Journal of Theoretical and Applied Information Technology
31st January 2022. Vol.100. No 2
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZED MIN-MIN TASK SCHEDULING ALGORITHM FOR SCIENTIFIC WORKFLOWS IN A CLOUD ENVIRONMENT

SALLAR SALAM MURAD[1], ROZIN BADEEL [2]*, NASHAT SALIH ABDULKARIM ALSANDI[3], RAFI FARAJ ALSHAAYA[2] REHAM A. AHMED[2], ABDULLAH MUHAMMED[2], MOHD DERAHMAN[2].

[1]Department of Computing, University Tenaga Nasional, Putrajaya, Malaysia
[2]Department of Communication Technology and Network, University Putra Malaysia.
[3] Department of Information Technology, Duhok Private Technical Institute, Duhok Kurdistan Region, Iraq

E-mail: sallarmurad@gmail.com , rozinbabdal1987@gmail.com, Nashat.alsandi@dhk-pti.com ,rafi.shaaya@gmail.com , rehamabdul.ahmed@gmail.com , abdullah@upm.edu.my, mnoord@upm.edu.my

Corresponding authors: gs53825@student.upm.edu.my or rozinbabdal1987@gmail.com.

ABSTRACT

Resource allocation and cloudlets scheduling are fundamental problems in a cloud computing environment. The scheduled cloudlets must be executed efficiently by using the available resources to improve system performance. To achieve this, we propose a new noble mechanism called Optimized Min-Min (OMin-Min) algorithm, inspired by the Min-Min algorithm. The objectives of this work are: i) to provide a comprehensive review of the cloud and scheduling process; ii) to classify the scheduling strategies and scientific workflows; iii) to implement our proposed algorithm with various scheduling algorithms (i.e., Min-Min, Round-Robin, Max-Min, and Modified Max-Min) for performance comparison, within different cloudlet sizes (i.e., small, medium, large, and heavy) in three scientific workflows (i.e., Montage, Epigenomics, and SIPHT); and iv) to investigate the performance of the implemented algorithms by using CloudSim. The main goal of this study is to obtain optimum results that satisfy the minimum completion time and achieve better utilization of resources, which lead to increased throughput. The algorithms were implemented in a Java environment. Results were discussed and analyzed by using formulas and were compared in percentages. According to the simulation results, the proposed algorithm produces the best solution among all algorithms in the proposed cases.

Keywords: *Cloud computing, Scheduling, Workflow scheduling, Scientific workflow, DAG*

1. INTRODUCTION

Cloud computing technology and cloud services are growing technologies that permits users to pay as much as their need. It includes hosted applications that deal with essential factors such as storage and processing. [1]. Cloud is measured as a web-built service offered by several network providers-based consumption of quality of service (QoS), load balancing (LB) and others [1]. An efficient algorithm is therefore required to allocate the correct task to the right resources. Without affecting cloud service parameters, the available resources should be used effectively, particularly if the cloud has a limited resource setting in order to efficiently and error-freely operate which most cloud systems on their need, and the cloud has to manage several factors in cloud planning that directly affects user Should run, leading to cloud service providers gaining more reputation when they compete, for example, with storage available for input and output operations in the cloud without causing any delay. The cloud programming process can be classified into three stages, which are i) the discovery and filtering of resources, ii) the selection of resources, and iii) the submission of tasks [1]. The data centre broker for resource discovery identifies and gathers the resources available in the network structure. The targeted resources are selected based on precise tasks and parameters related to resources throughout the resource selection process. The tasks are submitted to the selected resources during the submission phase, shown below in the research method through simulation process, section III.

Journal of Theoretical and Applied Information Technology
31st January 2022. Vol.100. No 2
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Tasks are chosen by priority and assigned to processors that perform a predefined objective function. There are four main types of cloud available [2][3]: i) public, ii) private, iii) hybrid and iv) community cloud. The public cloud refers to a pay-as-you-go way of having a cloud [4]. The public cloud has some primary advantages for cloud-service providers, and no initial network

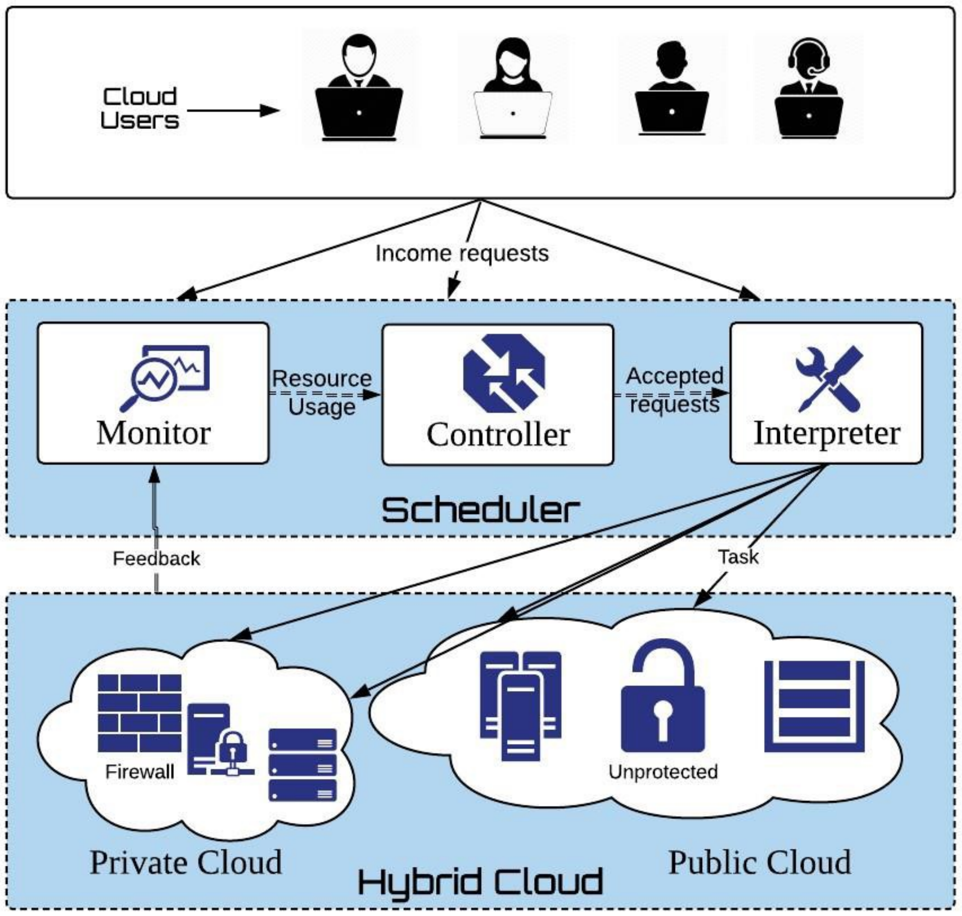


Figure 1. Cloud Types.

Capital costs or risk transfers to infrastructure providers. Public clouds, however, are not thoroughly managed over: i) data, ii) network, and iii) security settings, which many business situations hinder their efficiency [5]. In this case, it is hard to gather all VMs or host data. The private cloud concept refers to a business or other organization's internal data centre inaccessible to the all-purpose public. Community cloud, a variety of organizations with related objectives, have a shared cloud network (e.g., Assignments, security requirements, legislation, and regulation). This is generally managed on the spot or off-site by community-based or third-party organizations [3]. Communal clouds have advantages because they are private and protected networks rather than public clouds for specific organizations. The downside is that the cloud is not secure and has to be handled. However, public cloud benefits from lower investment in investment capital and a better deployment pace, a survey by IDG in [6] found that private clouds are much more common among businesses. A hybrid cloud system example is shown in Figure (1). Within a system like this, a scheduler determines the resource practice of public and private clouds together if the incoming cloud users require it. Upon approval of user

requests, the scheduler interprets these requests into tasks and then moves them to public or private clouds. The intracloud network, linking a client's instances and a cloud-based pooled infrastructure, is a suitable example of a private cloud. Within a cloud, the network of intra-data centres, as contrasted with the inter-datacenter network, is always entirely different [7]. A hybrid cloud incorporates private and public cloud models, which aim to overcome each approach's limitations. Part of the facility infrastructure in hybrid cloud turns in private clouds, whereas the rest runs on public clouds. Hybrid clouds have more excellent stability than private and public clouds. In particular, they deliver fitted to control and security over application data compared to public clouds, facilitating the growth and reduction of on-demand services. On the contrary, cross-cloud architecture involves the most acceptable breakdown between public and private cloud elements to be identified carefully [8]. It remains known minimizing transfer and processing time stays vital to the system when preparing any intensive data or computing an application (refer to Figure.8). Because of the reports and research that have been cone in the cloud domain, the cloud environment does not have many standard task planning algorithms. In [9], Pinedo's description of the scheduling problem is as follows: "It is a decision-making process that is used regularly in many manufacturing and services industries, It deals with the allocation of resources to tasks over given periods, and its goal is to optimize one or more objectives. Cloud scheduling means choosing the most suitable tool or allocating machines toward all tasks to decrease the completion time makespan. In general, each task is designed by prioritizing the list. It is required to seek the best algorithm to maximize productivity by seeking the least makespan for using resources in cloud computing systems where there are potentially vast numbers of users from different contexts and technologies and with various requirements and demands on thousands or millions of virtual machines waiting for their turn to be served. It is essential for cloud service providers to ensure that any available resource is completely exploited to prevent resources from being idle, thus incurring unnecessary costs [10][11]. Therefore, the resources can be sluggish at a certain point in the system. This provides an opportunity for the dynamic scheduler to use information that is changed during jobs execution, thereby updating the system. To do so, the dynamic scheduler sends a set of jobs in execution and monitors the system behavior, collecting information to schedule the next stage of jobs. This information collection cannot be accomplished unless the resource is idle [12]. Furthermore, there are algorithms that take advantage of idle time. One of those algorithms is the Heterogeneous Earliest Finish Time (HEFT); it uses a strategy that considers using the idle time between two tasks in the processor to schedule a new task. This will minimize the overall task completion time if the insertion strategy is feasible. In Addition, HEFT and Min–Min are efficient algorithms that belong to static scheduling processes, which assign resources corresponding to each job before the function begins; both of these algorithms mainly concentrate on the overall completion period as their optimization target. Nonetheless, these algorithms neglect the effect of expense. As a rule, the makespan of HEFT and Min–Min are small in general[13][14].

Due to the number of issues, including the lack of standards, different and often changing APIs, it is challenging to overcome limitations or enhance cloud computing system performance. The changeable load or daily load is a problem for algorithms and methods to improve the system performance. Some works have therefore adopted historical data to forecast the potential burden of overcoming this [6], [14], [15]. As loads are comparatively predicted and controlled compared to the public cloud, heuristic algorithms [16], for instance, Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), can be used. The fundamental issues with the cloud computing system are resource distribution and cloud planning [13]. There are also other examples of the general problem in the literature [17].

The problem is NP-Complete, even though polynomial-time solutions are available for a few scenarios [9]. In recent years, traditional optimum task planning has been well investigated and have proved to be a non-deterministic time-complete problem for polynomial (NP). To solve the problem, many heuristics and metaheuristic algorithms have been proposed [13]. This is commonly referred to as a problem with workflow scheduling. Overall, an NP-complete problem has been studied extensively in other paradigms [16], including grid and cluster computing, the workflow scheduling problem being to reduce the linear combination of the makespan [18].

In the previous work [19], the authors introduced Modified MMax-Min, and they produce good results in terms of (Makespan) than Max-Min and Round Robin, but the main problem with the MMax-Min is the precedence between the use of cloud and resources. Precedence is the condition of being considered more important than something else, priority in importance, order, or rank.

In this work, we introduce the OMin-Min algorithm inspired by the traditional Min-Min algorithm. We have used three scientific workflows SIPHT, MONTAGE, AND EPIGONMICS, with different sizes for each workflow. We apply different algorithms Max-Max, Round-Robin, MMax-Min [19], traditional Min-Min and finally, our proposed algorithm OMin-Min to achieve less makespan for each task in different workflow types and different workflow sizes and to compare the performance for each algorithm in different scientific workflows to find the best performance among these algorithms the proposed OMin-Min algorithm is compared and evaluated with other existing algorithms in the cloud computing environment, the performance of these algorithms in the cloud have been experimented and measured.

This study aims to develop a new modified version of the existing algorithm (MIN-MIN), that aims to provide better performance for task scheduling in a cloud computing environment.

The current methods and techniques designed for task scheduling are mostly based on primary and existing algorithms. However, the versions of Min-Min algorithms are limited, and not all of them were dedicated to task scheduling using scientific workflows, in addition, not all of them were used in the cloud environment. Moreover, the new techniques combine more than one algorithm (hybrid), which means higher complexity in the system and higher latency. This motivated us to present the following contributions:

Our main contributions of this work can be summarized as follows:

- Cloud cloudlets scheduling and resource allocation in cloud computing are being studied and introduced.
- To propose an efficient heuristic algorithm called OMin-Min.
- This work aims to implement, investigate, and evaluate the proposed OMin-Min Algorithm against other algorithms executed within three workflows under specific parameters, the main goal of proposed method is to deliver minimal makespan. The work focuses on the performance in the cloud computing environment and how efficient it is regarding task scheduling.
- Several workflows are used, such as Montage, Sipht and Epigenomics, to evaluate the proposed algorithm with diverse node sizes for more accuracy.
- Extensive simulations were done to make a good comparison of results.

The remaining of this paper will be distributed as follows: Section 2 is related work, in section 3, the research methodology has been detailed, the methodology follows the standard procedure for analyzing the makespan and how we collect the data, and the algorithms have been used in this research. In section 4, the implementation of the proposed algorithms in the scientific workflow and the generated results are discussed and presented. In section 5, the conclusion of this research and the future work.

2. LITERATURE REVIEW

In this literature, several works have been completed to solve the problem of workflow considering the optimization purposes of makespan. According to the number of users increasingly growing, virtualization is one of the critical strategies to increase the usage of physical resources in cloud environments [22]. These virtualization strategies involve network virtualization and enable users to build multiple virtual machines (VMs) on a single physical server. Specifically, VMs can be created by raising or lowering the CPU power or the number of CPUs. Successful meta-job scheduling algorithms are required to accurately leverage the virtualized resources to execute the computing-intensive task [23]. However, with the adoption of virtualization techniques in computational cloud systems, new algorithms are supposed to deal with problems arising from the cloud infrastructure. The typical job scheduling problem is that the scheduling of meta-workers in applications through computation resources to minimize the jobs accomplished times while ignoring the actual mutual existence of the network resource [24]. Various strategies are static or dynamic scheduling. Through static scheduling algorithms, all user tasks are assigned that arrive simultaneously and have independent machine resources with their availability. The static algorithm involves basic techniques like First Come First Serve (FCFS). Round Robin methods (RR), the FCFS and RR, the static method, get all the user tasks scheduled in a single time point considering various parameters [25]. Different assets are used for determining the specific priority of jobs. Depending on this priority, the tasks are scheduled in the cloud, and the activities are categorized based on the various attributes perceived and assessed by different users. The most acceptable completion time is found by assigning tasks to machines, improving the performance [26].

Study [25] proposed to us with Ljfr-Sjfr nature's heuristics for scheduling tasks on computational grids this at initiating this approach allocate the big tasks on quickest resource than in next stage, and this method assigns the minor task to most temporary resource and the most considerable task to the most immediate resource alternately. The experimental results demonstrate this algorithm represent good napping instead of just the previous. But not very much progress was achieved in formulating an efficient, globally optimized. Another work in [27], introduced the Macolb algorithm to boost the cloud tasks scheduling for load balancing; the Macolb algorithm handles the device load when attempting to reduce the makespan of an assigned task set. However, between tasks are not preemptive, and they cannot be interrupted or transferred to another processor throughout their execution. However, in our study, we do not consider the hardware side

In [28], authors have proposed an automated resource scheduling algorithm focusses on reaching the maximum, or part maximum for cloud scheduling outperforms to increase the productivity level of computing resources. Yet, they did not consider additional system parameters, such as I / O speed, network conditions. Moreover, Study [29], introduced an improved Max-Min algorithm. The enhanced algorithm operates on several parameters instead of one parameter, as in the case of the conditional Max-Min procedure going to occur. However, they did not find the workload that could be sub-divided prior scheduling phase takes place. While Study [29] presented a new algorithm RASA, RASA can deliver schedules with a relatively lower makespan. But, the deadline within each task, arriving level of the assignments, cost of the task execution on each of the assets, cost of the interaction and several other cases that can be a subject of research are not considered. All of the above works [26] [30] [31] are associated with classical task scheduling algorithms like SJF (shortest job first), FCFS (first come, first serve), RR (round-robin) concerning Min-Min and Max-Min algorithms. On the other hands, an algorithm was introduced in [32], consists of two components. In the first part, the effectiveness is increased by a randomized algorithm. A scheduling list algorithm is being used to reduce the makespan; they introduced virtual task algorithms to improve makespan and acceptable efficiency in the complicated multi-cloud environment. The furthermost mutual objective encountered in the distributed systems literature is minimizing the completion time, or makespan, accomplished by appropriate allocation of tasks in the available resources.

2.1 Min-Min Algorithms Related Literature

Many researchers have conducted different perspectives and versions on Min-Min algorithm scheduling and resource allocation based on algorithm studying and examining the same approaches in the literature. The following Table summarizes previous Min-Min versions in the literature. On the other hand, many studies focused on schedule using the Min-Min algorithm in [33]. The study introduces a unique algorithm based on an initial Min-Min algorithm called "QoS directed min-min algorithm." At the point of scheduling of tasks, it needs higher throughput than other algorithms. Whereas, if the necessary throughput for various tasks varies extremely, the QoS guided min-min provides a better outcome than Min-Min. However, if the throughput demand of all tasks is about the same, the QoS directed min-min algorithm performs like the Min-Min algorithm.

Moreover, Study [30] implemented a load balancing algorithm for equal scheduling over Max-Min. This algorithm attempted to use the best solution and limit the execution time and expected price to implement all jobs. Additionally, they didn't compare these outcomes to other swarm intelligence algorithms to discover a better alternative. Another study [31] have discussed and developed a Min-Min traditional algorithm in which tasks have relied on the metrics like server loads and user priority. This approach classifies the users who assign the tasks to usual and significant users. The full loaded servers and minimum loaded servers with make duration are used to allocate charges depending on these users. This approach provides better results in resources usage ratio and offers better user satisfaction.

Many studies have been introduced modified or Enhanced Min-Min. First of all, [32] introduced Improved Min-Min, which improves the Min-Min algorithm based on QoS constraints, and high-priority tasks can be implemented first on the condition that high QoS tasks get high QoS resources in the Grid environment. Moreover, in [34], they introduced an algorithm that tries to use the advantages of this basic algorithm and avoids its drawbacks to reduce completion time and improve load balancing in a Grid environment. While anther studies have been done in a cloud environment, in [35], An Improved Min-Min Algorithm in Cloud Computing has been introduced. They compared the improved min-min algorithm with the traditional min-min algorithm to enhance resource utilization rate. Another study [36] presented an efficient rescheduling based task scheduling algorithm (improved Min-Min Algorithm (I Min-Min) to improve system performance and load balancing, and utilization of system resources. All

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the above studies [37][34] [35],[36] introduced or proposed improved or Enhanced Min-Min algorithms. Still, none of them used these algorithms with a scientific workflow such as Sipht or Montage, while [38] introduced Enhanced Min-min algorithm in cloud environment To achieve the best performance and minimize total completion time, reduce response time finally maximize resources utilization by using two different scientific workflows with different sizes Montage_25 Montage_50 Montage_100 Montage_1000 CyberShake_30 CyberShake_50 CyberShake_100 CyberShake_1000.

In [19], they introduced MMax-Min with different scientific workflow and com-pared their work performance with RR and Max-Min's other algorithms. The proposed algorithm can achieve better results than other algorithms.

3. SCHEDULING

3.1 Scheduling Process

It is vital to bear in mind the difference between jobs, tasks and cloudlets. A job might consist of one task or set of tasks [39], as shown in Figure (2). A job can have many identical or non-identical tasks. Whether a job completes successfully or not depends upon the status of its tasks; for example, if its tasks do not complete within the timeout threshold or fail for whatever reason, then the job will be marked as timed out or for-gotten. A job will only be marked as completed when all of its tasks have been completed successfully. On the other hand, a cloudlet is mentioned in section IV(B). The scheduling process is summarized as follows:

- Resource discovery and filtering – the broker of the data center will dis-cover the resources available in the network system. It will then gather the information of status related to them.
- Selection of resources – Desired Resources is chosen based on specific parameters and resource criteria.
- Submission of Task – the jobs are forwarded toward the designated re-source.

3.2 The Goal Of Scheduling Algorithms

The key goal is to schedule the jobs according to adaptable time, which involves determining the correct order in which jobs can be done under transaction logic restrictions. There are two primary algorithm groups. 1) Static algorithm and 2) dynamic algorithm. Both have their benefits and constraints. Dynamic algorithms are more ef-fective than static algorithms but have more overheads compared with others.

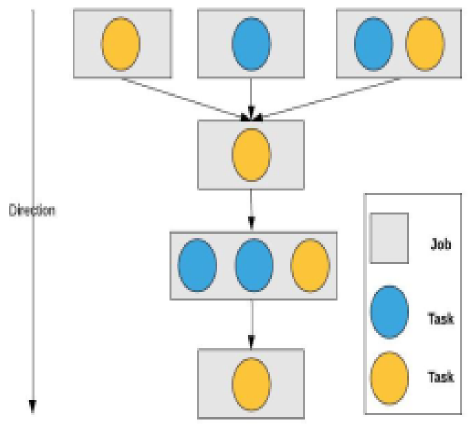


Figure 2. Jobs And Tasks In Workflow Scheduling

3.3 Workflow Scheduling Problem

The workflow applications are usually modelled as a directed acyclic graph (DAG). Graph denotes tasks and graph edges denotes the data dependencies among tasks with a load on the nodes that reflect the computational complexity, and load on the edges represents the communication volume [18]. The total running time is generally referred to as the time or makespan (the processing time). Many scheduling algorithms were proposed in recent decades, which are an NP-Hard issue to efficiently manage and execute business applications [18]. Appropriate heuristics, metaheuristics or approximation algorithms are necessary to find an almost optimal solution to the problem in polynomial time. Some standard workflow heuristics algorithms are heterogeneous Earliest-finish-time (HEFT), critical-path-on-a-processor (CPOP), [40] Min–Min [41], which seek to achieve execution performance level. So, the purpose of workflow scheduling techniques is to reduce the makespan of a parallel application by appropriate allocations of the tasks to the processors/resources and configuration of task execution patterns.

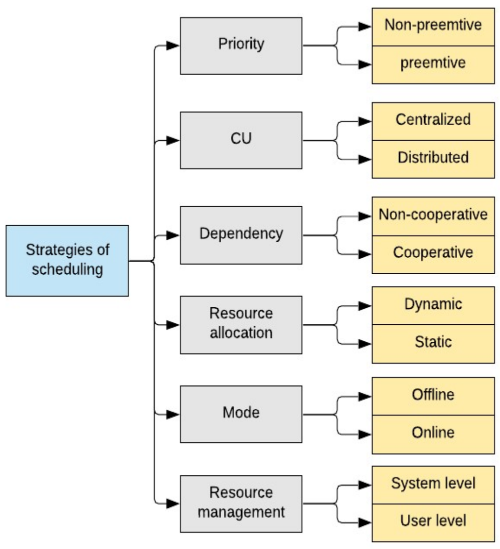


Figure 3. Different Strategies For Scheduling

3.4 Scheduling Strategies

According to [42][43][44][45][46][47][48][49] a classification of scheduling strategies was done and is shown in Figure (3).

3.5 Workflow Scheduling

Workflow implementations are typically conducted in a series since they are several separate tasks to achieve a specific result [50]. Task and the dependent relation of the tasks are shown respectively in the standard DAG models as node and edge. DAG is a finite, directed diagram, so nodes have topological sequences, and edges are sequences of execution. The collections of business activities and their effect on one another can be used for this reason. DAG may also be used as a consolidated representation of binary choice sequence data, such as the binary decision diagram. A DAG should formalize the accessibility relationship as a partial order with several applications while scheduling activities that function similarly to a workflow of business [51]. Since workflow tasks do have the functional sequence and partial order properties, in this context, we have set up a simple DAG model which represents the structure of various workflow tasks, as il-lustrated in Figure (4). The DAG of the application workflow contain eight tasks; thus, a particulate's respective dimension equals eight [51].

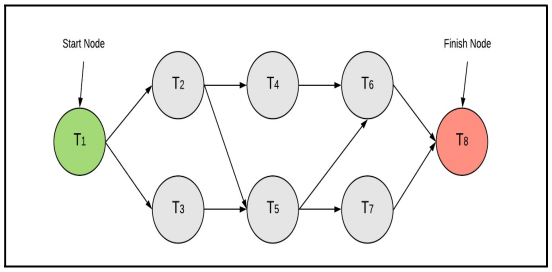


Figure 4. A Simple Dag Example Of Workflow Application.

Data transmission costs are estimated from two separate hardware with minimal operating costs for the relevant activities. The worst (w) value for finding the node weightiness is the worst rate. The edge value is calculated as data transmission costs from two various machines that are responsible for maximum execution costs of associated tasks. Simple best value (SB) takes account of the best performance and communication costs. The worst value (SW) of performance cost and communication costs [52].

3.6 Scientific Workflow Types

There are two types of workflows that are discussed [53][54]:

1.Basic workflow determines the output of any simple task in the formula of "DAG" as defined in Figure (8).

2.The scientific workflow contains a vast number of compound data in the formula of thousands of DAG tasks [61], which is also discussed shortly in Table (2), indicates the many real-life descriptions of workflows based on research applications.[10], [18], [23], [29], [51], they can be divided into two major categories: i) data-intensive and ii) compute intensive workflows. The communication-to-computation ratio CCR value is used to categorize these workflows. The CCR value is showing a very small, then it will be considered as compute-intensive workflows. Otherwise, it is data-intensive [23][55], refer to Figure (5). Researchers in [56] proposed to study the so-called PCP, mainly attentive to time, cost and energy efficiency, specifically on data-intensive workflows. They have developed an algorithm to design the workflow while fulfilling the QoS constraints. Study [57] has shown the advantages of the min-min and max-min by Ibarra and Kim [65] being computationally intensive algorithms.

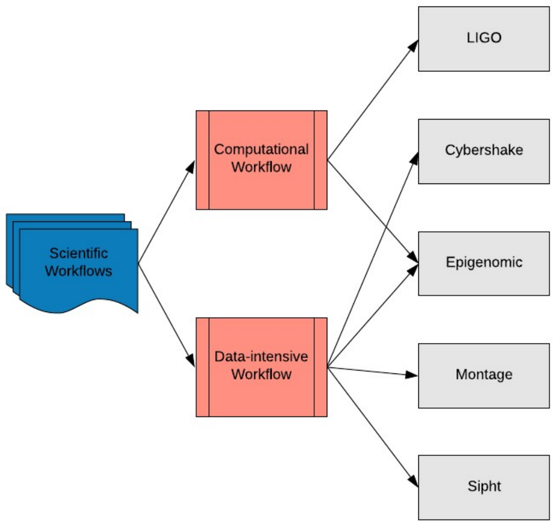


Figure 5. Types Of Scientific Workflows

3.7 NEED FOR IMPLEMENTING WORKFLOWS IN CLOUD

Application scalability is the main advantage of moving into Clouds. In comparison to grids, cloud scalability enables resources to meet the application requirements in re-al-time. In contrast with the traditional approach that requires advanced resource reservations in global multi-user Grid environments, workflow management systems can quickly meet application quality of service needs. Applications of workflow often re-quire very complex running environments. Such domains on grid resources are hard to construct. Furthermore, every grid site has a different setup, and each application must be ported to a new place with extra effort. The application developer can create an ap-plication environment that is fully customized, scalable and is designed for its application using Virtual machines (VM)[10], [18], [51].

3.5 Overview Of Scientific Workflows

3.5.1 Montage

The re-projected, revised images are then placed together into the final mosaic, and all images are modified to the same level [53]. The montage application was represented as a workflow in a grid environment like the TeraGrid [53][58]. The Nasa / Ipac infrared research archives (Montage: Astronomical Image Engine) was developed as an open-access toolkit for personalized mosaics in the sky with inbox-sized (fits) reference images [59][60]. The display's geometry is determined from the geometry of the input images during the creation of the final mosaic. The inputs will then be re-planted to the same spatial dimension and formation—the photos' background pollution. Montage is known as data-intensive workflow Figures (6).

3.5.2 Cybershake

The cybershake workflow is being used in the southern California earthquake Center in order to identify earthquake dangers in a region by the Probabilistic Seismic Risk Analysis technique (PSHA) [61]. A finite-difference simulation for the production of green strain tensors (SGTS) is carried out, given the region of focus. The Sgt data was used to measure synthetic seismograms for each of the expected ruptures. Phantom Acceleration and probabilistic hazard curves are generated once this is done[62]. The Pegasus workflow management system (Pegasus-Wms) on the tera- grid has been used to perform cybershake workflows, leading to over 800,000 jobs. data-intensive work-flow Figure (7) are recognized as Cybershake

3.5.3 Epigenomics

At present, the use centre Epigenomics is part of the age-wide mapping of the epigenetic state of human cells.

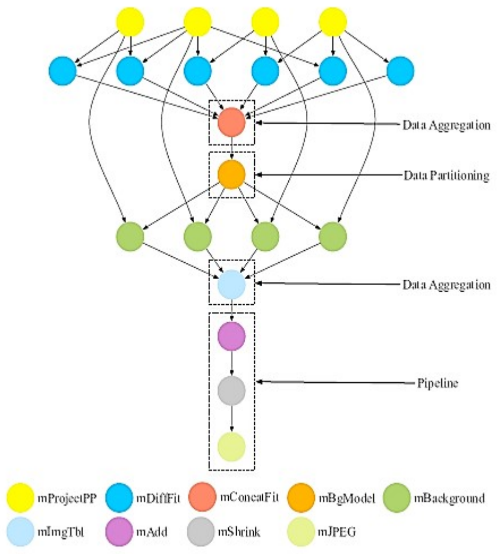


Figure 6. Montage Workflow [23].

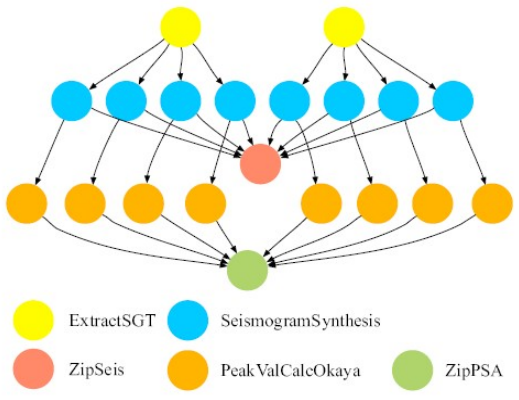


Figure 7. Cybershake Workflow [23].

The workflow of Epigenomics is a data processing pipeline, using the workflow framework of Pegasus to optimize the efficiency of various genome sequencing operations [63]. The genetic analyzer system is divided into several operational chunks that are parallel to the DNA sequence data produced by Illumi-na-soleXa ("Illumina."). The data in each piece is transformed to a file type that is usable by a Maq system (Maq: mapping and quality assembly). The remainder of these opera-tions consists of removing the noisy and contaminating sequences, moving sequences in a reference genome to the appropriate spot, producing a global map to classify the sequence concentration at each point in the genome. To manage the data on the development of DNA methylation and histone modification, this procedure is used by the Epigenomics Centre. Figure (8) categorize epigenomic workflows into categories i) da-ta-intensive and ii) compute-intensive workflow applications.

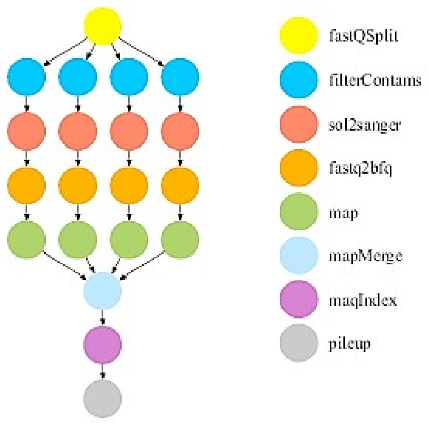


Figure 8. Epigenomics Workflow [23].

3.5.4 Sipht

An extensive search of small, translated RNAs (sRNAs) that regulate several processes such as secretion or virulence in bacteria occurs in the Harvard University bio-informatics project. In [63], The sRNAs ID Protocol (Sipht) uses a protocol to automatically scan for SRNA encoding genes in the National Biotechnology Information Center (NCBI) repository for all bacterial replicons. The Kingdom-wide forecasting and footnote of SRNA genes include using a range of programs with DAG (DAG man: Directed Acyclic Graph Manager) [64] capabilities to carry out the appropriate program-ming. These include predicting rho-independent transcriptional terminators, comparing the intergenetic regions with various replicas and footnotes of any sRNAs that are present [52] Figure (9), BLAST (Basic Local Alignment Search tools).

3.5.5 Inspiral (LIGO)

This workflow class applies gravitational physics for gravitational surfaces twisted by various events in the area. They require enormous data storage and computing time. To produce and interpret gravel shapes from the collected data during the coalescence of the compact binary system [65], the workflow of LIGO Inspiral Analysis is used.

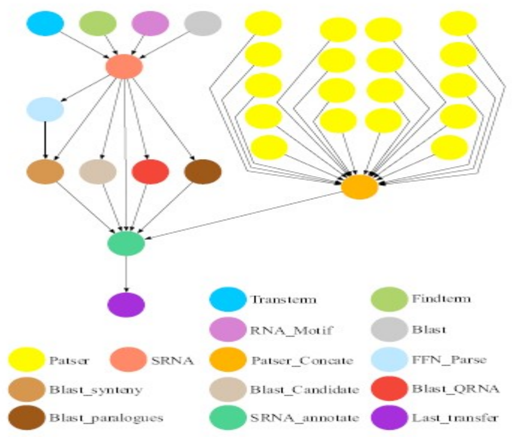


Figure 9. Sipht Workflow [23].

This workflow is thus classified as the compute-intensive workflow. In this study, this workflow is not used. All scientific workflows are shown, and Table (1) provides a brief description and application area [23].

3.6 Makespan

Standard resource utilization is used to estimate the algorithm's performance. The time lasting from submissions to the cloud is the makespan of workflow until the last function is complete [18], [55], [56][66].

3.7 Virtualization

Virtualization is a technology that differentiates physical hardware from computing functions and implementations. Work on early m44/44x was carried out in the 1960s by IBM TJ Watson and MIT [67]. Virtualization is also the backbone of cloud computing, allowing the integration of software and hardware among users, processes, and re-sources. The classical operating systems have not solved these separation problems well. The program should be used in the virtual environment with virtualization to run on the raw hardware. The virtualization happens depending upon the tier (layer) [67]. The virtualization techniques in cloud computing are used to build and control the cloud computing environment. Because this system is virtualized, it is easier to run and maintain. This virtualization architecture contains a data planning framework that executes the procedures on traffic that the mobile edge platform has provided, then route that traffic within applications, local and external networks. Virtualization technology allows consumers to pay-per-use for the installation, setup. The cloud service provider will combine services (e.g., CPUs, RAM, bandwidth, and disk) into VM's through virtualization technologies. CPU's processing power determines the execution time of tasks on VMs.

In contrast, the bandwidth and the amount of data transmitted in communication determine the time of transferring between various VM instances [18], [51]. It should be remembered that in this step, only one task can be waited on at a VM. This will only impact the waiting function of the same VM if the job runtime of a VM fluctuates. In addition, after the workflow schedule generates a completion signal, the input protocol updates the initial priorities and sub-dates of other activities within the workflow set. The feedback will also execute the modified tasks in the assignment process. It will also alleviate the propagated consequences of the fluctuating execution time. Each signal received will continue to activate each processing phase until all submitted workflows are completed. The assignment process of capital preserves assignments ready to be placed in a list with the target not falling. The next move is to arrange the tasks in the queue on VMs, where the resource management process is for the resource manager. The following section provides more details about the allocation of resources, including VMs, datacenters and hosts. The purpose of the simulation is to evaluate the performance of the min-min algorithm to benchmark it against the existing algorithms.

Table1. Description Of Scientific Workflows

Scientific workflow	DESCRIPTION	APPLICATION AREA
Montage	A workflow used to create an image mosaic of sky	Astronomy
Cybershake	A workflow used to depict the earthquake threats in an area with the help of the Probabilistic Seismic Hazard Analysis method	Earthquake
Epigenomics	A workflow interprets the processing of genetic data to implement the different genome sequencing operations	Genetic data
SIPHT	A workflow that implements the bioinformatics problems	Bioinformatics
LIGO	A workflow used for the identification of gravitational waves and to examine the data attained from compact binary systems	Gravitational works

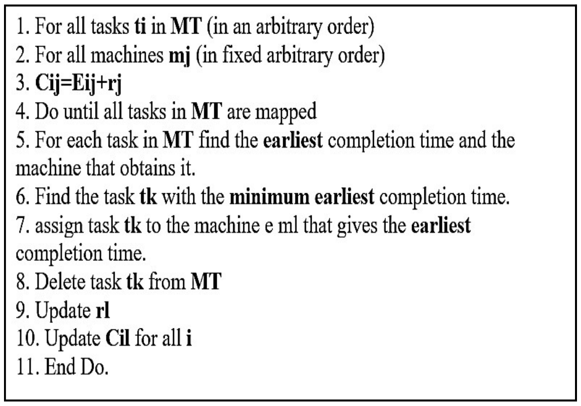
4. EXISTED SCHEDULING ALGORITHMS

4.1 Original Max-Min

The Max-Min algorithm [19] depends on "select a task with maximum execution time and allocate it to the resource with minimum execution time". The Max-Min algorithm priority is given to bigger tasks. Early implementation of the significant task will increase the total machine response time. The critical disadvantage of this algorithm is that this algorithm takes the most prominent tasks first, which might increase the response time of the system, and that issue was solved by improved Max-Min [50].

4.2 Min-Min

This algorithm functions in the opposite direction to Max-Min. Whereas first before considering smaller tasks, Max-Min algorithms select and allocate a resource with minimum completion time. Min-Min does the opposite by choosing and assigning small tasks in resources that can perform the task with minimum execution time. The objective of the Min-Min algorithm is to ensure, firstly, and then the same for Long Jobs [68], that all tasks with a minimum completion time are completed [7]. The Min-Min algorithm is displayed in Figure (10).



```
1. For all tasks ti in MT (in an arbitrary order)
2. For all machines mj (in fixed arbitrary order)
3. Cij=Eij+rj
4. Do until all tasks in MT are mapped
5. For each task in MT find the earliest completion time and the machine that obtains it.
6. Find the task tk with the minimum earliest completion time.
7. assign task tk to the machine e ml that gives the earliest completion time.
8. Delete task tk from MT
9. Update rl
10. Update Cil for all i
11. End Do.
```

FIGURE 10. Original Min-Min Algorithm

4.3 Round-Robin

Round Robin is one of the most widely used algorithms in allocating resources and processing tasks. In Round Robin, each job waiting in a queue to be scheduled by the scheduler is assigned a specific time slot. So, for more than the time allocated, no task is assigned to a resource and, if a task cannot complete the planning within the given period, another task will be prevented and sent back in the queue for other tasks.

4.4 Modified Max-Min (Mmax-Min)

The modified Max-Min algorithm is based on several parameters, including big cloudlets and small cloudlets. In contrast to the traditional Max-Min, it is a combination of Max-Min and Min-Min algorithms. In a precise manner, each cloudlet calculates completion times to identify the cloudlet with a minimum completion time and maximum completion time for all jobs submitted in the cloudlet list [19]. Table (2) shows a brief description of algorithms' advantages and disadvantages.

TABLE 2. ADVANTAGES AND DISADVANTAGES OF THE USED ALGORITHMS

Algorithm	ADVANTAGE	DISADVANTAGE
Max-Min	Cloud computing has been recognized as a preferred platform for data collection and information dissemination by the reliability of that algorithm.	The progress of a task with the highest completion time will improve the system's total turnaround time.
Min-Min	Make sure all tasks with minimum completion time are performed first in parallel, then for long tasks are performed in the same way [8][43]	It sometimes leaves the big tasks neglected.
Round Robin	A better response time and no delay for tasks of waiting for others tasks (no starvation).	No task will be allocated to a resource for more than the allocated task.
MMax-Min	Same as max-min, in addition, it achieved better quality solutions and better good values of Makespan in terms of performance analysis, experimentation and benchmarking [19]	Same as max-min plus, it has an issue for the priority.

4.5 Proposed Optimized Min-Min Scheduling Algorithms

There are two main stages when planning the execution of a workflow in a Cloud environment: i) the resource provisioning phase, ii) the scheduling generation phase. The computing resources that will be used to run the tasks are selected and provisioned during the first phase. In the second phase, a schedule is generated, and each task is mapped onto the best-suited resource. Virtual Machine (VM) performance is an additional challenge presented by Cloud platforms. VMs provided by current Cloud infra-structures do not exhibit a stable performance in terms of execution times. Study [69] reported an overall CPU performance variability of 24% on Amazon's EC2 Cloud. The shared nature of the infrastructure and virtualization and the heterogeneity of the un-derlying non-virtualized hardware are some reasons behind such variability. Many scheduling policies rely on estimating task runtimes on different VMs to make a map-ping decision, and this estimation is done based on the VMs computing capacity. If this capacity is always assumed to be optimal during the planning phase, the actual task execution will probably take longer. The task will be delayed, this delay will also impact the task's children, and the effect will continue to escalate until the workflow finishes executing [70].

In this work, we propose a new Min-Min algorithm called Optimized Min-Min OMin-Min for scheduling a scientific workflow application in a

Cloud environment. To achieve this, both resource provisioning and scheduling are considered optimization problems. Our contribution is, therefore, an algorithm with higher accuracy in terms of task execution time. Our work is based on the Min-Min algorithm, which has achieved a low execution time for tasks. Many problems in different areas have been successfully addressed by adapting Min-Min to many domains. For instance, this algorithm has been used and modified to evaluate and assist the performance against other algorithms.

4.6 Implementation

Within this section, the proposed Min-Min algorithm is explained in detail. It is accomplished by integrating CloudSim by inserting the workflow extension in a java environment. CloudSim is a Java-implemented event driven simulator. CloudSim's objective-oriented programming feature enables extensions and policy definitions in all software stack components, thus creating an appropriate research instrument that simulates the complexity of the environment [26]. This approach was suggested in the technique to accomplish the intended aim. The CloudSim toolkit supports the simulation of the system's components such as data centres, virtual machines (VMs) and re-source supply policies by using different algorithms.

```
1.While cloudlets in CloudletList
2. for all tasks ti in MT
3. for all virtual machines VMs, mj
4. Calculate completion time for all; CTij = ETij + rtj; (cloudlets and VMs);
5. Do until all tasks in MT are mapped;
6. Find cloudlets with minimum and maximum execution times MinClt and MaxClt; CTij;
7. for each task ti in MT
8. Select cloudlet ci in the ClouletList and compare it with MinClt;
9. If the cloudlet ci has minimum execution time, assign it to the resource that produces minimum
execution time;
10. Else Assign MaxClt, task tk to resource ml that produce minimum execution time;
11. End If
12. Delete cloudlet from the CloudletList; tk from MT;
13. Update ready times of selected virtual machines; rtmj;
14. Update completion time for all cloudlets; (CTij for all ti);
15. End While
```

FIGURE 11. Optimized Min-Min algorithm pseudo-code

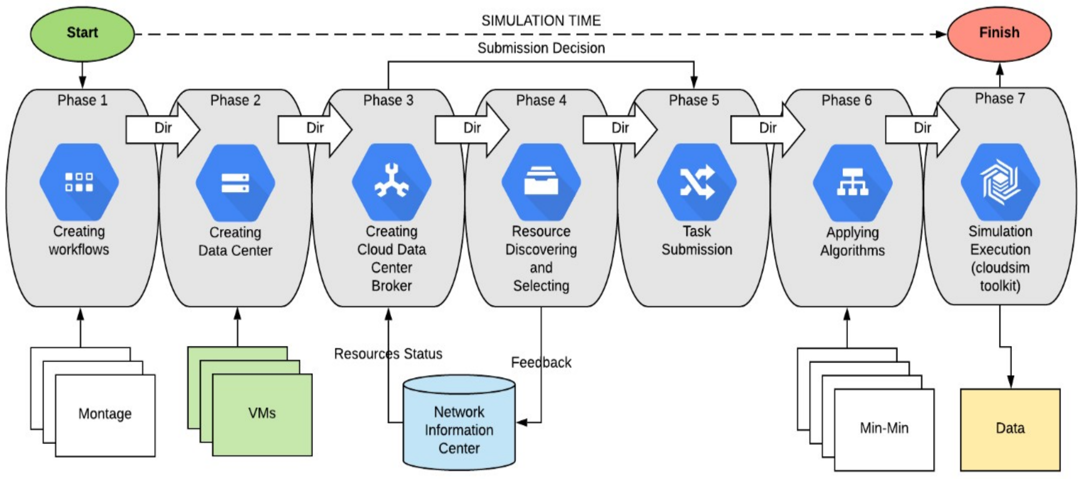


Figure 12. Research Method Through Simulation Process

The CloudSim toolkit supports both design and behaviour modelling. It incorporates standard application delivery technology that can currently be extended with simple and minimal effort. The Cloud System modelling and simulation are supported by single and inter-networked clouds (Domain federation). In addition, the Cloud Computing Scenarios have tailored frameworks for executing the rules and applying strategies for allocating VMs. For their work into the availability of cloud services and energy-efficient data centre infrastructure, many scientists use CloudSim. This saw a relatively rapid reduction in waiting for time (Min-Min), which led to fewer makespan as more cloudlets were scheduled [71]. Compared to the MMax-Min, which selects and assigns the cloudlet with maximum or minimal execution time, our approach can choose and allocate minimum time

for a resource first. In addition, we run each algorithm many times sequentially for better stability, and then we decided the minimum makespan to achieve better results. Figure (12) shows all phases of this research method and details as follows:

PHASE 1: CREATING WORKFLOW.

The initial creation of scientific workflows. This provides the basis and the dataset for simulation purposes.

PHASE 2: CREATING A DATA CENTRE.

A data centre would be built to run applications, process and store data (storage platform). The data centre is a pool of servers that contain a collection of software or applications which allow users to operate and access multiple virtual server instances over the Internet. The establishment of a data centre ensures cloud availability of resources.

PHASE 3: CREATING CLOUD DATA CENTRE BROKER.

The Cloud Datacenter Broker can detect resources, pick the resources that have been seen, send them to the network system, and gather information and updates for help.

PHASE 4: RESOURCE DISCOVERING AND SELECTING.

In resource discovery, the cloud datacenter broker who addresses it in a network system and collects status information is provided with a list of all available resources in the cloud. Information on available resources is collected, and then the best resource to comply with the application requirements for practical resources are selected during the selection of resources.

PHASE 5: TASK SUBMISSION.

The final step in preparing the task for a resource is task submission—the broker of the datacenter submission to the appropriate or unused resources for scheduling the picked tasks.

PHASE 6: APPLYING SCHEDULING ALGORITHMS.

The min-min is then used to schedule a task, and all the other algorithms are then implemented to measure the performance of the Min-Min algorithm to compare its and taken as a benchmark.

PHASE 7: SIMULATION AND COMPARISON OF THE RESULT.

A simulation of this work aims to test the OMax-Min algorithm's output to benchmark it against existing algorithms.

4.7 Design And Implementation Of Cloudsim.

Figure (13), Bw Provider: it is an abstract class that design the rule on bandwidth delivery for VMs, for CloudSim is shown as a class design diagram [72]. The critical function of this element is to allocate a collection of competing VMs to the network bandwidths in the data centre. Cloud technology developers and researchers will expand the class to represent the needs of their applications within their policy (priority, quality of service). The Bw Provisioning Simple enables VMs to store the necessary bandwidth, which is restricted by the host's overall bandwidth usage. Cloud Coordinator: This conceptual class expands the union into a cloud-based data centre. It constantly tracks and makes complex load shredding decisions on the internal status of datacenter services. Specific sensors and policies that should be followed during load shedding are part of the practical realization of this aspect. The Datacenter update technique is used to observe the datacenter resources by sending the Datacenter query Sensors. In the defined abstract system datacentre, infrastructure/network discovery is made to incorporate customized protocols and structures (multicast, Internet, peer-to-peer). The discovery of datacentres is carried out. In addition, the Amazon EC2 Load-Balancer can be extended to simulate a cloud-based service [72]. Developers who wish to implement apps throughout different clouds should enhance their inter-cloud policies to include this section.

Figure 13 shows the CloudSim design diagram.

4.8 Cloud Cloudlet

This shapes app services (such as the delivery of content, social networks and corporate workflow). In terms of their computation criteria, CloudSim organizes the complexity of an application. Each application service does have an overhead of the pre-assigned length of instruction and transmission of data (both pre-and post-fetches) which must be carried out throughout its lifecycle. This class could also be extended to cover other performance and composition metrics modelling for applications such as database-oriented transactions. Figure (13) shows an example of cloudlets. A Cloudlet is a small data centre on the edge of the network that uses virtualization to provide mobile or IoT cloud computing services, including smartphones, tablets, cameras, or controls. Cloudlets are designed to supply hosted services in the proximity of the devices re-quired, which avoids the need for a wide area network (WAN) connection to an application [73]. Cloudlets act as plugins to integrated cloud facilities that provide an intermediary platform for interactive and resource-intensive applications [74]. Any other sort of applications are likely to benefit from the shorter latency levels offered closer proximity.

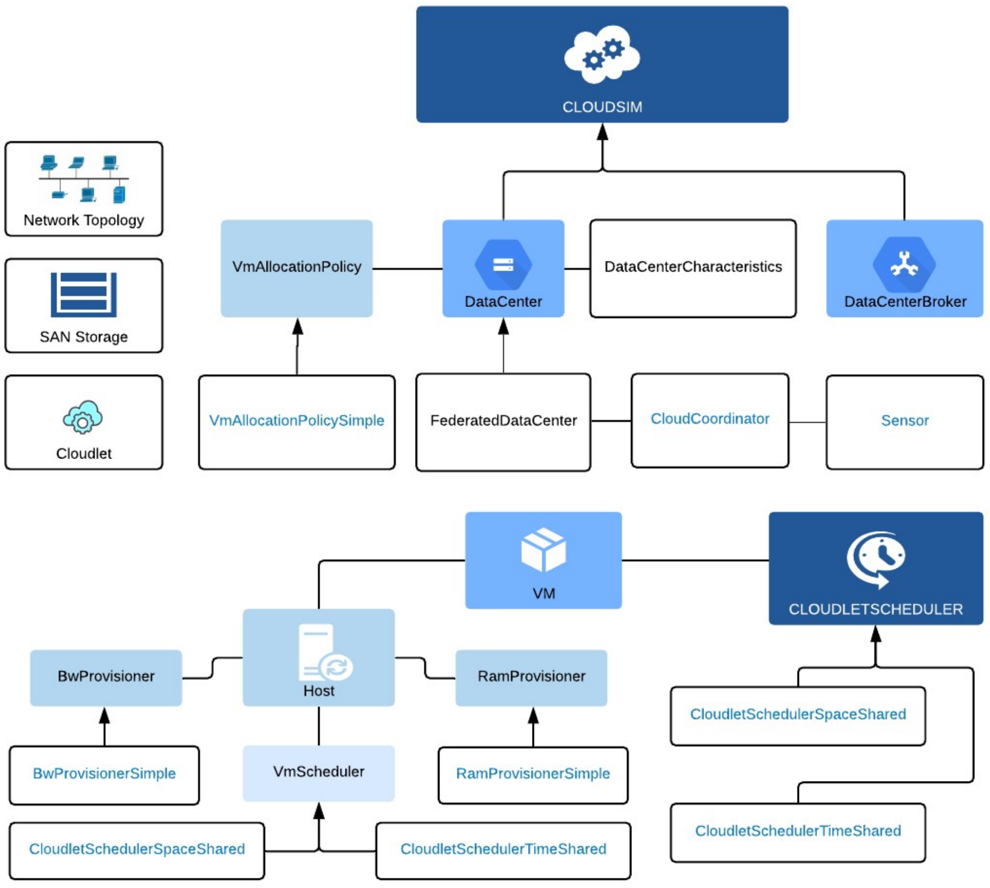


Figure 13. Cloudsim Class Design Diagram

Cloudlets are particularly advantageous for technologies that allow devices to load their computational processes towards more robust systems like machine learning, voice recognition, virtual reality, enhanced perception and language processing [73] Cloud-based devices can connect to host applications seamlessly high-bandwidth private wireless network with cloudlet. In conjunction with the proximity, this structure allows applications that allow real-time interaction to be supported. Companies can maintain latency levels for a few milliseconds. By using cloudlets, companies can optimize transaction rates among devices and applications so that the capabilities of centralized cloud infrastructures are much more significant. In some instances, cloudlets are more like private clouds than public clouds, particularly in self-management; the cloudlets often function as a buffer from the cloud. The current system runs parallel on virtual instances to the network nodes [75].

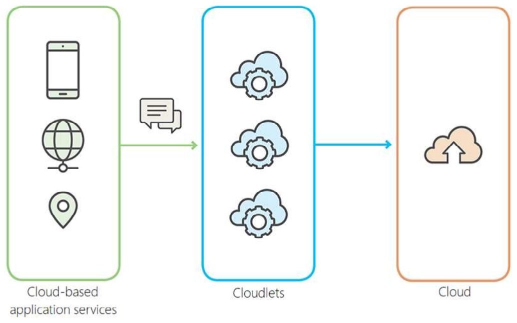


FIGURE 14. A Cloudlets in cloud

Respectively, cloud and private clouds enable organizations to launch and sustain their environments and evaluate services and applications. Many smartphones can send requests to the very same server, and other servers could remain idle. The cloudlet will act as an operations manager between all the mobile devices and the Cloud server in this environment [75]. Cloudlets restrict access to a local wireless network, while private clouds are accessible via the Internet and other WANs to promote plenty of users as required. The private cloud supports users anywhere they stand, whenever they need and from every device connected to the apps. Cloudlets vary in much the same manner from virtual private clouds (VPCs) as private clouds. Users wouldn't have to be near a VPC and can communicate from several computer types over the Internet. However, cloudlets offer better latency rates and a better consumer experience as a whole [76]. Cloudlets can effectively use virtualization to use computing resources, stabilize workloads, manage operations, and secure applications, allowing for the provision of scalable on-demand, logically divided resources from the underlying hardware elements.

While researchers have discussed cloudlets for several years, the industry is still catching up. There are also generally established principles and definitions [75]–[80]. Implementing different regulations expands this abstract class to evaluate the share of Cloudlets' computing power in a VM. As mentioned earlier, two forms of delivery schemes are available: space-shared (Cloud Scheduler Space Shared) and time-shared (Cloudlet Scheduler Time hared).

4.9 Cloud Datacenter

This class forms the core network (hardware) products offered by cloud providers (Amazon, Azure, Application Engine). It contains a series of host computers that can be consistent or heterogeneous with their hardware configurations (memory, backbone, capability and storage). In addition, each datacenter part designates a component to provide generalized applications that execute many guidelines for assigning bandwidth, memory and storage to hosts and VMs. Single or multiple several more centralized DCs are connected to a core network to host material of a cloud service provider [76], [81]. In the DCs network, servers, storage, aggregation circuits, and other edge routers are included. The contents of the data centre are transmitted through big core routers and optical connections to the edge network.

4.10 Atacenter Broker Or Cloud Broker

This class shapes a broker taking responsibility for facilitating bargaining among SaaS and cloud suppliers [82], and the QoS requirements guide the bargaining's. On behalf of the SaaS providers, the broker acts. In querying the CIS, it explores appropriate cloud service providers and performs an online negotiation process to allocate re-sources/services to satisfy the application's QoS requirements. Researchers and system developers must stretch this class to measure [83] and test custom brokering regulations. The divergence between the broker and the cloud coordinator would be that the first signifies the client (i.e. choices are taken on such elements to enhance key performance indicators for users). At the same time, the latter operates on behalf of the data centre, i.e. it attempts to optimize the total performance of the data centre without trying to take the requirements of specific clients into account. The cloud brokerage and brokerage services were studied [83],[84] and [85]. Because each broker differs with their service arrangement, numerous cloud brokers have advantages: 1) it allows users to define the best settings based on a variety of functional and non-functional criteria for their individual needs. This involves providing assis-tance and budget recommendations to determine how different programs can be chosen and implemented through various hybrid methods.

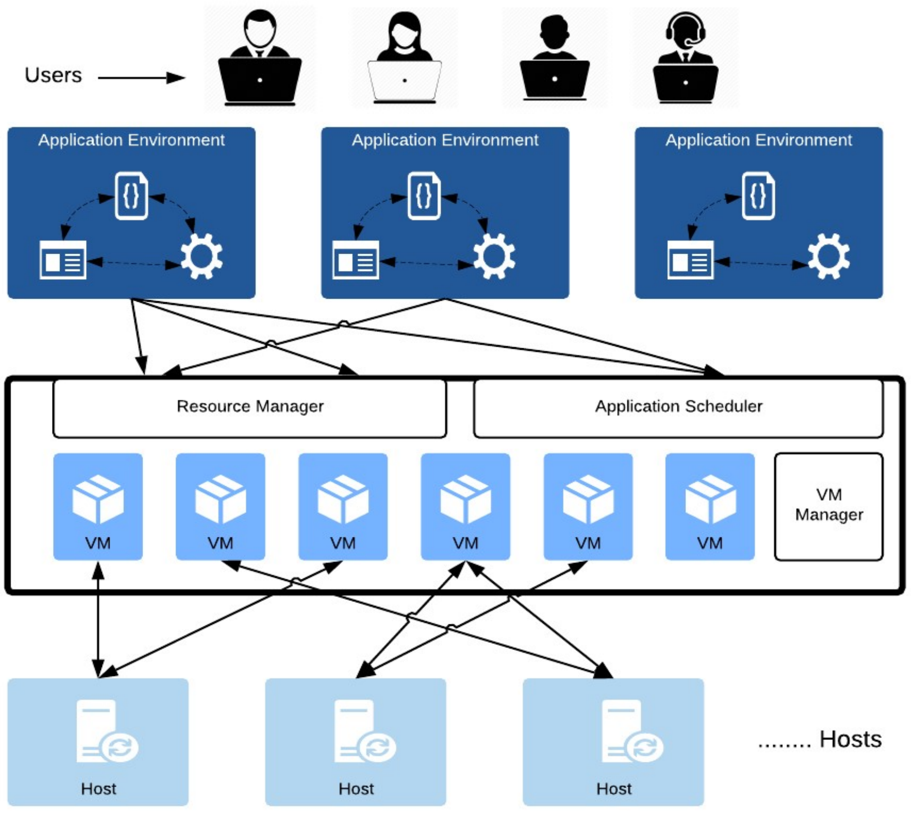


Figure 15. Vm, Application And Host Relationship In Cloud Datacenter

2) standardize cloud-based services available in the market through the integration of multi-services cloud provider outputs. 3) provide better deals and gain additional cloud providers information. 4) select for particular criteria the finest cloud service provider. 5) offer good extra workflow acceptance, enhanced control, enforcement and protection. However, cloud data centres are highly dynamic and changeful because 1) customers keep demanding VMs, unstable resource consumption patterns, 2) fluctuating use of VM resources, 3) unpredictable arrival and departure patterns for users of the datacentre, and 4) the productivity of hosts can differ tremendously when managing various loads levels. This can easily cause unequal loads in cloud datacentres and break down performance and service level agreements, which needs a load balancing technique to solve this problem. However, load balancing in clouds is a method that optimally balances the excess dynamic local workload throughout all nodes [86].

4. 11 Datacenter Characteristics

This module provides datacenter configuration information.

4. 12 Host

This class constructs a physical resource like a device or a storage server. It holds critical data such as capacity and storage quantity, a set of computing cores and size (to demonstrate a multi-core machine), resource sharing for VMs, and resource on capacity and bandwidth provisioning to VMs. A host can host multiple VMs with various potential assets and different load kinds. Servers hosting variable and unpredictable workloads heterogeneous VM can lead to an imbalance in resource use, resulting in deterioration inefficiency and infringement of the Service Level Agreements (SLAs) [107]. In situations such as VM operating a computation-intensive program with a low memory requirement, imbalance resource use [87] can be observed.

Figure (15) shows the VM, application and host relationship in cloud datacentres. Introduced first in the 1960s and is commonly used for hardware network restructuring in enterprise datacentres with innovations such as VMware [88] and Xen [89] in recent years. The hosts below are the primary source of requirements such as CPU, memory and storage. Above host, the virtualization platform server such as XEN virtualizes the physical resource and manages hosted VMs. Applications are running on VMs, and dependence between them may be predefined. Each host can be assigned with several VMs, and VMs with several applications are installed. Study [88] focuses primarily on VM threshold load balance algorithms to enhance host efficiency, mostly modelled as a dumpster container issue and proven an NP-hard problem [80]. Virtualization also strengthens the capacities and capacities of present infrastructure and resources and provides an opportunity for the datacenters to host typical infrastructure applications [80].

4.13 Network Topology

This class gives information for network behaviour induction (latencies) in the simulation. It collects topology information collected by the BRITE topology generator.

4.14 Ram Provisioner

Here is an abstract class that denotes the policy of primary storage (RAM) allocation to VMs. VM can only be executed and deployed on a host if the Ram Provisioner component authorizes that the host has the requisite amount of free memory. This same Ram Provisioner Simple does not qualify any memory restriction that a VM can require. But if the requirement exceeds the memory capacity available, it is refused.

4.15 San Storage

This class structures a storage area network that is widely used to store large pieces of data (such as Amazon S3 and Azure blob storage) in cloud-based datacentres. San Storage uses a simple protocol that simulates the retrieving of any volume of data subjected to network throughput capacity. Accessing the SAN files during execution leads to unnecessary disruptions in the operation of the job unit due to the extra latencies in the transmission of the data files via the internal datacenter network.

4.16 SENDOR

The said interface is necessary to install a sensor unit, which a cloud manager may use to track specific system performance (energy consumption, resource usage). Note that the Cloud Manager uses the complex output information for load balancing decision making. This interface defines the techniques: (i) set min, max output parameter thresholds and (ii) update the calculation from time to time. This class could be used to design the real-world services provided by top cloud service providers such as Amazon's Cloud Watch and Fabric Controller from Microsoft Azure. A datacentre could mount one or even more sensors responsible for tracking the output of a particular datacenter parameter.

4.17 VM

This class structures a VM controlled and hosted via a part of a cloud host. Every VM element has access to the following VM-related characteristics: 1) available memory, 2) CPU, 3) storage space, and 4) the internal provision of the VM, expanded from the Cloudlet Scheduler abstract element (Cloudlet Scheduler).

4.18 VM Allocation Policy

This abstract class demonstrates a supply strategy that a VM Controller uses to allocate VMs to hosts. The critical function of the VM placement regulation is to choose the host accessible in a datacenter that satisfies the storage, space and availability criteria for a VM implementation. Live migrations virtual machines indicate that the VM tends to respond from the user's perspective during migration. Live migration offers other advantages than conventional suspend / resume migration[90], such as energy conservation, load balance and online maintenance. Study [90] examines the effect of live VM migration on the efficiency of applications in the Xen VM and shows that overhead migration is appropriate but cannot be ignored entirely. As the new cloud storage datacenter promotes the live migration of many virtual machines, living migration is a typical activity. For cloud computing, the VM merger is also done for keeping with the resource needs of VMs. The merger of the VM increases the servers suspended and allows live migration of the VM. It will also help to improve fault tolerance by migration of error VMs.

4.18 VM Scheduler

Here is an abstract class executed by a host component representing the regulations needed to assign processors cores for VMs (space sharing, time-sharing). The features in this class can be cleverly exceeded to achieve rules for the sharing in processors in ap-plications. The development phase for VM can be split into the initial stage for VM deployment and the live migration phase of VM. In the initial positioning phase, some work has centred on the balance of VM load without considering the live migration [7]. At present, the VM approval policy specifies the host location to be assigned by VM, which is a crucial aspect of the scheduling process. The policy usually takes account of the host resource availability. The scheduling procedure's critical elements of the live migration phase are: VM migration rule allows cloud datacenters to set priorities as VMs are passed on to the other hosts. The VM migration rules show when a VM migration from a host to a different host can occur. This generally includes a migration threshold to initiate migration procedures. A data centre's administrator makes the decision an entry based on each host's computational capabilities, such as Red Hat and VMware [91][92]. For example, a CPU-intensive host can be installed with significantly higher CPU usage thresholds, whilst an I / O heavy host with a comparatively low CPU usage threshold can be installed. VM policy placement permits the setting of cloud datacenters' policies to choose whether the VMs from overstressed hosts must be transferred. In general, too many VMs are running an overloaded server. First of all, pick the overloaded hosts from the VM

allocation regulations. In addition to reducing host load, VM allocation regulations are designed to decide which VMs must be migrated to accomplish other objectives, such as lowering migration quantities and reducing migration latency [15].

Table 3. Detail Settings For The Workflows

Terms	MEANS	NODE COUNT
EpiSma	Epigenomics with Small Cloudlets	24
EpiMe	Epigenomics with Median Cloudlets	46
EpiLa	Epigenomics with Large Cloudlets	100
EpiHe	Epigenomics with Heavy Cloudlets	997
SiSma	Sipht with Small Cloudlets	30
SiMe	Sipht with Median Cloudlets	60
SiLa	Sipht with Large Cloudlets	100
SiHe	Sipht with Heavy Cloudlets	1000
MoSma	Montage with Small Cloudlets	25
MoMe	Montage with Median Cloudlets	50
MoLa	Montage with Large Cloudlets	100
MoHe	Montage with Heavy Cloudlets	1000

Table (3) shows the workflows detailed settings, specific terms for node sizes. VM approval regulations allow cloud datacenter to build strategies that VMs from other overloaded hosts are approved for cooperative load balancing among hosts via live migration of VMs. The rules for VM acceptance must gather information, including the following: (a) remaining hosts resource, (b) the CPU or memory of a related resource type, (c) a threshold above or below the remaining amount of resource. The acceptance policies for VM are then enforced to decide whether a specific VM is to be hosted. The following Table (4) shows the parameters used in the simulation for all algorithms in all cases, respectively. Therefore, the workflow is CloudSim based; thus, it operates within the CloudSim architecture, class diagram, etc.

Table 4. Parameters Used In The Simulation

VM	15
Datacenters	2
Number of runs	50

5. RESULTS AND DISCUSSION

A workflow application W = (T, E) is modelled as a Directed Acyclic Graph (DAG) where T = {t1, t2, …, tn} is the set of tasks and E is the set of directed edges. An edge eij of the form (ti, tj) exists if there is a data dependency between ti and tj, a case in which ti is said to be the parent task of ti and tj is said to be the child task of tj. Based on this definition, a child task cannot be executed until its parent tasks are completed. The provider specifies the unit of time in which the pay-per-use model is based; any partial utilization of the leased VM is charged as if the full-time period was consumed. The following table shows the tasks to resource mapping, where tn, are the tasks and the R = {r1, r2, …, rn} are the set of resources (VMs) that need to be linked. Each resource ri has a VM type VMri linked to it and estimated linking start time ELSTri and estimated linking finish time ELFTri. And according to the following: M\ rjti = (ti, rj, ST ti, ET ti) for all tasks; where M represents the mapping and is interpreted as follows: Task (ti) is scheduled to be run by (rj) and is starting the execution process at a time (ST ti) and finish at a time (ET ti). For comparing and benchmarking the proposed algorithm (Min-Min) with other previously used cloud task scheduling algorithms such as Max-Min, MMax-Min, and Round Robin, a simulation environment called CloudSim toolkit was used [51]. The simulation was run on Intel® core i5, 1TGB HDD and 8GB Ram on 64-bit Windows 8 operating system.

We considered three workflows, Epigenomics and Sipht, which differ from each other in terms of cloudlet size. The procession speediness of each cloudlet is measured in Million Instructions Per Second (MIPS). In analyzing the performance, we used the Makespan as a performance matrix to measure, compare and evaluate the performance of the proposed algorithms. Table 3 gives detailed descriptions of all the cases used in each of the three workflows used for the simulation. Makespan is referred to as the total execution time for the entire workflow [93]. This is mainly used to measure the efficiency of the algorithms in respect of time. It is calculated by using equations 1 and 2.

$$\textit{Makespan} = \textit{Max}\ (\textit{Ct Of}\ J(T_i, M_j)) \qquad (1)$$

$$\textit{Ct} = \textit{Job End Time} - \textit{Job Start Time} \qquad (2)$$

Makespan is used to estimate the performance of the algorithm. The time elapsed from the task submission to the cloud until the end of the request of its last task. To calculate the percentage of enhancement, the following equations are used:

$$IP = LM/HM*100 \qquad (3)$$

$$100\% - IP = FP \qquad (4)$$

Where IP is the initial percentage, LM is the lower makespan value, HM is the higher value of makespan, and FP is the final percentage. Percentage calculations of performance comparison were done in this section.

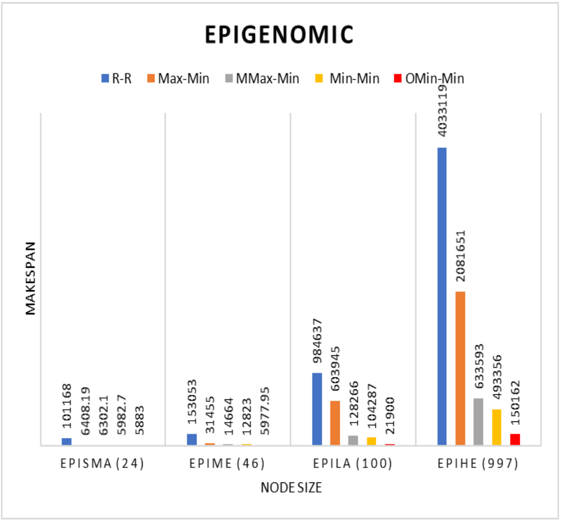


Figure 16. The Epigenomics Makespan Results Of Different Scheduling Algorithms, Small And Medium Node Size, And Large And Heavy Node Size

In table (5), a percentage of enhancements offered by our proposed algorithm against other existing algorithms is introduced, including the original Min-Min algorithm, for all workflows are shown. Specifically, our proposed algorithm outperforms the original Min-Min in all workflows as follows:

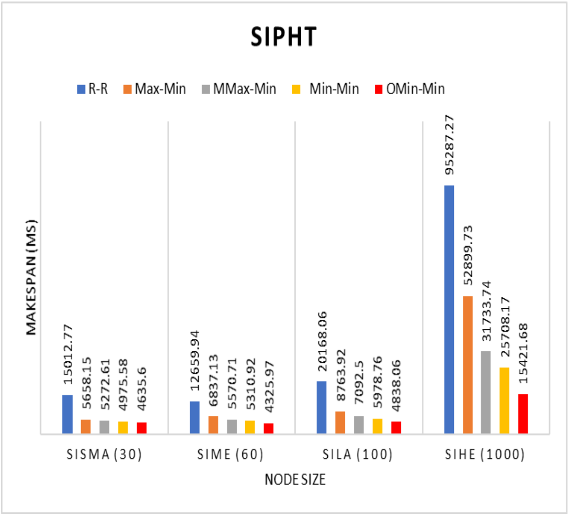


Figure 17. The Sipht Makespan Results Of Different Scheduling Algorithms, Small And Medium Node Size, And Large And Heavy Node Size

Epigenomics workflow: Achieves the best in large cloudlets by 79%, while the minor rate enhancement is in small cloudlets by 1.6%, Montage workflow: Achieves the best in large cloudlets by 43.7%, while the minor rate

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

enhancement is in small cloudlets by 4% Sipht workflow: Achieves the best in heavy cloudlets by 40%, while the mi-nor rate enhancement is in small cloudlets by 6.8%.

The OMin-Min scheduling algorithm has achieved the minimum (less) Makespan, which means better performance. In addition, Round Robin has the worst case of Makespan. In the light of Figure (16), in Epigenomics workflow results for the algorithms, the proposed OMin-Min algorithm achieves the best results compared to all other algorithms.

The RR achieves the worst among them. Refer to Figure (17), in Sipht workflow Results for the algorithms, again OMin-Min achieves the best result, and the RR achieves the worst for all workflows and in Figure (18). For calculating the exact enhancements of our proposed algorithm against all other used algorithms, we used Equations 3 and 4. The results are shown in Table 5 where the performance enhancements of our proposed algorithm in a percentage manner are compared against other algorithms for Epigenomics, Montage, and Sipht workflows. The proposed OMin-Min algorithm outperforms all other algorithms. Specifically, at the level of node sizes, the proposed algorithm achieves the best results with EpiLa for all algorithms in epigenomics workflow.

On the other hand, for the Montage workflow, the proposed algorithm achieves the best results with MoLa for Min-Min and Max-Min. It performs best with MoHe for MMax-Min, while for Round Robin, the proposed algorithm achieves very close results with MoMe and MoLa as the best. It illustrates that the algorithm output level relies on the computing time required to accomplish tasks. Each workflow has a particular architecture and many cloudlets in size (small, medium, large and heavy). Given that every workflow is dedicated to different types of applications, the comparison results show the algorithms' performance in every workflow separately, without comparing the output of the same algorithm with different workflows.

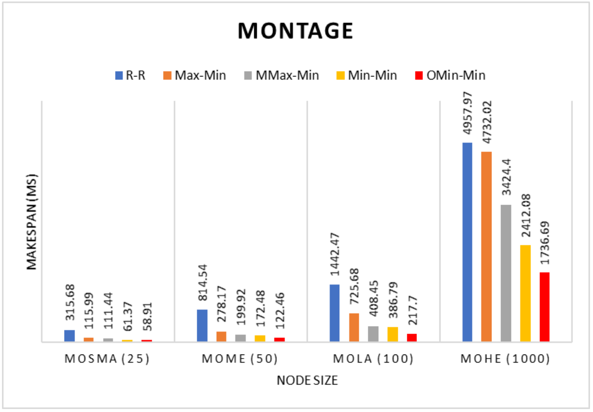


Figure 18. The Montage Makespan Results Of Different Scheduling Algorithms, Small And Medium Node Size, And Large And Heavy Node Size

Table 5. *Results Of In-Min Performance Comparison To Other Agorithms In Percentag.*

Workflow	Node size	R-R	Max-Min	MMax-Min	Min-Min
Epigenomics	EpiSma (24)	94.18%	8.19%	6.65%	1.66%
	EpiMe (46)	96.09%	80.99%	59.23%	53.38%
	EpiLa (100)	97.77%	96.37%	82.92%	79 %
	EpiHe (997)	96.27%	92.78%	76.29%	69.56%
Montage	MoSma (25)	81.33%	49.21%	47.13%	4%
	MoMe (50)	84.96%	55.97%	38.74%	29%
	MoLa (100)	84.90%	70%	46.70%	43.71%
	MoHe (1000)	64.97%	63.29%	49.28%	28%
Sipht	SiSma (30)	69.12%	18.07%	12.08%	6.83%
	SiMe (60)	65.82%	36.72%	22.34%	18.54%
	SiLa (100)	76.01%	44.79%	31.78%	19.07%
	SiHe (1000)	83.81%	70.84%	51.40%	40.01%

Table 6. Comparison Between Our Work And Oher Minmin Algorithms With Pros And Cons.

References	Year	Environment	Method used	Performance matrices	Discerption	Node size	Workflow Type and number of nodes	Pros	Cons
Our Work	2021	Could	Optimized Min-Min algorithm	Makespan	Our work proposes a new noble mechanism Optimize Min-Min (OMIN-MIN) algorithm, and we provide a comprehensive review of the cloud and scheduling process;	Montage_25 Montage_50 Montage_100 Montage_1000 Epigenomic 24 Epigenomic 46 Epigenomic 100 Epigenomic 997 Sipht 30 Sipht 60 Sipht 100 Sipht 1000	Three types of workflows' (Montage Cybershake and Epigenomic)	This algorithm selects tasks with minimum execution time on a faster available machine or resource that is capable of giving minimum completion time moreover, omin-Min can produce reasonable quality solutions, producing good makespan. Moreover, we provide a comprehensive review of the cloud and scheduling which can be good for other researcher and developers in cloud computing area	Our work considers 3 types of workflows and didn't consider all types of workflows such as cybershake and Lego.
[95]	2021	Cloud	Multi-Objective Workflow Optimization Strategy (MOWOS) And MaxVM and MinVM selection are presented in MOWOS for task allocations.	Cost Makespan	Resented a novel workflow scheduling algorithm named Multi-Objective Workflow Optimization Strategy (MOWOS) to jointly reduce execution cost and execution makespan, the other two algorithms And the other two algorithm, used to presented in MOWOS for task allocations.	Montage_25 Montage_50 Montage_100 Montage_1000 Sipht30 Sipht 60 Sipht100 Sipht 1000 Logo Inspiral30 Logo Inspiral50 Logo Inspirall00 Logo Inspirall000	Three types of workflows Cybershake, SIPHT, and LIGO Inspiral	Employs tasks splitting mechanism to split big tasks into sub-tasks to reduce their scheduling length. And to enable the tasks to s meet their deadlines at a less time and low cost	They didn't consider all types f workflows Moreover, they didn't consider Load lancing and Energy consumption.
[38]	2016	Cloud	Enhanced MIN-MIN algorithm	Makespan	Studied different task scheduling algorithms and an Enhanced Min-min algorithm is developed.	Montage_25 Montage_50 Montage_100 Montage_1000 CyberShake_30 CyberShake_50 CyberShake_100 CyberShake_1000	Two types of workflows' Montage and Cybershake	The current algorithm selects tasks with maximum execution time on a faster available machine or resource that is capable of giving minimum completion time moreover, it achieved better quality solutions and better good values of Makespan in terms of performance analysis, experimentation and benchmarking	Only consider montage and cybershake

Table 6. Count…

References	Year	Environme	Method used	Performance matrices	Discerption	Node size	Workflow Type and number of nodes	Pros	Cons
[96]	2015	Cloud	Enhanced load balance Min-Min	Makespan	This algorithm selects the task with maximum completion time then assign it to the appropriate resource to produce the best makespan and utilize resource efficiently.	No workflow used Node size =0	No workflow Used	Using the unutilized resources effectively, the algorithm selects the task with maximum completion time. It assigns it to the appropriate resource to produce less makespan Scheduling in two phases as it uses the advantages of two algorithms (Max-Min And Min-Min), Such as good completion time .	It covers the disadvantages of both Max-Min and Min-Min, such as poor load balancing and less consideration for QoS.
[35]	2013	Cloud	An Improved Min-Min Algorithm in Cloud Computing	Resource utilization rate	Introduce a comparison simulation for comparing the improved min-min algorithm with the traditional min-min algorithm.	No workflow used Node size =0	No workflow used	Enhance resource utilization rate, more prolonged activities may be completed in an acceptable amount of time, and users' expectations are met	Some priority constraints.
[36]	2013	Cloud	Improved Min-Min Algorithm	Load balancing and utilization of system resources	An efficient rescheduling based task scheduling algorithm (improved MIN-MIN Algorithm (I MIN-MIN) in order to enhance system performance .	No workflow used Node size =0	No workflow used	Minimizes the makespan with load balancing And guarantees the high system availability in system performance.	They didn't consider the priority of the task.
[37]	2010	Grid	Improved MIN-MIN	Total execution time	Priority model which improves the Min-min algorithm based on QoS constraints is introduced and High-priority tasks can be implemented first on the condition that High QoS tasks get high QoS resources,	No workflow used Node size =0	No workflow Used	High-QoS resources	Does not take into account. The level of task providers and The deadtime for completing the task into account.

7. CONCLUSION

In this work, we have shown various kinds of cloud systems, the scheduling strategies, the critical phases of the simulation, and a classification of scientific workflows. Reviewed several characteristics and features from literature for distributed systems. We have provided details about the importance of cloudlets, VMs and data centres in designing and implementing the scheduling process in the cloud environment.

This research seeks to perform task scheduling better and resource allocation than previous research works in a cloud computing environment. Performance analysis, experimentation, and benchmarking of the results indicate that our approach using OMin-Min can produce reasonable quality solutions, producing good makespan value compared to the standard Max-Min, Min-Min, Round-Robin and (Modified Max-Min (MMax

Min)) algorithms. We simulated our approach using the CloudSim toolkit, which saw a frequent reduction in waiting time, leading to less makespan as more and more cloudlets arrived to be scheduled. It can be seen that (in-Min) leads to better performance since it achieved less makespan comparing with other algorithms in three types of workflows (Epigenomics, Sipht and Montage), Min-Min has achieved less Makespan in three different workflows as well, in (data-intensive) scientific workflows and (computational) scientific workflow. For future work, balancing load effectively and cost can be considered with more algorithms that could be used to evaluate more system parameters and more workflows that may further improve the performance. In addition, deployment of task scheduling (TS) and resource allocation (RA) in other systems where resource allocation is essential, such as Li-Fi and Li-Fi/RF hybrid network systems, can be investigated.

REFERENCES

[1] A. Abraham, R. Buyya, and B. Nath, "Nature ' s Heuristics for Scheduling Jobs on Computational Grids," pp. 1–8.

[2] B. K. Rani, B. P. Rani, and A. V. Babu, "Cloud Computing and Inter-Clouds – Types, Topologies and Research Issues," Procedia Comput. Sci., vol. 50, pp. 24–29, 2015, doi: https://doi.org/10.1016/j.procs.2015.04.006.

[3] P. Mell, T. Grance, and others, "The NIST definition of cloud computing," 2011.

[4] A. Fox et al., "Above the clouds: A berkeley view of cloud computing," Dept. Electr. Eng. Comput. Sci. Univ. California, Berkeley, Rep. UCB/EECS, vol. 28, no. 13, p. 2009, 2009.

[5] L. Zhao, S. Sakr, A. Liu, and A. Bouguettaya, Cloud data management. Springer, 2014.

[6] W.-T. Wen, C.-D. Wang, D.-S. Wu, and Y.-Y. Xie, "An ACO-based scheduling strategy on load balancing in cloud computing environment," in 2015 Ninth International Conference on Frontier of Computer Science and Technology, 201s5, pp. 364–369.

[7] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers," in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, 2010, pp. 1–14.

[8] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," J. internet Serv. Appl., vol. 1, no. 1, pp. 7–18, 2010.

[9] M. Pinedo and K. Hadavi, "Scheduling: theory, algorithms and systems development," in Operations Research Proceedings 1991, Springer, 1992, pp. 35–42.

[10] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2011, pp. 1–12.

[11] M. Choudhary and S. K. Peddoju, "A dynamic optimization algorithm for task scheduling in cloud environment," Int. J. Eng. Res. Appl., vol. 2, no. 3, pp. 2564–2568, 2012.

[12] F. de O. Lucchese, E. J. H. Yero, F. S. Sambatti, and M. A. A. Henriques, "An adaptive scheduler for Grids," J. Grid Comput., vol. 4, no. 1, pp. 1–17, 2006.

[13] R. Xu, Y. Wang, W. Huang, D. Yuan, Y. Xie, and Y. Yang, "Near-optimal dynamic priority scheduling strategy for instance-intensive business workflows in cloud computing," Concurr. Comput. Exp., vol. 29, no. 18, Sep. 2017, doi: 10.1002/cpe.4167.

[14] L. F. Bittencourt, R. Sakellariou, and E. R. M. Madeira, "Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm," in 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010, pp. 27–34.

[15] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in 2010 3rd International symposium on parallel architectures, algorithms and programming, 2010, pp. 89–96.
[16] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," J. ACM, vol. 24, no. 2, pp. 280–289, 1977.
[17] B. Scholz-Reiter, J. Heger, and T. Hildebrandt, "Analysis and comparison of dispatching rule-based scheduling in dual-resource constrained shop-floor scenarios," in Proceedings of the World Congress on Engineering and Computer Science, 2009, vol. 2, pp. 20–22.
[18] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, "Cost and makespan-aware workflow scheduling in hybrid clouds," J. Syst. Archit., vol. 100, p. 101631, 2019.
[19] J. K. O. K. Konjaang, F. H. Ayob, and A. Muhammed, "AN OPTIMIZED MAX-MIN SCHEDULING ALGORITHM IN CLOUD COMPUTING," vol. 95, no. 9, pp. 1916–1926, 2017.
[20] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs," PLoS One, vol. 3, no. 9, p. e3197, 2008.
[21] C. Team, "Dagman (directed acyclic graph manager)," See website http//www. cs. wisc. edu/condor/dagman, 2005.
[22] F. Baroncelli, B. Martini, and P. Castoldi, "Network virtualization for cloud computing," Ann. Telecommun. des télécommunications, vol. 65, no. 11, pp. 713–721, 2010.
[23] S. Bharathi and M. Rey, "Characterization of Scientific Workflows." In 2008 third workshop on workflows in support of large-scale science, pp. 1-10. IEEE, 2008.
[24] S. S. Brar and S. Rao, "Optimizing workflow scheduling using Max-Min algorithm in cloud environment," Int. J. Comput. Appl., vol. 124, no. 4, 2015.
[25] T. D. Braun et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," J. Parallel Distrib. Comput., vol. 61, no. 6, pp. 810–837, 2001.
[26] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, "A task scheduling algorithm based on QoS-driven in cloud computing," Procedia Comput. Sci., vol. 17, pp. 1162–1169, 2013.
[27] A. B. El-Sisi, M. A. Tawfeek, and others, "Cloud Task Scheduling for Load Balancing based on Intelligent Strategy.," Int. J. Intell. Syst. \& Appl., vol. 6, no. 5, 2014.
[28] H. Zhong, K. Tao, and X. Zhang, "An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems," pp. 0–5, 2010, doi: 10.1109/ChinaGrid.2010.37.
[29] N. Kaur and K. Kaur, "Improved Max-Min Scheduling Algorithm," vol. 17, no. 3, pp. 42–49, 2015, doi: 10.9790/0661-17314249.
[30] S. S. Chauhan, "QoS Guided Heuristic Algorithms for Grid Task Scheduling," vol. 2, no. 9, pp. 24–31, 2010.
[31] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in 2013 National Conference on Parallel computing technologies (PARCOMPTECH), 2013, pp. 1–8.
[32] A. Girault, E. Saule, and D. Trystram, "Reliability versus performance for critical applications," J. Parallel Distrib. Comput., vol. 69, no. 3, pp. 326–336, 2009.
[33] X. He, X. Sun, and G. Von Laszewski, "QoS guided min-min heuristic for grid task scheduling," J. Comput. Sci. Technol., vol. 18, no. 4, pp. 442–451, 2003.
[34] C. N. Yang, "Message from the UMAS 2013 chair," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7861 LNCS, no. May 2013, pp. 102–113, 2013, doi: 10.1007/978-3-642-38027-3.
[35] G. Liu, J. Li, and J. Xu, "An improved min-min algorithm in cloud computing," Adv. Intell. Syst. Comput., vol. 191 AISC, no. 1, pp. 47–52, 2013, doi: 10.1007/978-3-642-33030-8_8.
[36] R. kaur and P. Kumar Patra, "Resource Allocation with improved Min-Min Algorithm," Int. J. Comput. Appl., vol. 76, no. 15, pp. 61–67, 2013, doi: 10.5120/13327-0918.
[37] J. Liu and G. Li, "An improved MIN-MIN grid tasks scheduling algorithm based on QoS constraints," OPEE 2010 - 2010 Int. Conf. Opt. Photonics Energy Eng., vol. 1, pp. 281–283, 2010, doi: 10.1109/OPEE.2010.5508132.
[38] M. Haladu and J. Samual, "Optimizing Task Scheduling and Resource allocation in Cloud Data Center, using Enhanced Min-

Min Algorithm," IOSR J. Comput. Eng., vol. 18, no. 04, pp. 18–25, 2016, doi: 10.9790/0661-1804061825.

[39] W. Chen, M. Rey, and M. Rey, "WorkflowSim : A Toolkit for Simulating Scientific Workflows in Distributed Environments," 2012.

[40] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, 2002.

[41] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," J. Parallel Distrib. Comput., vol. 59, no. 2, pp. 107–131, 1999.

[42] H. Cao, H. Jin, X. Wu, S. Wu, and X. Shi, "DAGMap: efficient and dependable scheduling of DAG workflow job in Grid," J. Supercomput., vol. 51, no. 2, pp. 201–223, 2010.

[43] Y. Chawla and M. Bhonsle, "A study on scheduling methods in cloud computing," Int. J. Emerg. Trends \& Technol. Comput. Sci., vol. 1, no. 3, pp. 12–17, 2012.

[44] Y. Xu, K. Li, L. He, L. Zhang, and K. Li, "A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems," IEEE Trans. parallel Distrib. Syst., vol. 26, no. 12, pp. 3208–3222, 2014.

[45] R. Vijayalakshmi and V. Vasudevan, "Static batch mode heuristic algorithm for mapping independent tasks in computational grid," 2015.

[46] S. Nesmachnow, H. Cancela, and E. Alba, "A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling," Appl. Soft Comput., vol. 12, no. 2, pp. 626–639, 2012.

[47] S. Ravichandran and E. R. Naganathan, "Dynamic scheduling of data using genetic algorithm in cloud computing," Int. J. Comput. Algorithm, vol. 2, no. 01, pp. 127–133, 2013.

[48] S. Patel and U. Bhoi, "Priority based job scheduling techniques in cloud computing: a systematic review," Int. J. Sci. \& Technol. Res., vol. 2, no. 11, pp. 147–152, 2013.

[49] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," IEEE Trans. Softw. Eng., vol. 14, no. 2, pp. 141–154, 1988.

[50] N. Kaur and K. Kaur, "Improved max-min scheduling algorithm," IOSR J. Comput. Eng., vol. 17, no. 3, pp. 42–49, 2015.

[51] R. Duan, R. Prodan, and X. Li, "Multi-Objective Game Theoretic Schedulingof Bag-of-Tasks Workflows on Hybrid Clouds," IEEE Trans. Cloud Comput., vol. 2, no. 1, pp. 29–42, 2014, doi: 10.1109/TCC.2014.2303077.

[52] S. Meraji and M. R. Salehnamadi, "A batch mode scheduling algorithm for grid computing," J. Basic Appl. Sci. Res., vol. 3, no. 4, pp. 173–181, 2013.

[53] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," J. Netw. Comput. Appl., vol. 66, pp. 64–82, 2016, doi: https://doi.org/10.1016/j.jnca.2016.01.018.

[54] A. Belgacem, K. Beghdad-Bey, and H. Nacer, "Task scheduling in cloud computing environment: A comprehensive analysis," in International Conference on Computer Science and its Applications, 2018, pp. 14–26.

[55] M. Maciej, G. Juve, and others, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press, 2012.

[56] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 8, pp. 1400–1414, 2011.

[57] R. F. Freund et al., "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet," in Proceedings Seventh Heterogeneous Computing Workshop (HCW'98), 1998, pp. 184–199.

[58] S. Pars and R.-R. Maleki, "A new task scheduling algorithm in grid environment," Int. J. Digit. Content Technol. its Appl., vol. 3, no. 4, pp. 152–160, 2009.

[59] G. B. Berriman and J. C. Good, "The application of the montage image mosaic engine to the visualization of astronomical images," Publ. Astron. Soc. Pacific, vol. 129, no. 975, p. 58006, 2017.

[60] G. B. Berriman and J. Good, "TOASTing Your Images With Montage," in American Astronomical Society Meeting Abstracts\# 229, 2017, vol. 229, pp. 209–236.

[61] E. Deelman et al., "Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example," in 2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06), 2006, p. 14.
[62] S. Callaghan et al., "Reducing time-to-solution using distributed high-throughput mega-workflows-experiences from SCEC CyberShake," in 2008 IEEE Fourth International Conference on eScience, 2008, pp. 151–158.
[63] E. Deelman et al., "Pegasus, a workflow management system for science automation," Futur. Gener. Comput. Syst., vol. 46, pp. 17–35, 2015, doi: https://doi.org/10.1016/j.future.2014.10.008 .
[64] M. E. A. Budimir, P. M. Atkinson, and H. G. Lewis, "Seismically induced landslide hazard and exposure modelling in Southern California based on the 1994 Northridge, California earthquake event," Landslides, vol. 12, no. 5, pp. 895–910, 2015.
[65] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, "A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis," in Workflows for e-Science, Springer, 2007, pp. 39–59.
[66] A. Verma and S. Kaushal, "Cost-Time Efficient Scheduling Plan for Executing Workflows in the Cloud," J. GRID Comput., vol. 13, no. 4, pp. 495–506, Dec. 2015, doi: 10.1007/s10723-015-9344-9.
[67] I. Gupta, M. S. Kumar, and P. K. Jana, "Compute-intensive workflow scheduling in multi-cloud environment," in 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016, pp. 315–321, doi: 10.1109/ICACCI.2016.7732066.
[68] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," J. Grid Comput., vol. 3, no. 3–4, pp. 171–200, 2005.
[69] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," Proc. VLDB Endow., vol. 3, no. 1–2, pp. 460–471, 2010.
[70] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds," IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 222–235, 2014, doi: 10.1109/TCC.2014.2314655.
[71] T. D. Braun et al., "A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems," in Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99), 1999, pp. 15–29.
[72] R. N. Calheiros, R. Ranjan, A. Beloglazov, and A. F. De Rose, "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," no. August 2010, pp. 23–50, 2011, doi: 10.1002/spe.
[73] 大里齊, "Research and development of microwave dielectric ceramics for wireless communications," J. Ceram. Soc. Japan, vol. 113, no. 1323, pp. 703–711, 2005.
[74] M. Satyanarayanan et al., "An open ecosystem for mobile-cloud convergence," IEEE Commun. Mag., vol. 53, no. 3, pp. 63–70, 2015.
[75] U. Shaukat, E. Ahmed, Z. Anwar, and F. Xia, "Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges," J. Netw. Comput. Appl., vol. 62, pp. 18–40, 2016.
[76] G. I. Klas, "Fog computing and mobile edge cloud gain momentum open fog consortium, etsi mec and cloudlets," Google Sch., 2015.
[77] Q. Shi, L. Liu, W. Xu, and R. Zhang, "Joint transmit beamforming and receive power splitting for MISO SWIPT systems," IEEE Trans. Wirel. Commun., vol. 13, no. 6, pp. 3269–3280, 2014, doi: 10.1109/TWC.2014.041714.131688.
[78] P. M. Kalaivaanan et al., "Measuring Contention and Congestion on Ad-Hoc Multicast Network towards Satellite on Ka-Band and LiFi Communication under Tropical Environment Region," IEEE Access, vol. 8, pp. 108942–108951, 2020, doi: 10.1109/ACCESS.2020.3001619.
[79] Y. Xu et al., "Joint Beamforming and Power-Splitting Control in Downlink Cooperative SWIPT NOMA Systems," IEEE Trans. Signal Process., vol. 65, no. 18, pp. 4874–4886, 2017, doi: 10.1109/TSP.2017.2715008.
[80] E. G. C. man Jr, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: A survey," Approx. algorithms NP-hard Probl., pp. 46–93, 1996.
[81] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," IEEE J. Sel. Areas Commun., vol. 34, no. 5, pp. 1728–1739, 2016.

[82] L. M. BIJU, S. S. Kumar, R. M. Biju, M. L. Madhavu, and others, "A Survey on Various Cloud Aspects.," Int. J. Adv. Res. Comput. Sci., vol. 7, no. 5, 2016.

[83] F. Fowley, C. Pahl, and L. Zhang, "A comparison framework and review of service brokerage solutions for cloud architectures," in International Conference on Service-Oriented Computing, 2013, pp. 137–149.

[84] F. Fowley, C. Pahl, P. Jamshidi, D. Fang, and X. Liu, "A classification and comparison framework for cloud service brokerage architectures," IEEE Trans. Cloud Comput., vol. 6, no. 2, pp. 358–371, 2016.

[85] D. M. Elango, F. Fowley, and C. Pahl, "Using a cloud broker API to evaluate cloud service provider performance," in Joint Pre-Proceedings of the Workshops Associated with ESOCC 2017, 2017, p. 63.

[86] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, 2010, pp. 551–556.

[87] A. Kerr, G. Diamos, and S. Yalamanchili, "A characterization and analysis of GPGPU kernels," 2009.

[88] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," Concurr. Comput. Pract. Exp., vol. 29, no. 12, p. e4123, 2017.

[89] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, 2008, pp. 1–12.

[90] Y. Grunenberger, I. Tinnirello, P. Gallo, E. Goma, and G. Bianchi, "Wireless card virtualization: From virtual NICs to virtual MAC machines," in 2012 Future Network \& Mobile Summit (FutureNetw), 2012, pp. 1–10.

[91] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," Neural Comput. Appl., vol. 26, no. 6, pp. 1297–1309, 2015.

[92] W. Xiaojun, W. Yun, H. Zhe, and D. Juan, "The research on resource scheduling based on fuzzy clustering in cloud computing," in 2015 8th International Conference on Intelligent Computation Technology and Automation (ICICTA), 2015, pp. 1025–1028.

[93] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A {GSA} based hybrid algorithm for bi-objective workflow scheduling in cloud computing," Futur. Gener. Comput. Syst., vol. 83, pp. 14–26, 2018, doi: https://doi.org/10.1016/j.future.2018.01.005
.

[94] G. B. Berriman et al., "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," in Optimizing Scientific Return for Astronomy through Information Technologies, 2004, vol. 5493, pp. 221–232.

[95] Konjaang, J. Kok, and Lina Xu. "Multi-objective workflow optimization strategy (MOWOS) for cloud computing." Journal of Cloud Computing 10, no. 1 (2021): 1-19.

[96] G. Patel, R. Mehta, and U. Bhoi, "Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing," Procedia Comput. Sci., vol. 57, pp. 545–553, 2015, doi: 10.1016/j.procs.2015.07.385.