobs into VMs based on the required processing speed of the job. The main objectives of their work were process the jobs within their deadline and to balance the load among the resources. In the proposed approach, the VMs are dynamically clustered as small, medium and large according to process speed and the jobs are mapped into a suitable VM existing in the cluster. The clusters are sometimes overloaded due to the arrival of a similar kind of job. In that situation, the VMs may either split or integrate the VMs in the data center based on the request of the job using a receiver-initiated approach. After reconfiguration, the VMs will dynamically regroup based on the processing speed of the VMs. The proposed methodology is composed of three tiers: (1) web tier, (2) schedule tier, (3) resource allocation tier. Users' requests are submitted to the web tier at any arbitrary time, which are forwarded to the scheduler tier. The deadline-based scheduler classifies and prioritizes the incoming jobs. These jobs are processed efficiently by VMs in the resource allocation tier. The proposed approach automatically improves the throughput and also increases the utilization of the resources.

- Domanal and Reddy (2015) proposed a hybrid scheduling algorithm for load balancing in a distributed environment by combining the methodology of Divide-and-Conquer and Throttled algorithms referred to as DCBT. The authors defined two scenarios. In scenario 1, they deployed a distributed environment that consists of a client, a load balancer and *n* nodes, which act as Request Handlers (RH) or servers. The requests come from different clients and the load balancer assigns incoming requests or tasks to the available RHs or servers. In scenario 2, the CloudSim simulator was used for simulation which consisted of a data center, VMs, servers, and the load balancer. Here, the client's requests were coming from the Internet users. In both scenarios, the DCBT algorithm was used for scheduling the incoming client's requests to the available RHs or VMs depending on a load of each machine. The proposed DCBT utilizes the VMs more efficiently while reducing the execution time of the tasks.
- Chien et al. (2016) proposed a novel load-balancing algorithm based on the method of estimating the end of service time. In their algorithm, they considered the actual instant processing power of VM and size of assigned jobs. They included two factors in the method of estimating the end-of-service time in VMs: (1) the selected VM should be able to finish it as soon as possible, (2) on the next allocation request, the load-balancing algorithm has to estimate the time that all queuing jobs and the next incoming job are completely done in every VM. The VM that corresponds to the earliest will be chosen to distribute the job. The simulation results showed that the proposed algorithm improves response time and processing time.
- Kulkarni and BA (2015) proposed a novel VM load-balancing algorithm that ensures a uniform assignment of requests to VMs even during peak hours (i.e., when the frequency of received requests in the data center is very high) to ensure faster response times to users. They modified the active VM algorithm implemented in the CloudAnalyst toolkit that has problems during the peak traffic situation. For this purpose, in addition to an allocation table, they used a reservation table between the phases of selection and allocation of VMs. The reservation table maintains the information of the VM reservations suggested by the load balancer to data center controller, but they did not update the allocation table until the notification arrives from allocation phase. The proposed load balancer takes into account both reservation table entry and allocation statistics table entry for a particular VM id to select a VM for the next request. The simulations results showed that the algorithm allocated requests to VM uniformly even during peak traffic situations.

We have investigated and analyzed the general load balancing category; the results are presented in Table 6. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher.

4.5. Application oriented load balancing techniques

In this section, we have surveyed and overviewed the literature in the field of application-oriented load balancing techniques.

- Wei et al. (2015) proposed an efficient application scheduling in mobile cloud computing based on MAX–MIN ant system. Firstly, the authors presented a local mobile cloud model with detail application scheduling structure. Secondly, they presented a scheduling algorithm for the mobile cloud model based on MAX–MIN Ant System (MMAS). Experiments results showed that the algorithm could effectively promote the performance of the mobile cloud.
- Wei et al. (2013) defined the Hybrid Local Mobile Cloud Model (HLMCM) consisting of cloudlet and mobile devices where cloudlet plays the role of a central broker while both neighboring mobile devices and cloudlet play the role of service provider. The objective of application scheduling is to maximize the profit as well as a lifetime of HLMCM while considering the capacity limitations of service providers. They proposed the Hybrid Ant Colony-based Application Scheduling (HACAS) algorithm to solve the scheduling problem. The algorithm only considers the available resources and does not consider overhead when calculating the advantage ratio of mobile devices for joining the cloudlet. Simulation results revealed that when the load of the system was heavy, HACAS algorithm could select those applications with maximum profit and minimum energy consumption.
- Deye et al. (2013) proposed an approach to make load balancing more dynamic to better manage the QoS of multi-instance applications in the cloud, the approach mainly limits the number of requests through a load balancer equipped with a queue for incoming user requests at given time to send and process the requests effectively. Simulation results showed that the approach improved the system performance.
- Sarood et al. (2012) developed techniques that reduce the gap between application performance on cloud and supercomputers. The scheme uses object migration to achieve load balance for tightly coupled parallel applications executing in virtualized environments that suffer from interfering jobs. While restoring load balance, it not only reduces the timing penalty caused by interfering jobs but also reduces energy consumption significantly.

We have investigated and analyzed the application-oriented load balancing techniques; the results are presented in Table 7. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher.

4.6. Network-aware task scheduling and load balancing

In this section, we have surveyed and overviewed the literature in the field of network-aware task scheduling and load balancing techniques.

- Shen et al. (2016) proposed a probabilistic network-aware task placement for MapReduce scheduling to minimize overall data transmission cost and delays and hence to reduce job completion time while balancing the transmission cost reduction and resource

Table 7
An overview of application-oriented load balancing techniques.

Year	Authors	Key Ideas	Main Objectives	Advantages	Disadvantages	Evaluation Techniques	Publication/Presentation
2016	Wei et al. (2015)	● Designing a mobile cloud model with efficient application processing	● Efficient exploiting of idle computing, storage and sensing capability of mobile device ● Improving QoS	● Improving performance ● Reducing energy consumption ● Improving the profit of scheduling	● No considering dynamic resource requirement of applications	CloudSim	Soft computing (springer)
2013	Deye et al. (2013)	● Limiting the number of requests through load balancer at a given time	● Making load balancing more dynamic ● Mitigating the effects of interference of sharing resources	● More scalable ● Reducing the rate of request rejection in two cases: with and without resource sharing ● Reducing response time	● No prediction the intensity of requests to obtain appropriate number of instances	CloudSim	International Conference on Cloud Computing and Big Data (IEEE)
2013	Wei et al. (2013)	● Verifying the definitions of mobile cloud computing ● Providing a Bio-inspiring application scheduling algorithm	● Reducing response latency ● Maximizing the profit ● Improving utilization	● Decreasing response time ● Decreasing energy consumption ● Integrated cloudlet with mobile device ● In heavy load situation selects an application with highs profit and min energy consumption	● No adaptive ● No distributed ● Mobile device can be overloaded ● No considering the completion time of application in the scheduling algorithm	CloudSim	Journal of Applied mathematics (Hindawi)
2012	Sarood et al. (2012)	● Using a message driven adaptive runtime system	● Load balancing for tightly coupled parallel application	● Reducing timing penalty caused by interfering jobs ● Reducing energy consumption ● Reducing execution time	● No implementation in a public cloud ● No decision making every time a load balancer is invoked	A testbed located at department of computer science at the university of illinois Urbara Champaaign	41st International Conference on Parallel Processing Workshops(IEEE)

utilization. They found that a task is faced with three challenges: (1) the available servers for running tasks dynamically change due to resource allocation and release over time; (2) the data fetching time of reduce tasks depends on both the placement of reduce tasks and the locations and sizes of the intermediate data produced by map tasks; (3) the link load on the routing path also has a significant impact on the data access latency. In order to reduce the latency, the link status of the network must be considered in the scheduling decision. The experimental results showed that the scheduling algorithm improved the job completion time and cluster resource utilization.

- Scharf et al. (2015) presented an extension of the OpenStack scheduler that enables a network-aware placement of instances by taking into account bandwidth constraints to and from nodes. Their solution follows the host-local network resource allocation, and it can be combined with bandwidth enforcement mechanisms such as rate limiting. The author presented a prototype that requires only very few changes in the OpenStack open source software. The authors showed that for heterogeneous VMs, a network-aware placement could achieve a larger network throughput and a more predictable performance, for example, by avoiding the congestion of network resources.
- Shen et al. (2016) proposed a new cloud job scheduler with elastic bandwidth reservation in clouds, in which each tenant only needs to specify job deadline and each job's reserved bandwidth is elastically determined by leveraging the elastic feature to maximize the total job rewards, which represent the worth of successful completion by deadlines. It also considers both the computational capacity of VMs and reserved VM bandwidth in job scheduling. A simulation and real cluster implementation results showed the efficiency and effectiveness of the algorithm in comparison with other scheduling algorithms.
- Kliazovich et al. (2016) proposed a model, called CA-DAG, for cloud computing applications taking into account a variety of communication resources of various types used in real systems. This communication-aware model of cloud applications allows making separate resource allocation decisions, assigning processors to handle computing jobs and network resources for information transmissions, such as requests for application database. It is based on DAGs that in addition to computing vertices include separate vertices to represent communications. The proposed communication-aware model creates space for optimization of many existing solutions to resource allocation and, together with performance and energy efficiency metrics of communication systems, will become an essential tool in the design of completely new scheduling schemes of improved efficiency.

We have investigated and analyzed the network-aware task scheduling and load balancing techniques; the results are presented in Table 8. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher.

4.7. Workflow specific scheduling algorithms

In this section we have surveyed and overviewed the literature on workflow specific scheduling algorithms; articles with regard to bag of tasks, dependent task, priority based task scheduling are reviewed.

- Ghosh and Banerjee (2016) proposed a new enhanced algorithm and implemented it in cloud computing environment, which adds a new feature like priority basis service of each request. Determining the priorities of a request, the request allocated to VMs. A Switching queue has proposed to hold the requests, which have been removed temporarily from the VM due to the arrival of higher priority request

Table 8
An overview of network-aware task scheduling and load balancing.

Year	Authors	Key Ideas	Main Objectives	Advantages	Disadvantages	Evaluation Techniques	Publication/Presentation
2016	Shen et al. (2016)	• Considering network topology and transmission cost in job scheduling	• Minimizing data transmission cost • Balancing transmission cost reduction and resource utilization	• Reducing job Completion time • Increasing cluster utilization • Minimizing delay	• The optimality of exponential model is not knownPerformance of the model did not evaluate under different network conditions	• Implementing the algorithm on Apache HadoopConduct experiments on a high-performance computing platform	IEEE International Conference on cluster computing
2016	Shen et al. (2016)	• Using elastic bandwidth reservation in clouds	• Finding a job schedule to satisfy the deadline requirements	• Minimizing total job rewards • Reducing job execution time • Efficiency and effectivenessReal implementation	• No automatic bandwidth reservation	• Using Facebook synthesized workload • Simulated datacenter with a rack consists of 40 machinesImplementing the algorithm	IEEE International Conference on Cloud Computing Technology an Science
2016	Kliazovich (2016)	• Using a communication-aware model in cloud computing (CA-DAG)	• Assigning processors to handle computing jobsUsing network resources for information transmission	• Considering the dynamics of cloud environment • Using context-data including network topology for scheduling • optimizing makespan	• No practical validation of proposed solution • No considering heterogeneity of cloud environment	• Using Winkler graph generator to produce workload • Testbed system architecture composed of a set of identical servers	Journal of Grid Computing (Springer)
2015	Scharf et al. (2015)	• Network aware- placement of instances by taking into account bandwidth constraints	• Extension of OpenStack Scheduler	• Increasing throughput • More predictable performance	• Fixing time granularity of control loop to 10 s • Relying on existing filters prevents the use of bandwidth as a metric • No considering the actual topology of network	A testbed setup of OpenStack “Icehouse” consisting of a few servers	IEEE 24th International Conference on Computer Communication and Network (ICCCN)

Table 9
An overview of workflow specific scheduling algorithms.

Year	Authors	Key Idea	Main Objectives	Advantages	Disadvantages	Evaluation techniques	Journal/Conference	Focus on
2017	Cai et al. (2017)	• Providing dynamic cloud resource scheduling algorithm for BoT workflow	• Full fill the workflow deadline	• Minimizing the cloud resource renting cost • Fully use the bag structure • A single type based greedy method for each ready BoT	• Expectation and variance based task execution time estimation method overestimate the practical task execution times to some degree	• Using ElasticSim tool	Journal of Future Generation Computer Systems (Elsevier)	Bag of tasks
2016	Ghosh and Banerjee (2016)	• Priority based service allocation of each user request	• Improving average execution time	• Reducing response time • Reducing execution time • Improving service quality	• May cause starvation for low priority requests • Long response time for low priority jobs	CloudSim simulation tool	International Conference on Inventive Computation Technology (IEEE)	Priority based
2016	Cinque et al. (2016)	• Providing failure-aware scheduling approaches and scalable monitoring for grid	• Improving the execution of heavy job batches	• Scalable • Improving performance • Reducing bandwidth consumption • Improving throughput	• No, consider the situations where multicast is not available	• Implementing on a real grid	Proceedings of the 31st Annual ACM Symposium on Applied Computing (ACM)	Dependable scheduling
2016	Bellavista et al. (2016)	• Using the publish/subscribe paradigm for intra-domain scenarios	• Providing scalable monitoring and enhanced dependable job scheduling	• Decentralized scheduler • A novel troubleshooting algorithm • Use standard technology to avoid vendor lock-in • Providing failure-aware scheduling	• Not fully scalable • No job completion efficiency	• implemented and tested in a real deployment on distributed data centers across Europe	Journal of Future Generation Computer Systems (Elsevier)	Dependable scheduling
2016	Kianpisheh et al. (2016)	• Using ant colony system to develop a robust workflow scheduler	• Minimizing the violation of workflow constraints • Minimizing probability of violations	• Reducing the probability of violation of workflow constraints • Reducing expected penalty at run-time • Decreasing makespan and cost • Considering budget and deadline of workflows	• No avoid run-time violations • No track the workflow at runtime	• Simulation on real world workflow	Cluster Computing (Springer)	Workflow Scheduling
2015	Moschakisa and Karatzaa (2015)	• Scheduling of bag of tasks applications	• Optimizing interlinked cloud systems • Optimizing performance and cost	• Reducing Makespan • Maintaining a good cost-performance trade-off • Increasing utilization • Improving performance	• The policy of spreading parallel tasks between the clouds is not clear • No real implementation in cloud system	• Proposed approach tested in a scientific federated cloud	Journal of Systems and Software (Elsevier)	Bag of tasks
2015	Zhang and Li (2015)	• Using an adaptive heuristic algorithm for workflow scheduling	• Proper mapping of tasks to resources	• Reducing response time • Optimizing makespan • Optimizing load balancing • Optimizing failure rate of tasks	• Not fully adaptive • Focus only on compute intensive applications • No considering communication-intensive	• CloudSim simulation tool	Third International Conference on Advanced Cloud and Big Data (IEEE)	Workflow Scheduling
2014	Jaikar et al. (2014)	• Presenting priority-based VM allocation algorithm	• Increasing resource utilization • Reducing energy consumption	• Considering a scientific federated cloud • Reducing total job execution time • Improving resource utilization • Increasing system performance	• No providing migration policies • There is no cost function for cross-cloud VM migration	Implementing the algorithm in OpenNebula	IEEE 3rd International Conference on Cloud Computing (CloudNet)	Priority based

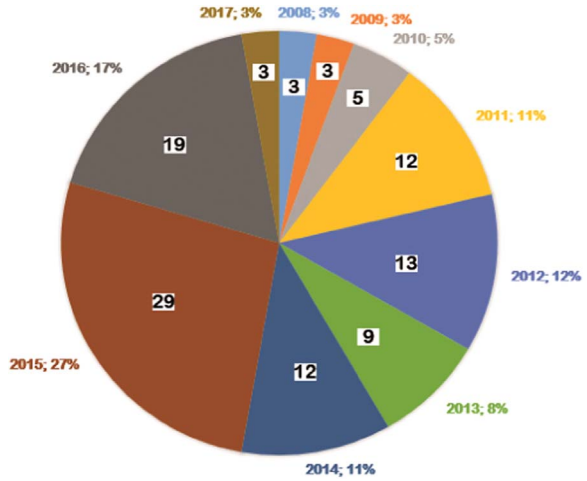

Fig. 4. The distribution of studied articles over time From 2008 until February 2017.

earlier. The authors analyzed the performance of their algorithm with respect to Throttled Load Balancing algorithms and Round Robin.

- Jaikar et al. (2014) proposed system architecture and the VM allocation algorithm for the load balancer in a scientific federated cloud. They tested the proposed approach in a scientific federated cloud. Experimental results showed that the proposed algorithm not only increased the utilization of resources but also reduced the energy consumption.
- Moschakisa and Karatzaa (2015) developed the simulated annealing and thermodynamic simulated annealing in the multi-criteria scheduling of a dynamic multi-cloud system with VMs of heterogeneous performance serving Bag-of-Tasks (BoT) applications. The scheduling heuristics applied, consider multiple criteria when scheduling said applications and try to optimize both for performance and cost. Simulation results indicated that the use of these heuristics could have a significant impact on performance while maintaining a good cost-performance trade-off.
- Cai et al. (2017) proposed A delay-based dynamic scheduling algorithm (DDS), DDS is a dynamic cloud resource provisioning and scheduling algorithm to minimize the resource renting cost while meeting workflow deadlines. New VMs are dynamically rented by the DDS according to the practical execution state and the estimated task execution times to fulfill the workflow deadline. The bag-based deadline division and bag-based delay scheduling strategies consider the bag structure to decrease the total renting cost. The results showed that the algorithm decreased the resource renting cost while guaranteeing the workflow deadline compared to the existing algorithms
- Cinque et al. (2016) proposed a Grid Architecture for scalable Monitoring and Enhanced dependable job ScHeduling (GAMESH). GAMESH is a completely distributed and highly efficient management infrastructure for the dissemination of monitoring data and troubleshooting of job execution failures in large-scale and multi-domain Grid environments. The solutions improve job processing throughput in both intra/inter-domain environments.
- Bellavista et al. (2016) proposed GAMESH, a Grid Architecture for scalable Monitoring and Enhanced dependable job ScHeduling. The proposed solution is conceived as a completely distributed and highly efficient management infrastructure. The paper relevantly extends the authors previous work appeared in Cinque et al. (2016). With respect to it, this extended version provides additional details about the effective design and implementation of selected and primary GAMESH components. In addition, it reports a novel and extensive measurements in both intra-domain and inter-domain deployments. Moreover, it provides the detailed description of the original Stochastic Reward Network models that have been produced to perform the simulation of the GAMESH failure-aware scheduling solution. The proposed solution improves job processing throughput in both intra/inter- domain environments.
- Kianpisheh et al. (2016) investigated the problem of workflow scheduling regarding the user-defined budget and deadline. The probability of violation of constraints (POV) has been used as robustness criteria for a workflow schedule at run-time. By aggregating the execution time distributions of the activities on the critical path the Probability Density Function (PDF) of makespan is computed. Ant Colony System has been utilized to minimize an aggregation of violation function and POV.
- Zhang and Li (2015) proposed an Improved Adaptive heuristic algorithm (IAHA). At first, the IAHA algorithm makes tasks prioritization in complex graph considering their impact on each other, based on graph topology. Through this technique, the completion time of application can be efficiently reduced. Then, it is based on adaptive crossover rate and mutation rate to cross and mutate to control and lead the algorithm to an optimized solution. The experimental results showed that the proposed method improved response time and makespan.

We have investigated and analyzed the network-aware task scheduling and load balancing techniques; the results are presented in Table 9. The analysis table contains article year, authors, key ideas, main objectives, advantages and disadvantages, evaluation techniques, and the journal or conference that the article presented. We also showed the name of the publisher. We specified in column “Focus on”, which subcategory the algorithm belongs to.

5. Discussion and statistics

In this section, we provide some statistics based on the studied articles. Fig. 4 shows the distribution of the reviewed articles by the year of publication from 2008 until February 2017. In the Figure we see the number of articles in each year on the corresponding slice; for example, the number of studied articles in 2015 is 29 that is the highest. The percentage of studied articles in each year is shown in the Figure too. Moreover, the number of articles in 2016 is noteworthy. Fig. 4 shows that 3% of the articles were published in 2017, 17% of them were published in 2016, 27% of them were published in 2015, 11% of them published in 2014, and 9% of them published in 2013. It means that 72% of the studied articles have been published in the last five years.

The distribution of the studied from different publishers is shown in Fig. 5. In the Figure, the article frequency of each publisher is shown on the corresponding slice, where 29 out of 108 total articles of journals belong to IEEE (27%). To further investigate the foundation journal of article, 12% of the literature is related to Elsevier, 10% of the literature is related to Springer, 8% of the literature belongs to ACM, 2% of the literature belongs to IJMECE, 2% of the literature is related to ACEEE, 42% of them published by others.

In Table 10 and Fig. 6, we showed how the studied articles addressed the load balancing QoS metrics. The information is extracted from Tables 3–9. By referring to Table 10, we can differentiate the articles based on single objective and multi-objective load balancing techniques. References Hsueh et al. (2014), Hou et al. (2014), Babu et al. (2013), Chien et al. (2016), Scharf et al. (2015), Shen et al. (2016), Bok et al. (2016), and Kliazovich et al. (2016) in the Table are single objective while the others are multi-objective. Fig. 6 shows that 22% of the studied techniques addressed the response time metric, 24% addressed the makespan, 19% addressed the resource utilization metric, 9% addressed the throughput, %9 addressed the energy saving, 9% addressed the scalability, 8% addressed the migration time metric. We see that the majority of techniques have concentrated on the response time and makespan metrics.

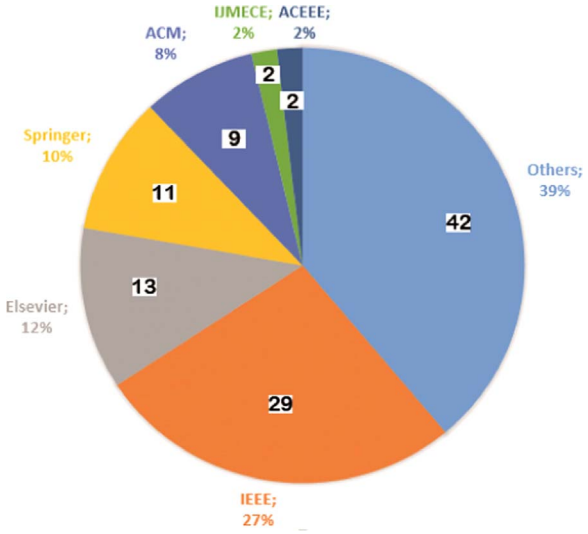

Fig. 5. The distribution of studied articles based on different publishers.

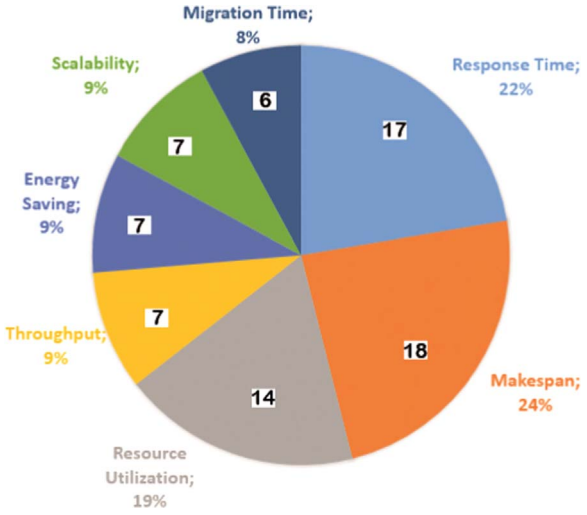

Fig. 6. An overview of load balancing metrics addressed by the reviewed techniques.

Evaluation techniques used in the articles and the corresponding statistics are shown in Fig. 7. We divided evaluation techniques in five classes: Real Testbed, CloudSim, Witten Program, CloudAnalyst and the others. Article frequencies in each class are shown on the corresponding slice; 19 articles used a real testbed for algorithm evaluation, eight articles used CloudSim, authors of three articles wrote a program for algorithm evaluation, two articles used CloudSim, and seven articles used other techniques. To further investigate the foundation evaluation techniques of the articles, 49% of the literature used a real testbed that is the highest, and 20% of them used ClousSim, 3% of them used an arbitrarily written program, 2% of them used CloudAnalyst, and 18% of them used other tools.

The venue types of papers are shown in Fig. 8. In the Figure, the absolute number of papers in each venue and the percentage are shown. It is shown that 20 papers presented on the conference, 16 papers published in journals, and two papers presented in a symposium. To further investigate the paper, 53% of the literature presented on the conference, 42% published in journals, and 5% presented in symposium.

6. Open issues and future trends

In this section, we offer major load balancing techniques issues that

Table 10
Load balancing QoS metrics in the reviewed techniques.

#	References	Energy saving	Migration time	Response time	Scalability	Resource utilization	Throughput	Makespan
1	Scharf et al. (2015)						•	
2	Shen et al. (2016)							•
3	Keliazovich (2016)							•
4	Hou et al. (2014)							•
5	Bok et al. (2016)							•
6	Hsueh et al. (2014)			•				
7	Babu et al. (2013)					•		
8	Chient et al. (2016)			•				
9	Yakhchi et al. (2015)	•	•			•	•	
10	Dasgupta et al. (2013)			•		•		•
11	Nishant et al. (2012)					•		•
12	Singh et al. (2015)		•	•	•	•		•
13	Gutierrez-Garcia and Ramirez-Nafarrate (2015)	•	•	•			•	
14	Keshvadi et al. (2015)		•	•		•	•	
15	Tasquire (2015)		•	•			•	•
16	Kumarasamy et al. (2015)	•		•	•	•		
17	Domanal and Reddy (2015)			•	•	•		•
18	Kulkarni and BA (2015)		•	•				
19	Ghoneem and Kulkarni (2016)				•			•
20	Benfia et al. (2017)				•	•		•
21	Yang and Chen (2015)			•			•	
22	Wei et al. (2015)	•		•				
23	Deye et al. (2013)			•	•			
24	Wei et al. (2013)	•		•		•		
25	Sarood et al. (2012)	•						•
26	Shen et al. (2016)			•		•		•
27	Ghosh and Banerjee (2016)			•				•
28	Cinque et al. (2016)				•		•	
29	Kianpisheh et al. (2016)					•		•
30	Maschakis et al. (2015)					•		•
31	Zhang and Li (2015)			•				•
32	Jaikar et al. (2014)	•						•
33	Total	7	6	17	7	14	7	18

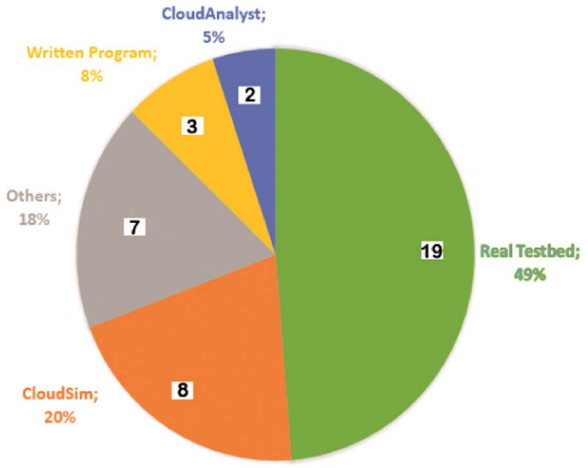

Fig. 7. Evaluation techniques used by articles.

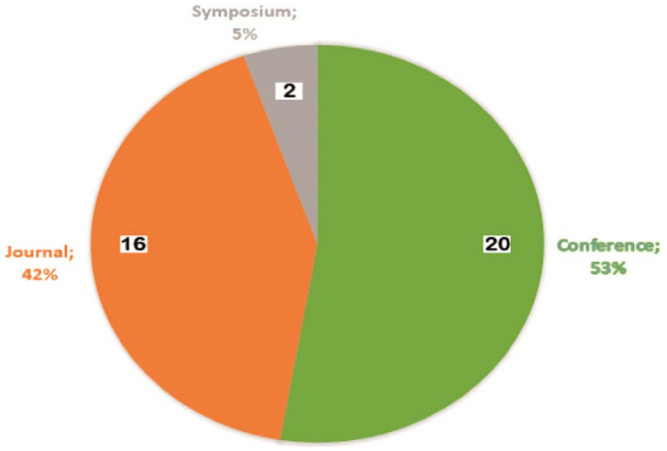

Fig. 8. Studies venue types.

have not been comprehensively and completely addressed. In our literature review, we found that there is not a perfect technique for improving the entire load balancing metrics. For example, some techniques considered response time, resource utilization, and migration time, while the others ignored these metrics and considered other metrics. However, it seems that some metrics are mutually exclusive. For example, relying on VM migration for load balancing may cause an increase in the response time. Service cost is another metric, which is not considered in the studied articles. Presenting a comprehensive technique to improve as many metrics as possible is, therefore, very desirable.

Furthermore, our study showed that the energy consumption and carbon emission are two important drawbacks due to the incremental growth of the number of datacenters. However, just a few articles addressed these two drawbacks. Energy consumption is regarded as an economic efficiency factor while carbon emission is regarded as a health-related, and/or an environmental factor. Each of these issues is critically important. Therefore, providing load balancing mechanisms in a cloud environment while also addressing these two problems is very desirable too.

Recently, a large volume of data is produced daily from social networks, medical records, e-commerce, e-shopping, e-pay, banking records, etc. This huge volume of data makes big data, and therefore needs near-perfect distribution for fast servicing. Our study showed that in recent years just a few articles addressed this topic. Further optimization of Hadoop MapReduce for processing big data in the future research, is quite promising.

Recently, in addition to the existing popular cloud providers such as Google, Microsoft, and Amazon, other cloud providers are growing too. In some situations, it is necessary for a cloud provider to send some workload to another cloud provider for processing for the purpose of load balancing. In other words, using resources of more than one cloud provider is a critical requirement for load balancing in the future. In this case, the cloud providers will face data lock-in problems. Our study shows that just a few articles have paid attention to these topics. Therefore, another interesting line for future research can be the investigation of data lock-in and cross-cloud servicing problems.

7. Conclusion and future works

Balancing of the workload among cloud nodes is one of the most important challenges that cloud environments are facing today. In this paper, we surveyed research literature in the load balancing area, which is the key aspect of cloud computing. We found in the literature, several metrics for load balancing techniques that should be considered in future load balancing mechanisms. Based on our observations, we have presented a new classification of load balancing techniques: (1) Hadoop MapReduce load balancing category, (2) natural phenomenon-based load balancing category, (3) agent-based load balancing category, and (4) general load balancing category. In each category, we studied some techniques and analyzed them in terms of some metrics and summarized the results in tables. Key ideas, main objectives, advantages, disadvantages, evaluation techniques, publication year were metrics that we considered for load balancing techniques. Recently, load balancing techniques are focusing on two critical metrics, That is, energy saving and reducing carbon dioxide emission. As future works, we suggest the followings: (1) Study and analyze more recent techniques in each of our proposed categories, (2) Evaluate each technique in a simulation toolkit and compare them based on new metrics.

References

Abdolhamid, M., Shafi'i, M., Bashir, M.B., 2014. Scheduling techniques in on-demand grid as a service cloud: a review". J. Theor. Appl. Inf. Technol. 63 (1), 10–19.
Abdullahi, M., Md, Asri Ngadi, Md.A., Abdulhamid, S.M., 2015. Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Gener. Comput. Syst. 56, 640–650.
Aditya, A., Chatterjee, U., Gobata, S., 2015. A comparative study of different static and dynamic load-balancing algorithm in cloud computing with special emphasis on time factor. Int. J. Curr. Eng. Technol. 3 (5).
Ahmad, F., Chakradhar,S.T., Raghunathan,A., Vijaykumar, T.N., 2012. Tarazu: optimizing mapreduce on het-erogeneous clusters. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 40(1), 61-74.
Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., 2015. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J. Netw. Comput. Appl. 52, 11–25.
Alakeel, A.M., 2010. A guide to dynamic load balancing in distributed computer systems. Int. J. Comput. Sci. Netw. Secur. 10 (6), 153–160.
Apostu, A., Puican, F., Ularu, G., George Suciu, G., Todoran, G., 2013. Study on advantages and disadvantages of cloud computing – the advantages of telemetry applications in the cloud. Recent Adv. Appl. Comput. Sci. Digit. Serv..
Babu, L.D.D., Krishna, P.V., 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13 (5), 2292–2303.
Bellavista, P., Cinque, M., Corradi, A., Foschini, L., Frattini, F., Molina, J.P., 2016. GAMESH: a grid architecture for scalable monitoring and enhanced dependable job scheduling. Future Gener. Comput. Syst..
Benifa, J.V.B., Dejey, 2017. Performance improvement of MapReduce for heterogeneous clusters based on efficient locality and Replica aware scheduling (ELRAS) strategy. Wirel. Personal. Commun., 1–25.
Bhatia, J., Patel, T., Trivedi, H., Majmudar, V., 2012. HTV Dynamic Load-balancing algorithm for Virtual Machine Instances in Cloud. International Symposium on Cloud and Services Computing, 15–20.
Bok, K., Hwang, J., Jongtae Lim, J., Kim, Y., Yoo, J., 2016. An efficient MapReduce scheduling scheme for processing large multimedia data. Multimed. Tools Appl., 1–24.
Cai, Z., Li, X., Ruizc, R., Lia, Q., 2017. A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds. J. Future Gener. Comput. Syst. 71, 57–72.
Chethana, R., Neelakantappa, B.B., Ramesh, B., 2016. Survey on adaptive task assignment in heterogeneous Hadoop cluster. IEAE Int. J. Eng. 1 (1).
Chien, N.K., Son, N.H., HD, 2016. Load-balancing algorithm Based on Estimating Finish Time of Services in Cloud Computing, International Conference on Advanced

Commutation Technology (ICACT), 228-233.
Cinque, M., Corradi, A., Luca Foschini,L., Frattini, F., Mol, J.P., 2016. Scalable Monitoring and Dependable Job Scheduling Support for Multi-domain Grid Infrastructures. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing.
Dagli, M.K., Mehta, B.B., 2014. Big data and Hadoop: a review. Int. J. Appl. Res. Eng. Sci. 2 (2), 192.
Daraghmi, E.Y., Yuan, S.M., 2015. A small world based overlay network for improving dynamic load-balancing. J. Syst. Softw. 107, 187–203.
Dasgupta, K., Mandalb, B., Duttac, P., Mondald, J.K., Dame, S., 2013. A Genetic Algorithm (GA) based Load-balancing strategy for Cloud Computing, International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA), 10, 340-347.
Destanoğlu, O., Sevilgen, F.E., 2008. Randomized Hydrodynamic Load Balancing Approach, IEEE International Conference on Parallel Processing, 1, 196-203.
Deye, M.M., Slimani, Y., sene, M., 2013. Load Balancing approach for QoS management of multi-instance applications in Clouds. Proceeding on International Conference on Cloud Computing and Big Data, 119–126.
Domanal, S.G., Reddy, G.R.M., 2015. Load Balancing in Cloud Environment using a Novel Hybrid Scheduling Algorithm. IEEE International Conference on Cloud Computing in Emerging Markets, 37-42.
Doulkeridis, C., Nørvåg, K., 2013. A survey of large-scale analytical query processing in MapReduce. VLDB J., 1–26.
Dsouza, M.B., 2015. A survey of Hadoop MapReduce scheduling algorithms. Int. J. Innov. Res. Comput. Commun. Eng. 3 (7).
Fadika, Z., Dede, E., Govidaraju, M., 2011. Benchmarking MapReduce Implementations for Application Usage Scenarios. In: 2011 IEEE/ACM Proceedings of the 12th International Conference on Grid Computing, 0, 90–97.
Farrag, A.A.S., Mahmoud, S.A., 2015. Intelligent Cloud Algorithms for Load Balancing problems: A Survey. IEEE In: Proceedings of the Seventh International Conference on Intelligent Computing and Information Systems (ICICIS 'J 5), 210-216.
Gautam, J.V., Prajapati, H.B., Dabhi, V.K., Chaudhary, S., 2015. A Survey on Job Scheduling Algorithms in Big Data Processing. IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT'15), 1-11.
Ghoneem, M., Kulkarni, L., 2016. An Adaptive MapReduce Scheduler for Scalable Heterogeneous Systems. Proceeding of the International Conference on Data Engineering and Communication Technology, 603–6011.
Ghosh, S., Banerjee, C., 2016. Priority Based Modified Throttled Algorithm in Cloud Computing. International Conference on Inventive Computation Technology.
Goyal, S., Verma, M.K., 2016. Load balancing techniques in cloud computing environment: a review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (4).
Gupta, H., Sahu, K., 2014. Honey bee behavior based load balancing of tasks in cloud computing. Int. J. Sci. Res. 3 (6).
Gutierrez-Garcia, J.O., Ramirez-Nafarrate, A., 2015. Agent-based load balancing in Cloud data centers. Clust. Comput. 18 (3), 1041–1062.
Hefny, H.A., Khafagy, M.H., Ahmed, M.W., 2014. Comparative study load balance algorithms for MapReduce environment. Int. Appl. Inf. Syst. 106 (18), 41.
Hou, X., Kumar, A., Varadharajan, V., 2014. Dynamic Workload Balancing for Hadoop MapReduce. Proceeding of International Conference on Big data and Cloud Computing, 56-62.
Hsueh, S.C., Lin, M.Y., Chiu, Y.C., 2014. A load-balanced MapReduce algorithm for blocking-based entity-resolution with multiple keys. Parallel Distrib. Comput. (AusPDC), 3.
Hwang, K., Dongarra, J., Fox, G.C., 2013. Distributed and Cloud Computing: from Parallel Processing to the Internet of Things.
Ivanisenko, I.N., Radivilova, T.A., 2015. Survey of Major Load-balancing algorithms in Distributed System. Information Technologies in Innovation Business Conference (ITIB).
Jadeja, Y., Modi, K., 2012. Cloud Computing - Concepts, Architecture and Challenges. International Conference on Computing, Electronics and Electrical Technologies [ICCEET].
Jaikar, A., Dada, H., Kim, G.R., Noh, S.Y., 2014. Priority-based Virtual Machine Load Balancing in a Scientific Federated Cloud. IEEE In: Proceedings of the 3rd International Conference on Cloud Computing.
Kabir, M.S., Kabir, K.M., Islam, R., 2015. Process of load balancing in cloud computing using genetic algorithm. Electr. Comput. Eng.: Int. J. 4 (2).
Kanakala, V.R.T., Reddy, V.K., 2015a. Performance analysis of load balancing techniques in cloud computing environment. TELKOMNIKA Indones. J. Electr. Eng. 13 (3), 568–573.
Kanakala, V.R.T., Reddy, V.K., 2015b. Performance analysis of load balancing techniques in cloud computing environment. TELKOMNIKA Indones. J. Electr. Eng. 13 (3), 568–573.
Kansal, N.J., Inderveer Chana, I., 2012. Cloud load balancing techniques: a step towards green computing. Int. J. Comput. Sci. Issues 9 (1), 238–246.
Kaur, R., Luthra, P., 2014. Load Balancing in Cloud Computing, International Conference on Recent Trends in Information. Telecommunication and Computing, ITC, pp. 1–8.
Kc, K., Anyanwu, K., 2010. Scheduling Hadoop Jobs to Meet Deadlines. In: Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 388–392.
Keshvadi, S., Faghih, B., 2016. A multi-agent based load balancing system in IaaS cloud environment. Int. Robot. Autom. J. 1 (1).
Khalil, S., Salem, S.A., Nassar, S., Saad, E.M., 2013. Mapreduce performance in heterogeneous environments: a review. Int. J. Sci. Eng. Res. 4 (4), 410–416.
Khiyaita, A., Zbakh, M., Bakkali, H.E.I., Kettani, D.E.I., 2012. Load balancing cloud computing: state of art. Netw. Secur. Syst. (JNS2), 106–109.
Kianpisheh, S., Charkari, N.M., Kargahi, M., 2016. Ant colony based constrained workflow scheduling for heterogeneous computing systems. Clust. Comput. 19, 1053–1070.
Kliazovich, D., Pecero, J.E., Tchernykh, A., Bouvry, P., Khan, S.U., Zomaya, A.Y., 2016. CA-DAG: modeling communication-aware applications for scheduling in cloud computing. J. Grid Comput., 1–17.
Kolb, L., Thor, A., Rahm, E., 2011. Block-based Load Balancing for Entity Resolution with MapReduce. International Conference on Information and Knowledge Management (CIKM), 2397–2400.
Kolb, L., Thor, A., Rahm, E., 2012. Load Balancing for MapReduce-based Entity Resolution, IEEE In: Proceedings of the 28th International Conference on Data Engineering, 618-629.
Komarasamy, D., Muthuswamy, V., 2016. A novel approach for dynamic load balancing with effective Bin packing and VM reconfiguration in cloud. Indian J. Sci. Technol. 9 (11), 1–6.
Koomey, J.G., 2008. Worldwide electricity used in datacenters. Environ. Res. Lett. 3 (3), 034008.
Kulkarni, A.K., B, A, 2015. Load-balancing strategy for Optimal Peak Hour Performance in Cloud Datacenters. In: Proceedings of theIEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES).
Kumar, S., Rana, D.H., 2015. Various dynamic load-balancing algorithms in cloud environment: a survey. Int. J. Comput. Appl. 129 (6).
Lee, K.H., Choi, H., Moon, B., 2011. Parallel data processing with MapReduce: a survey. SIGMOD Rec. 40 (4), 11–20.
Li, R., Hu, H., Li, H., Wu, Y., Yang, J., 2015. MapReduce parallel programming model: a state-of-the-art survey. Int. J. Parallel Program., 1–35.
Lin, C.Y., Lin, Y.C., 2015. A Load-Balancing Algorithm for Hadoop Distributed File System, International Conference on Network-Based Information Systems.
Lua, Y., Xie, Q., Klito, G., Geller, A., Larus, J.R., Greenberg, A., 2011. Join-Idle-Queue: a novel load-balancing algorithm for dynamically scalable web services. Int. J. Perform. Eval. 68, 1056–1071.
Malladi, R.R., 2015. An approach to load balancing In cloud computing. Int. J. Innov. Res. Sci. Eng. Technol. 4 (5), 3769–3777.
Manjaly, J.S., A, CE, 2013. Relative study on task schedulers in Hadoop MapReduce. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3 (5).
Mesbahi, M., Rahmani, A.M., 2016. Load balancing in cloud computing: a state of the art survey. Int. J. Mod. Educ. Comput. Sci. 8 (3), 64.
Milani, A.S., Navimipour, N.J., 2016. Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends. J. Netw. Comput. Appl. 71, 86–98.
Mishra, N.K., Misha, N., 2015. Load balancing techniques: need, objectives and major challenges in cloud computing: a systematic review. Int. J. Comput. 131 (18).
Moschakisa, I.A., Karatzaa, H.D., 2015. Multi-criteria scheduling of Bag-of-Tasks applications on heterogeneous interlinked clouds with simulated annealing. J. Softw. Syst. 101, 1–14.
Mukhopadhyay, R., Ghosh, D. , Mukherjee, N., 2010. A Study on the application of existing load-balancing algorithms for large, dynamic, and heterogeneous distributed systems ACM, A Study on the Application of Existing Load-balancing algorithms for Large, Dynamic, and Heterogeneous Distributed System. In Proceedings of 9th International Conference on Software Engineering, Parallel and Distributed Systems, 238–243 .
Neeraj, R., Chana, I., 2014. Load balancing and job migration techniques in grid: a survey of recent trends. Wirel. Personal. Commun. 79 (3), 2089–2125.
Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K.P., Nitin, N., Rastogi,R., 2012. Load Balancing of Nodes in Cloud Using Ant Colony Optimization. In: Proceedings of the 14th International Conference on Modelling and Simulation, 3-8.
Nuaimi, K., Mohamed, N., Mariam Al-Nuaimi, M., Al-Jaroodi, J., 2012. A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms, IEEE In: Proceedings of the Second Symposium on Network Cloud Computing and Applications.
Palta, R., Jeet, R., 2014. Load balancing in the cloud computing using virtual machine migration: a review. Int. J. Appl. Innov. Eng. Manag. 3 (5), 437–441.
Patel, H.M., 2015. A comparative analysis of MapReduce scheduling algorithms for Hadoop. Int. J. Innov. Emerg. Res. Eng. 2 (2).
Polato, I., Re, R., Goldman, A., Kon, F., 2014. A comprehensive view of Hadoop research – a systematic literature review. J. Netw. Comput. Appl. 46, 1–25.
Rajabioun, R., 2011. Cuckoo optimization algorithm. Appl. Soft Comput. 11, 5508–5518.
Randles, M., Lamb, D., Tareb-Bendia, A., 2010. A Comparative Study into Distributed Load-balancing algorithms for Cloud Computing, IEEE In: Proceedings of the 24th International Conference on Advanced Information Networking and Applications Workshops, pp. 551–556.
Rao, B.T., Reddy, L.S.S., 2011. Survey on improved scheduling in Hadoop MapReduce in cloud environments. Int. J. Comput. Appl. 34 (9).
Rastogi, G., Sushil, R., 2015. Analytical Literature Survey on Existing Load Balancing Schemes in Cloud Computing. International Conference on Green Computing and Internet of Things (ICGCloT).
Rathore, N., Channa, I., 2011. A Cognitive Analysis of Load Balancing and job migration Technique in Grid World Congress on Information and Communication Technologies Congr. Inf. Commun. Technol. (WICT). pp. 77–82.
Rathore, N., Chana, I., 2013. A Sender Initiate Based Hierarchical Load Balancing Technique for Grid Using Variable Threshold Value. Signal Processing, Computing and Control (ISPCC), IEEE International Conference.
Ray, S., Sarkar, A.D., 2012. Execution analysis of load-balancing algorithms in cloud computing environment. Int. J. Cloud Comput.: Serv. Archit. (IJCCSA) 2 (5).
Sarood, O., Gupta, A., Kale, L.V., 2012. Cloud Friendly Load Balancing for HPC Applications: Preliminary Work. International Conference on Parallel Processing Workshops, 200–205.

Scharf, M., Stein, M., Voith,T., Hilt, V., 2015. Network-aware Instance Scheduling in OpenStack. International Conference on Computer Communication and Network (ICCCN), 1-6.
Selvi, R.T., Aruna, R., 2016. Longest approximate time to end scheduling algorithm in Hadoop environment. Int. J. Adv. Res. Manag. Archit. Technol. Eng. 2 (6).
Shadkam, E., Bijari, M., 2014. Evaluation the efficiency of cuckoo optimization algorithm. Int. J. Comput. Sci. Appl. 4 (2), 39–47.
Shaikh, B., Shinde, K., Borde, S., 2017. Challenges of big data processing and scheduling of processes using various Hadoop Schedulers: a survey. Int. Multifaceted Multiling. Stud. 3, 12.
Shen, H., Sarker, A., Yuy, L., Feng Deng, F., 2016. Probabilistic Network-Aware Task Placement for MapReduce Scheduling. In: Proceedings of the IEEE International Conference on Cluster Computing.
Shen, H., Yu, L., Chen,L., Li, Z., 2016. Goodbye to Fixed Bandwidth Reservation: Job Scheduling with Elastic Bandwidth Reservation in Clouds. In: Proceedings of the International Conference on Cloud Computing Technology and Science.
Sidhu, A.K., Kinger, S., 2013. Analysis of load balancing techniques in cloud computing. Int. J. Comput. Technol. 4 (2).
Sim, K.M., 2011. Agent-based cloud computing. IEEE Trans. Serv. Comput. 5 (4), 564–577.
Singh, P., Baaga, P., Gupta, S., 2016. Assorted load-balancing algorithms in cloud computing: a survey". Int. J. Comput. Appl. 143 (7).
Singha, A., Juneja, D., Malhotra, M., 2015. Autonomous Agent Based Load-balancing algorithm in Cloud Computing. International Conference on Advanced Computing Technologies and Applications (ICACTA), 45, 832–841.
Sui, Z., Pallickara, S., 2011. A survey of load balancing techniques forData intensive computing. In. In: Furht, Borko, Escalante, Armando (Eds.), Handbook of Data Intensive Computing. Springer, New York, 157–168.
Tasquier, L., 2015. Agent based load-balancer for multi-cloud environments. Columbia Int. Publ. J. Cloud Comput. Res. 1 (1), 35–49.
Vaidya, M., 2012. Parallel processing of cluster by Map Reduce. Int. J. Distrib. Parallel Syst. 3 (1).
Valvåg, S.V., 2011. Cogset: A High-Performance MapReduce Engine. Faculty of Science and Technology Department of Computer Science, University of Tromsö, 14.
Valvåg, S.V., Johansen, D., 2009. Cogset: A unified engine for reliable storage and parallel processing, In: Proceedings of the Sixth IFIP International Conference on Network and Parallel Computing, 174–181.
Vasic, N., Barisits, M., 2009. Salzgeber, V. Making Cluster Applications Energy-Aware, In ACDC '09 In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, ACM, New York, NY, USA, pp. 37–42.
Vernica, R., Balmin, A., Beyer, K.S., Ercegovac, V., 2012. Adaptive MapReduce using situation-aware mappers. International Conference on Extending Database Technology (EDBT), 420–431.
Wei, X., Fan, J., Lu, Z., Ding, K., 2013,. Application scheduling in mobile cloud computing with load balancing. J. Appl. Math., 1–13.
Wei, X., Fan, J., Wang, T., Wang, Q., 2015. Efficient application scheduling in mobile cloud computing based on MAX–MIN ant system. Soft Comput., 1–15.
Xia, Y., Wang, L., Zhao, Q., Zhang, G., 2011. Research on job scheduling algorithm in Hadoop. J. Comput. Inf. Syst. 7, 5769–5775.
Yahaya, B., Latip, R., Othman, M., Abdullah, A., 2011. Dynamic load balancing policy with communication and computation elements in grid computing with multi-agent system integration. Int. J. New Comput. Archit. Appl. (IJNCAA) 1 (3), 757–765.
Yakhchi, M., Ghafari, S.M., Yakhchi, S., Fazeliy, M., Patooghi, A., 2015. Proposing a Load Balancing Method Based on Cuckoo Optimization Algorithm for Energy Management in Cloud Computing Infrastructures. Published In: Proceedings of the 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO).
Yang, S.J., Chen, Y.R., 2015. Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds. J. Netw. Comput. Appl. 57, 61–70.
Zaharia, M., 2009. Job Scheduling with the Fair and Capacity Schedulers 9. Berkley University.
Zaharia, M., Borthakur, D., Sarma, J.S., 2010. Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling, in Proceedings of the European conference on Computer systems (EuroSys'10), 265–278.
Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I., 2008. Improving MapReduce Performance in Heterogeneous Environments. In: Proceedings of the 8th conference on Symposium on Opearting Systems Design and Implementation, 29–42.
Zhang, Y., Li, Y., 2015. An improved Adaptive workflow scheduling Algorithm in cloud environments. In: Proceedings of the Third International Conference on Advanced Cloud and Big Data, 112-116.