Journal of Computer Science

Original Research Paper

Cost Effective Expa-Max-Min Scientific Workflow Allocation and Load Balancing Strategy in Cloud Computing

James Kok Konjaang, Fahrul Hakim Ayob and Abdullah Muhammed

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia

Article history
Received: 22-09-2017
Revised: 26-12-2017
Accepted: 13-01-2018

Corresponding Author:
James Kok Konjaang
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia,
Malaysia
Email: larisco10@yahoo.com
jameskonjangkok@bpoly.edu.gh

Abstract: The rise in demand for cloud resources (network, hardware and software) requires cost effective scientific workflow scheduling algorithm to reduce cost and balance load of all jobs evenly for a better system throughput. Getting multiple scientific workflows scheduled with a reduced makespan and cost in a dynamic cloud computing environment is an attractive research area which needs more attention. Scheduling multiple workflows with the standard Max-Min algorithm is a challenge because of the high priority given to task with maximum execution time first. To overcome this challenge, we proposed a new mechanism call Expanded Max-Min (Expa-Max-Min) algorithm to effectively give equal opportunity to both cloudlets with maximum and minimum execution time to be scheduled for a reduce cost and time. Expa-Max-Min algorithm first calculates the completion time of all the cloudlets in the cloudletList to find cloudlets with minimum and maximum execution time, then it sorts and queue the cloudlets in two queues based on their execution times. The algorithm first select a cloudlet from the cloudletList in the maximum execution time queue and assign it to a resource that produces minimum completion time, while executing cloudlets in the minimum execution time queue concurrently. The experimented results demonstrats that our proposed algorithm, Expa-Max-Min algorithm, is able to produce good quality solutions in terms of minimising average cost and makespan and able to balance loads than Max-Min and Min-Min algorithms.

Keywords: Cloud Computing, Workflows, Expa-Max-Min, Load Balancing, Makespan and Cost

Introduction

The evolution of cloud computing has taken place in various phases which include grid computing, utility computing, software as a service and now cloud computing (Sharma and Pariha, 2014). Cloud computing is classified as a type of grid computing, which is aimed at allocating resources properly to address the quality of service and reliability problems facing information dissemination and data storage (Sharma and Pariha, 2014; Hamdaqa and Tahvildari, 2012). NIST defined cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources like, networks, servers, storage, applications and other services" (Mell and Grance, 2011). Cloud technology has a broader network range that enables cloud resources to be available anywhere in the world by the cloud service providers to consumers on demand. Consumers or users all over the world can access these resources through the use of their digital devices such as laptops, smart phones, tablets, personal computers etc., on pay-as-you-go model or other models through cloud standard mechanisms (Mell and Grance, 2011).

Though cloud computing has gain a lot of successes since its inception, but scientific workflow allocation with the standard Max-Min algorithm in cloud is still a major issue in research because it is unable to select and assign both cloudlets with maximum execution time and minimum execution time concurrently. Scientific workflow allocation is described as a method used to get resources channeled to cloud users at a reduced cost and time. In scheduling, a workflow task has to be selected and submitted to the resource manager and then, the submitted workflow task will be put in a queue if there are other workflow tasks with higher priority than it, until it is time for it to be scheduled (Saraswathi *et al*., 2015) as displayed in Fig. 1. A workflow task may be required to wait in a queue for a long or short period of time depending on some scheduling factors which include system capacity to execute, task priority, system load and requested resources availability.

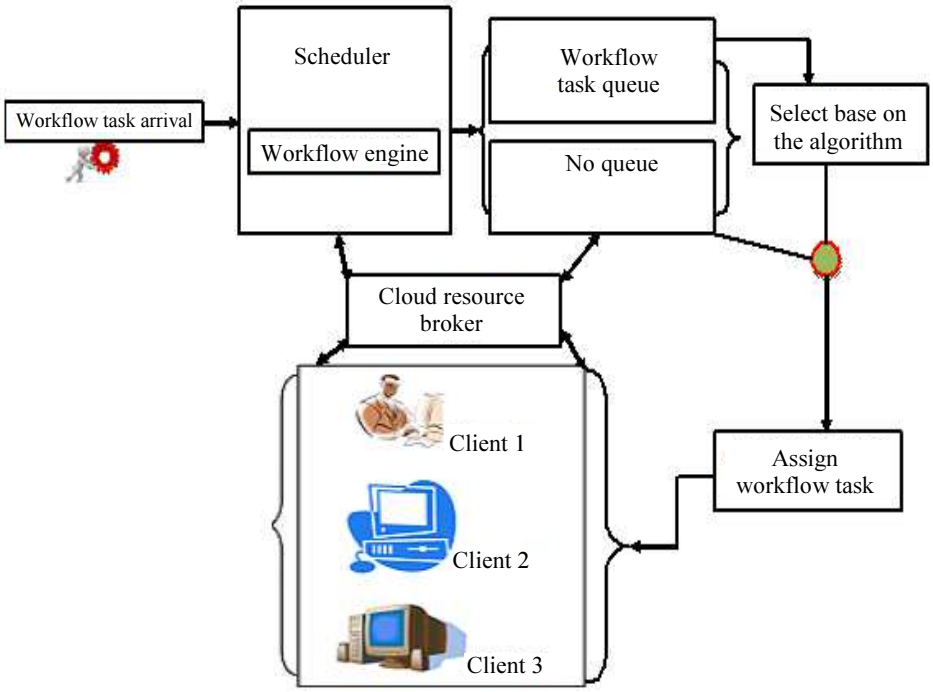


Fig. 1: Workflow task scheduling in cloud

The focus of this paper is on how to achieve the control of workload, makespan and cost on cloud resources by ensuring that all the workloads on cloud resources are distributed properly to allow free flow of cloudlets on all the network nodes and to guarantee that, all the resources are allocated to cloud users at a lower cost. This will optimise the use of scientific workflows in cloud computing to ensure that multiple resources are made available to cloud users at a reduced cost and time. To achieve this, we proposed a mechanism called Expa-Max-Min algorithm as a way to ensure better load balancing, makespan and cost of executing cloudlets on cloud resources in cloud computing environment. The algorithm takes the advantages of the standard Max-Min algorithm by selecting and assigning cloudlets that requires minimum executing time in the second queue of the resources alongside the original Max-Min steps as well in the first queue of the resource.

We incorporated the proposed algorithm (Expa-Max-Min) into a simulation environment called cloudsim. Cloudsim is a simulation toolkit that allows for the modeling and simulation of all cloud applications and systems for comparison of result in cloud computing environment (Calheiros *et al*., 2011). The algorithm is implemented to guarantee that all workloads on cloud resources are distributed evenly on all the available network nodes to avoid traffic during cloudlets distribution and also to reduce the high priority in the standard Max-Min algorithm where cloudlets with maximum execution time must finish execution before executing those with minimum execution time. Expa-Max-Min algorithm first calculates the completion time of all the cloudlets in the cloudletList to find cloudlets with minimum and maximum execution time, then it sorts and queue the cloudlets in two queues based on their execution times. The algorithm first selects a cloudlet from the cloudletList in the maximum execution time queue and assigns it to a resource that produces minimum execution time, followed by cloudlet in the minimum execution time queue. The proposed algorithm is able to boost up cloud scheduling processes by simultaneously selecting and assigning cloudlets with both maximum and minimum execution time concurrently at a reduced cost, makespan and balancing loads fairly.

Load Balancing

The increasing demand for cloud applications coupled with the increased in web traffic daily, where millions of cloud users are queued and demanding for cloud services on different servers has made load balancing an interesting topic to be investigated. Load balancing is a method used in cloud and in distributed computing to mediate access to client requests to servers by distributing cloudlets evenly across multiple servers, (Samarsinh and Deshpande, 2014). The purpose of balancing load on cloud is to ensure that workloads on cloud resources are distributed evenly on all the available network nodes to avoid traffic during cloudlets distribution. Also, load balancing in cloud helps cloud users and enterprises to manage their application effectively for higher performance at a lower cost. Figure 2 displays the processes of load balancing in Cloud.

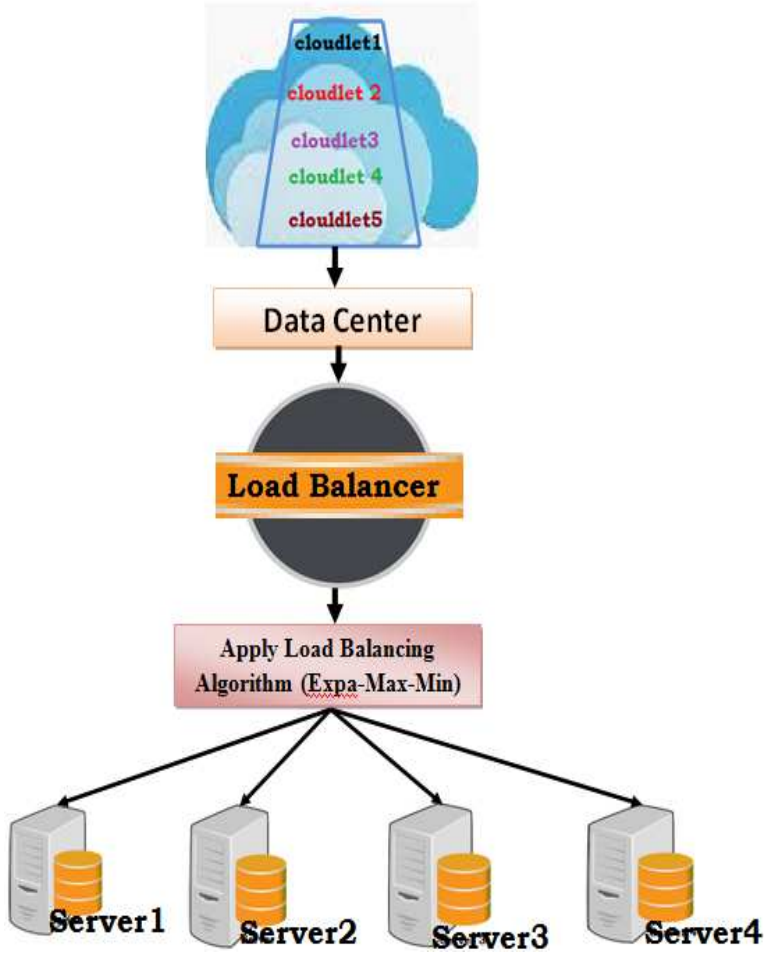


Fig. 2: The processes of load balancing in cloud

Scientific Workflows

Scientific Workflows allocation and Scheduling is a procedure adapted to manage the execution of interdependent workflow tasks on a distributed pool of resources. This procedure ensures that appropriate resources are identified and distributed on all the available VMs for a successful execution of all tasks to meet the demands of cloud users (Prathibha *et al*., 2014). The three types of scientific workflows used in this research are:

- Laser Interferometer Gravitational Wave Observatory (LIGO) (Silva *et al*., 2014), also known as Inspiral Workflow is a data-intensive workflow designed to monitor, detect and capture gravitational waves that is produced through various activities in the earth. The main goal of LIGO Inspiral is to identify and give measurement of gravitational waves forecasted by general relativity (Einstein's theory of gravity), in which gravity is described as due to the curvature of the fabric of time and space (Berriman *et al*., 2004; Mehta and Gideon, 2014)
- CyberShake: This is a fraction of Southern California Earthquake Center's (SCEC) (Graves *et al*., 2011), which uses 3D wave detector to determine source and wave that allows it to compute Probabilistic Seismic Hazard curves for geographic sites in order to simulate, broadcast and to forecast the movement of ground for the production of accurate and reliable environmental estimates rather than relying on the use of empirical-based ground motion methods (Silva *et al*., 2014; Cpedia, 2016)
- SIPHT: This is a scientific workflow application from the bioinformatics project at Harvard, which is suitable for conducting a broader search for non-translated RNAs (sRNAs) for the control of several processes like secretion or virulence in bacteria (Mehta and Gideon, 2014; Livny, 2016). Figure 3 displays a summarized Scientific Workflows.

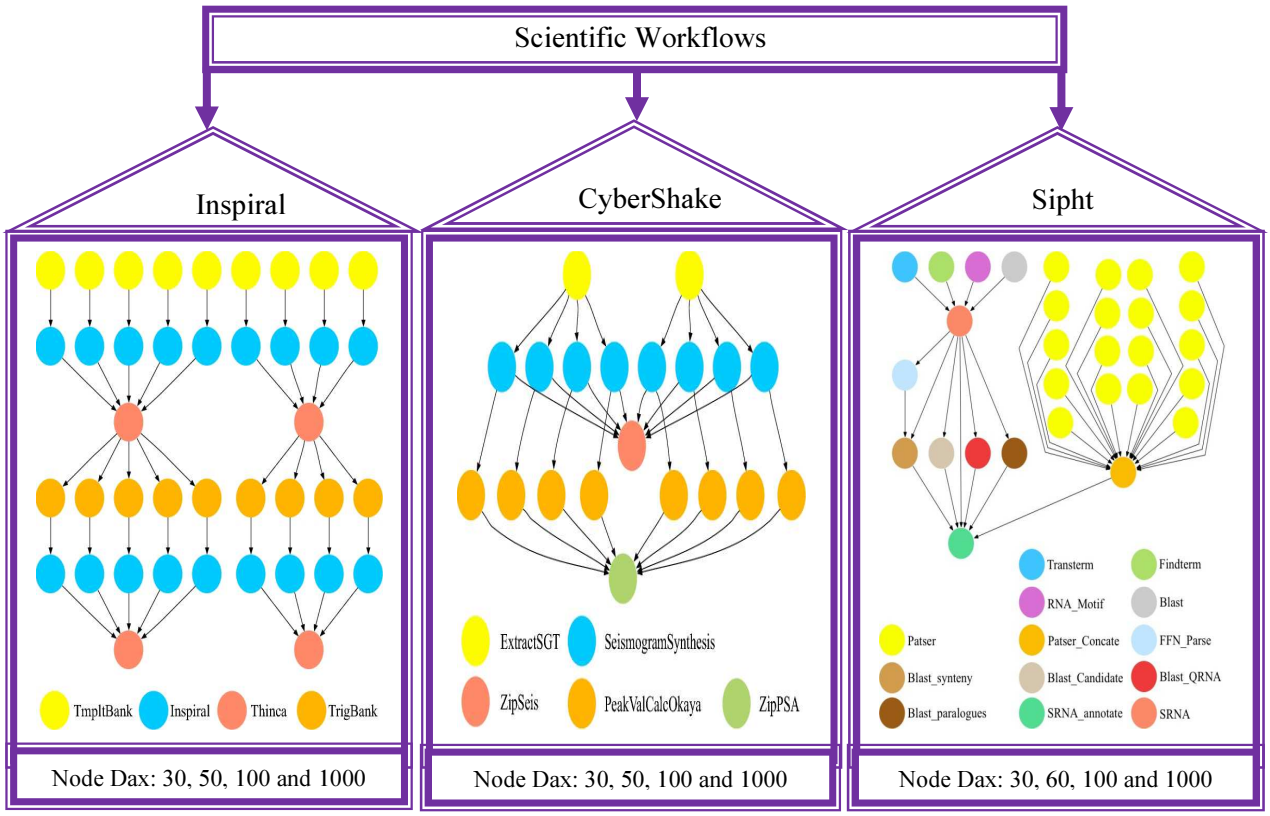


Fig. 3: Scientific workflows

Related Work

The focus of scientific workflow scheduling in cloud computing is to ensure that multiple resources are made available to cloud users at their door steps. This is to ensure that, cloud resources are allocated fairly to cloud users at a reduced cost and time. Scientific workflows have been lengthily used by many researchers as a means of exploring into the parallel distribution of cloud computing applications on distributed systems such as cluster, grids and clouds (Talia, 2013; Gannon *et al*., 2007; Silva *et al*., 2014). Scientific workflow allocation and Scheduling is defined in (Prathibha *et al*., 2014) as a process use to map, manage and execute interdependent workflow cloudlets effectively on a distributed pool of cloud resources to ensure that every available cloud resources is distributed efficiently and fairly on all network notes.

There are many scheduling algorithms that exist in distributed computing, most of which are applicable in the cloud environment, but their performance is questionable in relation to cost, makespan and load balancing, (Thomas *et al*., 2014). These algorithms include

Max-Min, (Mao *et al*., 2014; Kaur and Luthra, 2014), Min-Min (Liu *et al*., 2013), Game Theoretic (Duan *et al*., 2014; Wei *et al*., 2010), FCFS (Marphatia *et al*., 2013), Round Robin (Agarwal and Rastogi, 2014). Though these algorithms have brought some improvement on resource scheduling in cloud computing, but they require more improvement to transform how goods and services are sold and bought in today's competitive environment (Wei *et al*., 2010).

Max-Min is the most commonly used scheduling algorithm that assigns tasks to resources based on the priority of the tasks. It gives high priority to tasks with high completing time by selecting and assigning them to resources first before considering any other task (Brar and Rao, 2015). Max-Min has been considered as the most proactive algorithm in scheduling because of its ability to assign multiple tasks to a resource at a reduced time and cost. Much work has been done by various researches in a quest to improve the performance of Max-Min scheduling algorithm. For example, Li *et al*., (2014) proposed for an improved Max-Min algorithm for cloud computing. The idea behind the improved algorithm was to allow task with maximum execution time to be assigned to resources that produces minimum completion time rather than the traditional Max-Min algorithm that select and assigns task with maximum completion time to resource that produces minimum execution time. Though the results of their study suggest a reduction in makespan, however the study was not

applied in a real cloud computing environment. Also, the result was only limited to makespan analysis while the other parameters were left out.

Similarly, in (Kaur and Ghumman, 2014), hybrid improved Max-Min Ant algorithms for load balancing and makespan optimisation by selecting and assigning tasks based on their execution time was proposed. Assigning tasks based on execution time reduces the load imbalance, but increases the finishing time (makespan).

Dehkordi and Bardsiri (2015), a compounding algorithm built in the conditions of both Min-Min and Sufferage algorithms called Task-Aware Scheduling Algorithm (TASA) was proposed. The aim of the proposed algorithm was to map tasks with odd number on a machine using Min-Min strategy and tasks with even number on machine using Sufferage strategy. The algorithm was able to achieve higher performance by reducing response time of scheduling task to a resource. However, more emphasis was placed on the number of tasks available to be scheduled instead of placing emphasis on both tasks and resources. Moreover, the parameters for benchmarking was only limited to makespan.

Also, Job scheduling approach in cloud computing based on genetic algorithm on load balancing of virtual machines was proposed in (Hu *et al*., 2010). The paper took advantage of generic algorithm and suggested for a historical data and current data state to be computed in advance to determine the solution that will have influence on the current Virtual Machine and the resources to be deployed on every physical node to allow for the selection of any deployment mode that will have least influence on the system to be selected. The authors implemented the algorithm on open-source Virtual Machine management platform called OpenNebula (2010) to analyse the load balancing effect and the migration cost on the systems. The advantages of the algorithm were that; it achieves the best result of balancing load fairly and reduces the migration cost of scheduling. Unfortunately, the delay in migrating tasks to VM which may itself waste the processing bandwidth was not address.

Moreover (Bhoi and Ramanuj, 2013; Santhosh and Manjaiah, 2014) improved on Max-Min algorithm. Bhoi and Ramanuj (2013), an Enhanced Max-Min algorithm was proposed. The mandate of the proposed algorithm was to assign tasks with an average execution time to resources that can best give minimum completion time. Santhosh and Manjaiah (2014) proposed an improved task scheduling algorithm based on Max-Min algorithm which selects and assigns tasks that are greater than average execution time to a resource that gives minimum completion time. Though all these algorithms have improved the performance of the standard Max-Min algorithm, nevertheless, the issues of task priority of scientific workflow and scalability were left out, which is the more reason why this research needs to be conducted. Table 1 displays abbreviations used in this research.

Table 1: Abbreviations and its definitions

Abbreviations	Definition
Expa-max-min	Expanded max-min
CT_{ij}	Completing time *i*
ET_{ij}	Execution time *i* on resource *j*
RT	Ready time
VMs	Virtual machines
MaxExtn	Maximum execution
MinExtn	Minimum execution
MaxClt	Maximum cloudlet
MinClt	Minimum cloudlet

Existing Scheduling Algorithms in Cloud Computing

There are many scheduling algorithms in cloud computing environment, but for the purpose of comparison, we will discuss Max-Min and Min-Min algorithms.

Max-Min Algorithm

This algorithm is designed to schedule task in cloud by considering the completion time of every task in the meta-task. In Max-Min algorithm, priority is placed on tasks that have maximum completion time. The algorithm selects and assigns tasks with maximum completion time to resource that produces its minimum execution time. Max-Min algorithm first computes the completion time (CT_{ij}) = et_{ij} + rt_j; (for each tasks in all VMs) to find tasks with maximum completion time. Then it starts by assigning those tasks with maximum completion time to resources that gives minimum execution time first, (El-Kenawy *et al*., 2012; Kanani and Maniyar, 2015). Figures 4 and 5 present the Pseudo code and flowchart of Max-Min algorithm.

Standard Max-Min Algorithm

For all submitted tasks in the meta-task T_i
For all resource Rj
Calculate completion time (Ct_{ij})=et_{ij}+rt_j; (for each task in all VMs)
Find a task with maximum completing time
Assign the task to a resource that gives minimum completion time
End if
Update the meta-tasks
Update ready time (rt_j) of the selected Rj
Update ct_{ij} for all T_i
End while

Fig. 4: Pseudo code of Max-Min algorithm

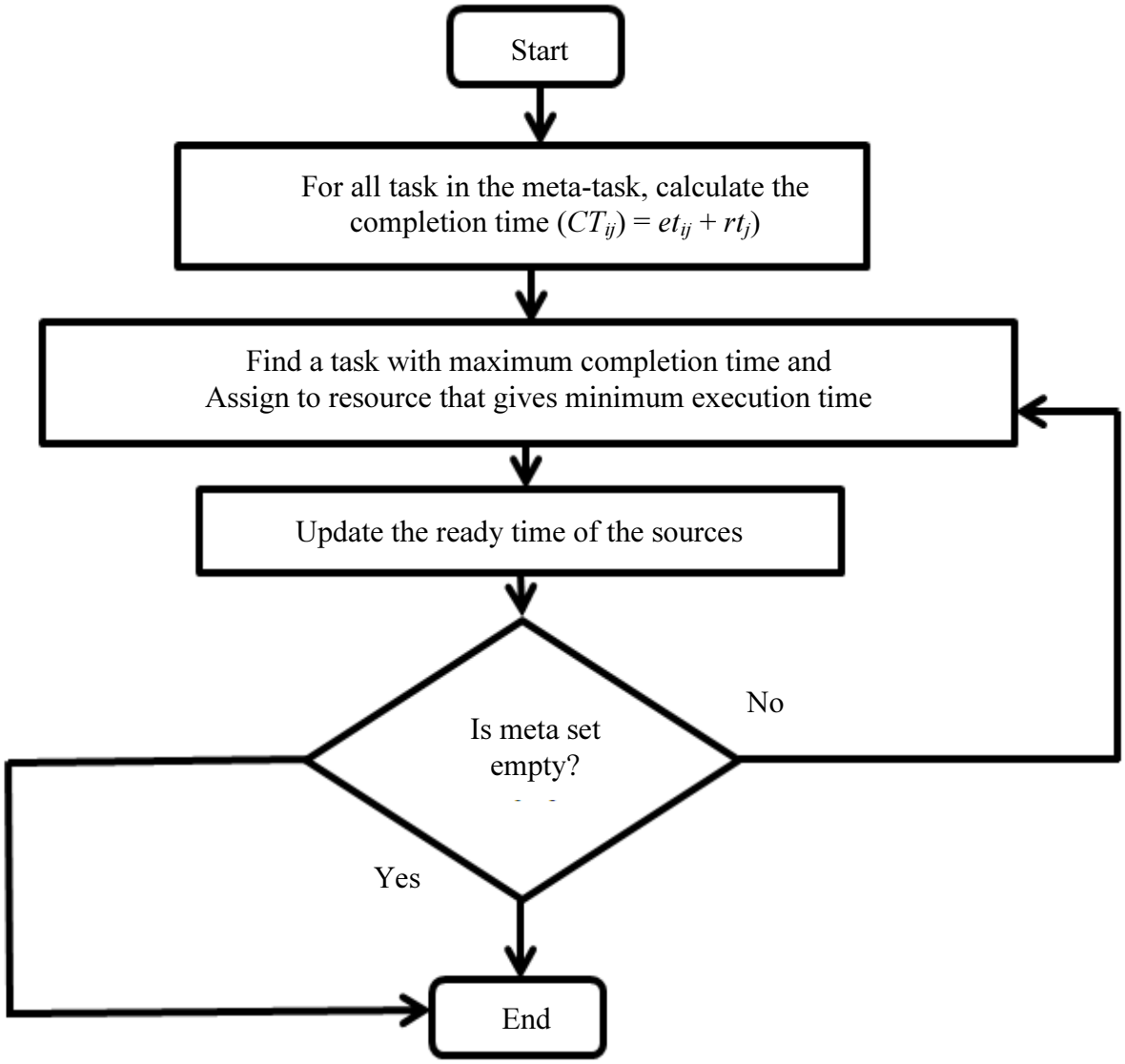


Fig. 5: Flowchart of Max-Min Algorithm

Min-Min Algorithm

Min-Min scheduling algorithm was built on a concept of selecting and assigning task with minimum completion time to resource that has the capability to execute them within a shorter period of time. This algorithm has two steps to consider when scheduling task. The first step is to compute all the available resource on metatask to find the completion time (CT_{ij}) = et_{ij} + r_{tj}; (for each tasks in all VMs) and the second step involve selecting task with minimum expected completion time to a corresponding resource that can execute it faster, (Sharma and Tyagi, 2017; Etminani and Naghibzadeh, 2007; Anousha and Ahmadi, 2013). The pseudo code and the flowchart of Min-Min algorithm are displayed in Fig. 6 and 7.

Standard Min-Min Algorithm

For all submitted tasks in the meta-task T_i
For all resource R_j
Calculate completion time (Ct_{ij})=et_{ij}+rt_j; (for each task in all VMs)
Find a task with minimum completing time
Assign the task to a resource that gives minimum completion time
End if
Update the meta-tasks
Update ready time (rt_j) of the selected R_j
Update ct_{ij} for all T_i
End while

Fig. 6: Pseudo code of standard Min-Min algorithm

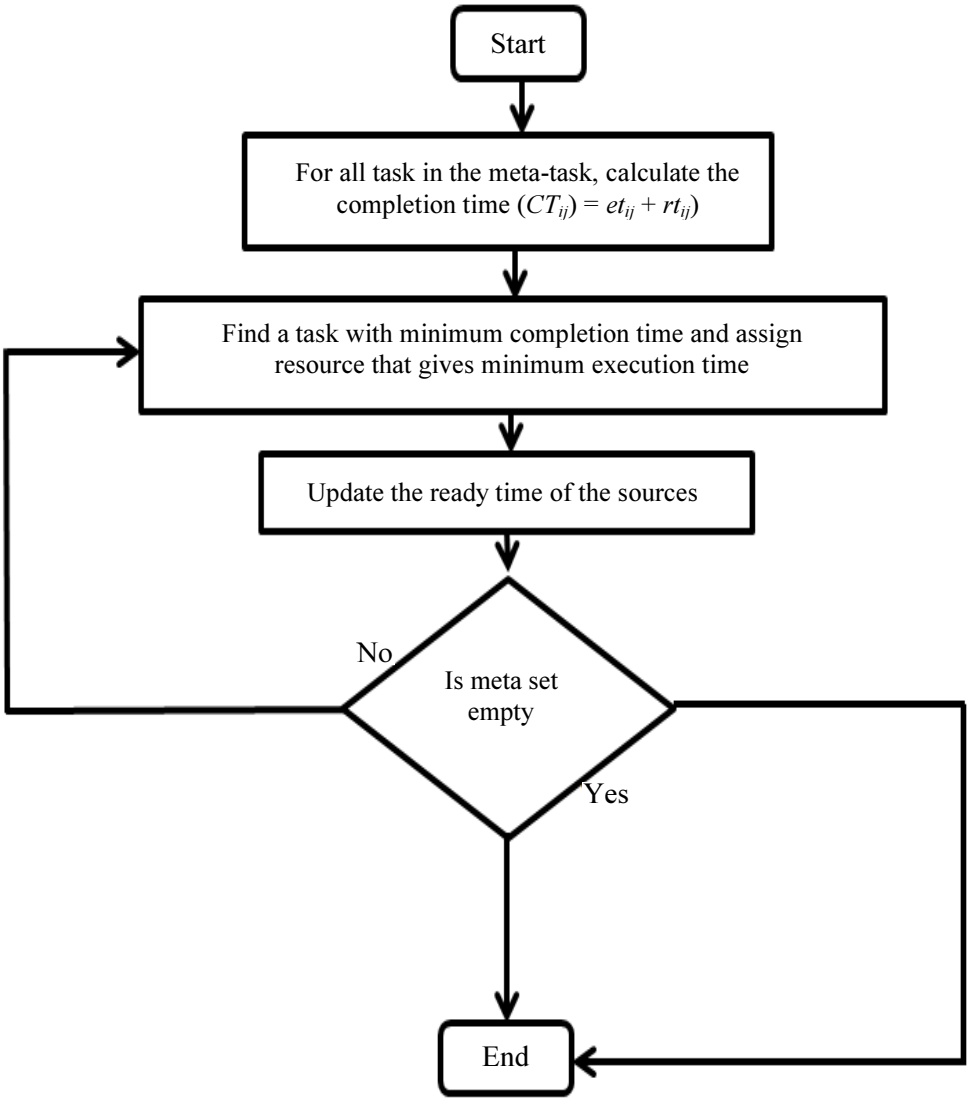


Fig. 7: Flowchart of Min-Min algorithm

Proposed Method (Expa-Max-Min)

The main challenges faced by the standard Max-Min algorithm are the production of high makespan, imbalanced loads and the high priority it gives in favour of cloudlets with maximum completion time in the expense of cloudlets with minimum completion time. To solve these, we proposed a method called Expanded Max-Min (Expa-Max-Min) algorithm. It is an expansion of the standard Max-Min algorithm that reduces the priority given to workflow cloudlets with maximum execution time. Expa-Max-Min allows both workflow cloudlets with maximum and minimum execution time to be selected and assigned to a resource that can execute it within a shorter period of time. Expa-Max-Min algorithm first calculates the completion time of all the cloudlets in the cloudletList to find cloudlets with minimum and maximum execution time, then it sorts and queue the cloudlets in two queues based on their execution times. The first queue will be for cloudlets with maximum execution time and the second queue will be for cloudlets with minimum execution time. First the algorithm will select and assign a cloudlet from maximum cloudlets (MaxClt) queue to a resource that produces its minimum completion time and then followed by the minimum cloudlets (MinClt) queue or else it will continue with the maximum cloudlets. The pseudo code and the flowchart of Exp-Max-Min algorithm are presented in Fig. 8 and 9.

Proposed algorithm (Expa-Max-Min algorithm)

While there are cloudlets in the cloudletList
for all submitted cloudlets,
for all VMs; vmj
calculate the completion time (CT_{ij})=et_{ij}+r_{tj}; of all VMs
Sort the cloudlets with maximum execution time (MaxExtn) and queue it
Sort the cloudlets with minimum execution time (MinExtn) and queue it
Select a cloudlet in the MaxExtn queue and assign to a resource that produce it minimum completion time for execution
Select a cloudlet in the MinExtn queue and assign to a resource that produces it minimum completion time for execution
Else continue with MaxExtn queue /MinExtn queue
Update CloudletList
Update ready time (r_{ij}) of the selected vmR_j
update Ct_{ij} for call c_i
End while

Fig. 8: Pseudo code for Expa-Max-Min algorithm

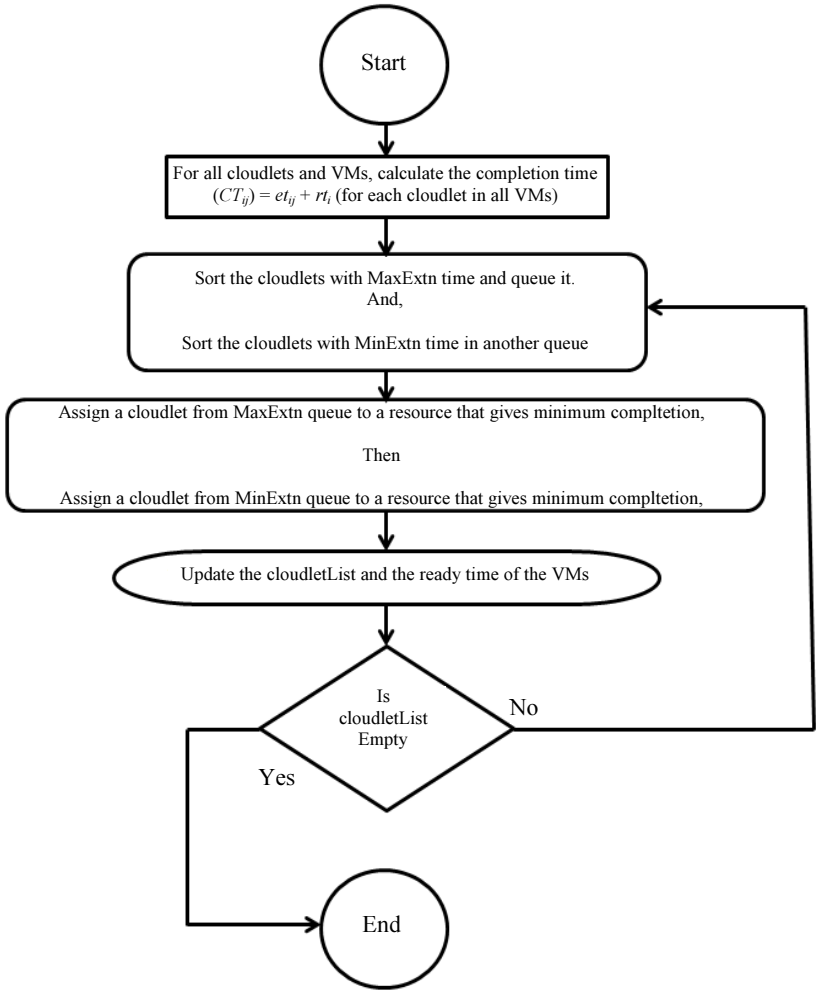


Fig. 9: Flowchart of the proposed algorithm (Expa-Max-Min)

Results and Analysis

Our proposed algorithm, Expa-Max-Min, is implemented using cloudsim, the simulation is run on Intel® core i3, 500GB HDD and 4GB RAM on 64 bit windows 8 operating system. We considered three workflows such as Inspiral, CyberShake and Sipht. The processing speed of each workflow cloudlet is measured in Million Instructions Per Second (MIPS). Table 2 gives a detailed setting of the simulator parameters whereas Table 3-5 gives scenarios involved in using each of the three (3) workflows selected in this research.

Table 2: Simulation parameters Setting

Entities		Number of jobs
Workflows	Inspiral, CyberShake	30, 50, 100, 1000
	Sipht	30, 60, 100, 1000
Data centre		1
Virtual machines		10
Data centre broker		1

Table 3: Inspiral workflows distribution

Scenarios	Inspiral workflow		Data set
1	Inspi-light	Inspiral with light workflows load	30
2	Inspi-median	Inspiral with median workflows load	50
3	Inspi-large	Inspiral with large workflows load	100
4	Inspi-heavy	Inspiral with heavy workflows load	1000

Table 4: CyberShake workflows distribution

Scenarios	CyberShake workflow		Data set
1	Cyber-light	CyberShake with light workflows load	30
2	Cyber-median	CyberShake with median workflows load	50
3	Cyber-large	CyberShake with large workflows load	100
4	Cyber-heavy	CyberShake with heavy workflows load	1000

Table 5: Sipht workflows distribution

Scenarios	Sipht workflow		Data set
1	Sipht-light	Sipht with light workflows load	30
2	Sipht-median	Sipht with median workflows load	60
3	Sipht-large	Sipht with large workflows load	100
4	Sipht-heavy	Sipht with heavy workflows load	1000

Performance Measures

There are various standard performance measurement used in cloud computing to measure, compare and benchmark the performance of every scheduling algorithm. In this research, we used three performance measures such as:

Load Balancing Metrics

Load balancing is the process by which cloudlets are distributed evenly in the cloud computer environment to avoid traffic and allow free flow of cloud resource on all network nodes. Maipan-uku *et al*. (2016; Cao *et al*., 2005), load balancing is calculated as in Equations 1 and 2:

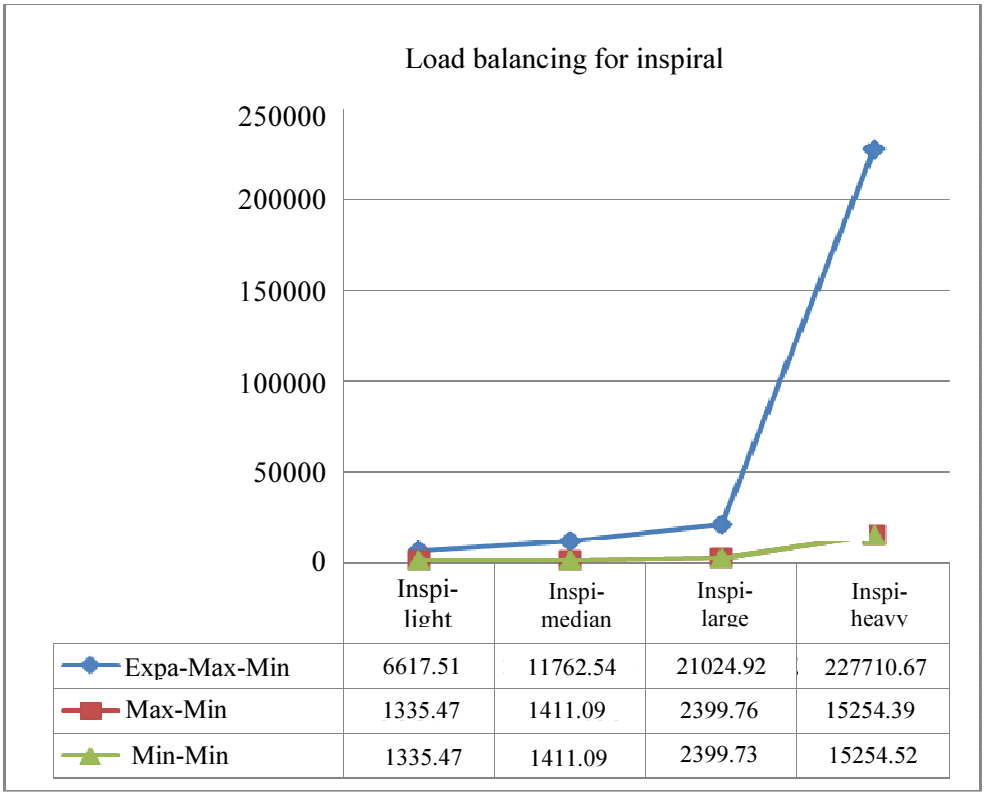
$$\beta = \left(1 - \frac{d}{Avgru}\right) * 100 \quad (1)$$

The deviation (d) is defined as:

$$d = \sqrt{\frac{\sum_{i=1}^{n} (Avgru - ru_i)^2}{n}} \quad (2)$$

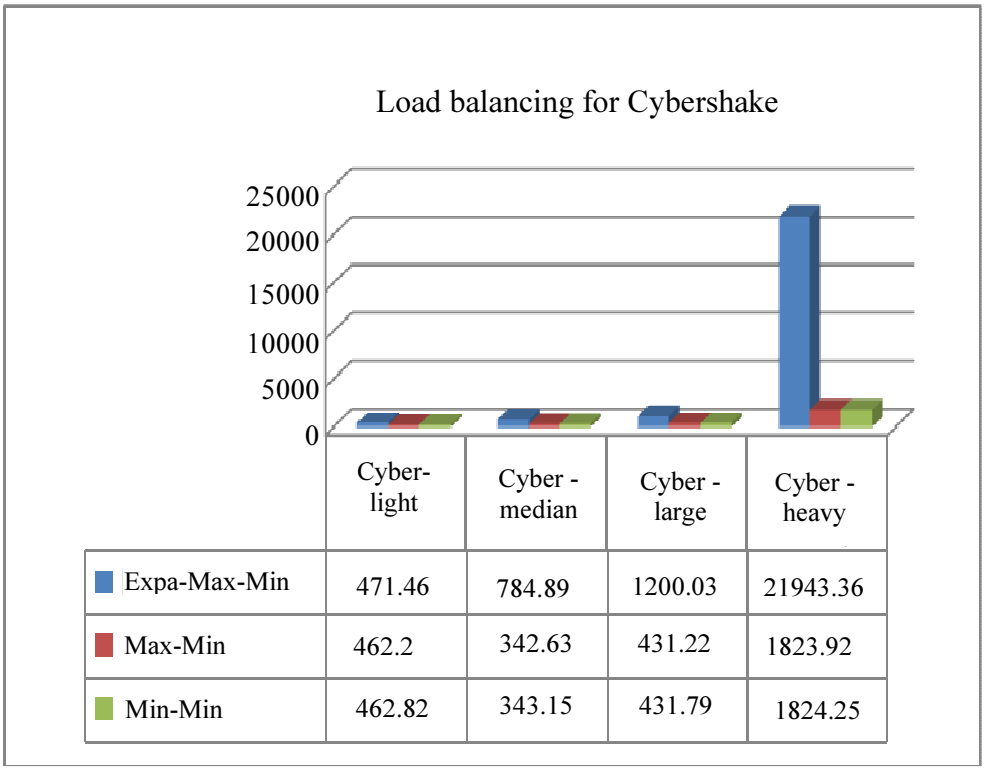
The results are analysed based on the four scenarios given in Table 3, 4 and 5, with different algorithms like Expa-Max-Min, Max-Min and Min-Min. Figure 10 presented results on the performance of the evaluated proposed Expa-Max-Min, Max-Min and Min-Min algorithms with regards to load balancing using Inspiral. The proposed Expa-Max-Min algorithm gives remarkable results in terms of load balancing efficiency at all the four scenarios as compared to Max-Min and Min-Min. This is because Expa-Max-Min was able to do away with cloudlets priority by allowing both cloudlets with maximum and minimum execution time to be scheduled concurrently.

Figure 11 and 12 also illustrate results of load balancing using CyberShake and Sipht for each of the evaluated algorithms. Based on the implemented results, it has been observed that, at every execution of cloudlet, the load balancing of the proposed algorithm remains more balanced than the standard Max-Min and Min-Min algorithms, this is because the proposed algorithm was able to assign all the cloudlets to the correct resources properly. Moreover, the comparison also shows that, Min-Min algorithm has produced good values of load balancing in Fig. 11 and 12 at all the four scenarios than Max-Min algorithm.



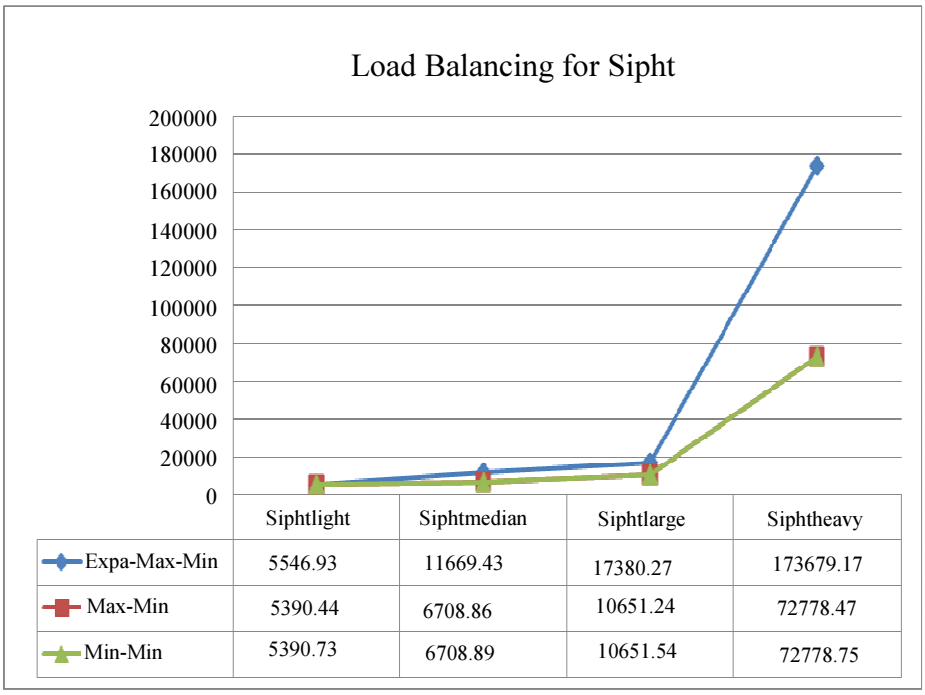
	Inspi-light	Inspi-median	Inspi-large	Inspi-heavy
Expa-Max-Min	6617.51	11762.54	21024.92	227710.67
Max-Min	1335.47	1411.09	2399.76	15254.39
Min-Min	1335.47	1411.09	2399.73	15254.52

Fig. 10: Load balancing of different algorithms using Inspiral



	Cyber-light	Cyber - median	Cyber - large	Cyber - heavy
Expa-Max-Min	471.46	784.89	1200.03	21943.36
Max-Min	462.2	342.63	431.22	1823.92
Min-Min	462.82	343.15	431.79	1824.25

Fig. 11: Load balancing of different algorithms using CyberShake



	Siphtlight	Siphtmedian	Siphtlarge	Siphtheavy
Expa-Max-Min	5546.93	11669.43	17380.27	173679.17
Max-Min	5390.44	6708.86	10651.24	72778.47
Min-Min	5390.73	6708.89	10651.54	72778.75

Fig. 12: Load balancing of different algorithms using Sipht

Makespan Metrics

Makespan is the total amount of time that cloudlets spend during execution.

Figures 13-15 present results in respect of makespan performances produced by different algorithms using different workflows such as Inspiral, CyberShake and Sipht respectively. Results in Fig. 13 shows that Expa-Max-Min algorithm is outperforming the standard Max-Min and Min-Min by producing lower values of makespan in every execution of cloudlet at all the four scenarios. In Fig. 14 and 15, Expa-Max-Min was able to schedule all the resources on all the available Virtual Machines at a reduced makespan in all the scenarios except at scenario 4 of Sipht (Sipht-heavy) that Min-Min algorithm performed slightly better than the proposed algorithm, this is because Min-Min was able to assign all the cloudlets at that stage properly.

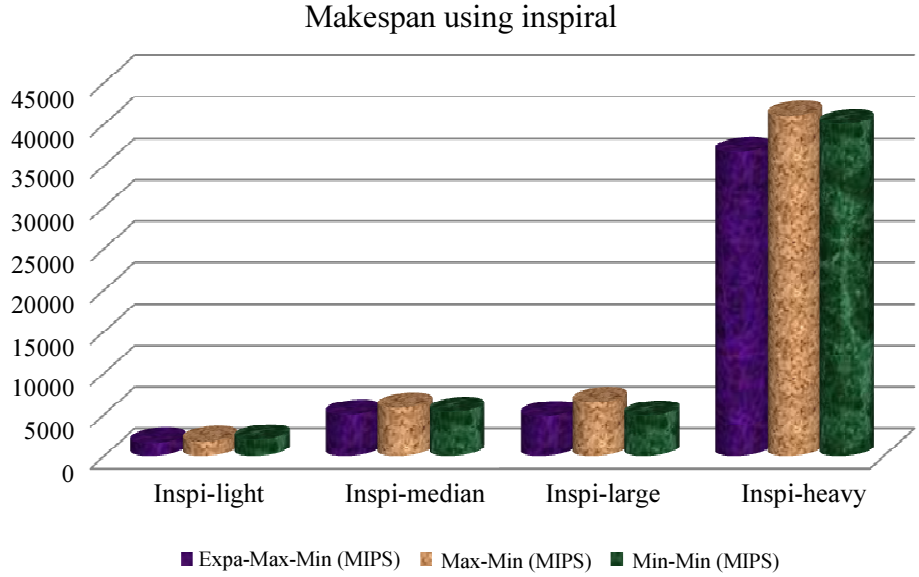


Fig. 13: Makespan of different algorithms using inspiral

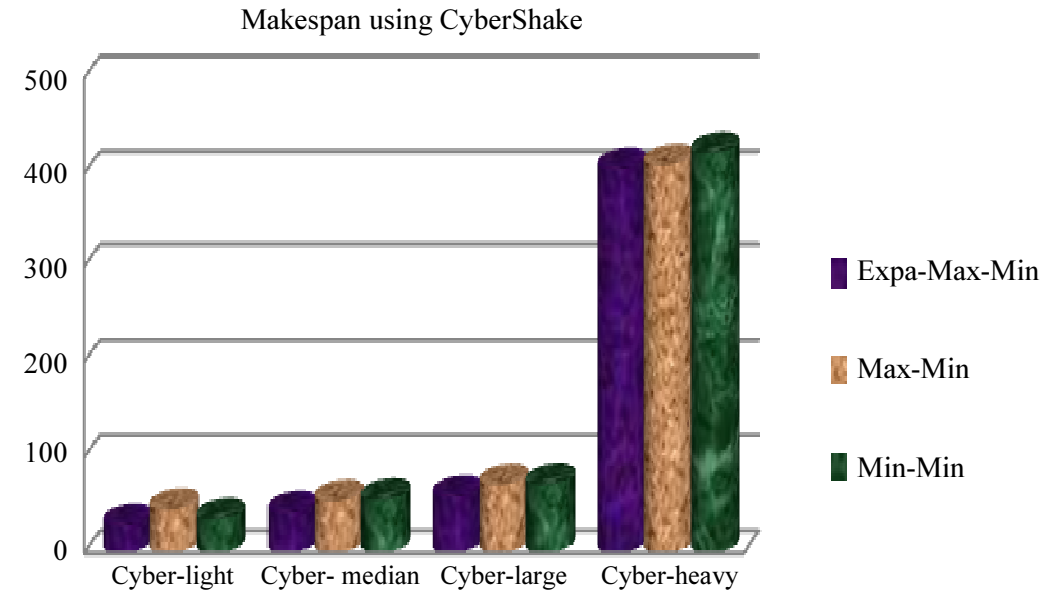


Fig. 14: Makespan of different algorithms using CyberShake

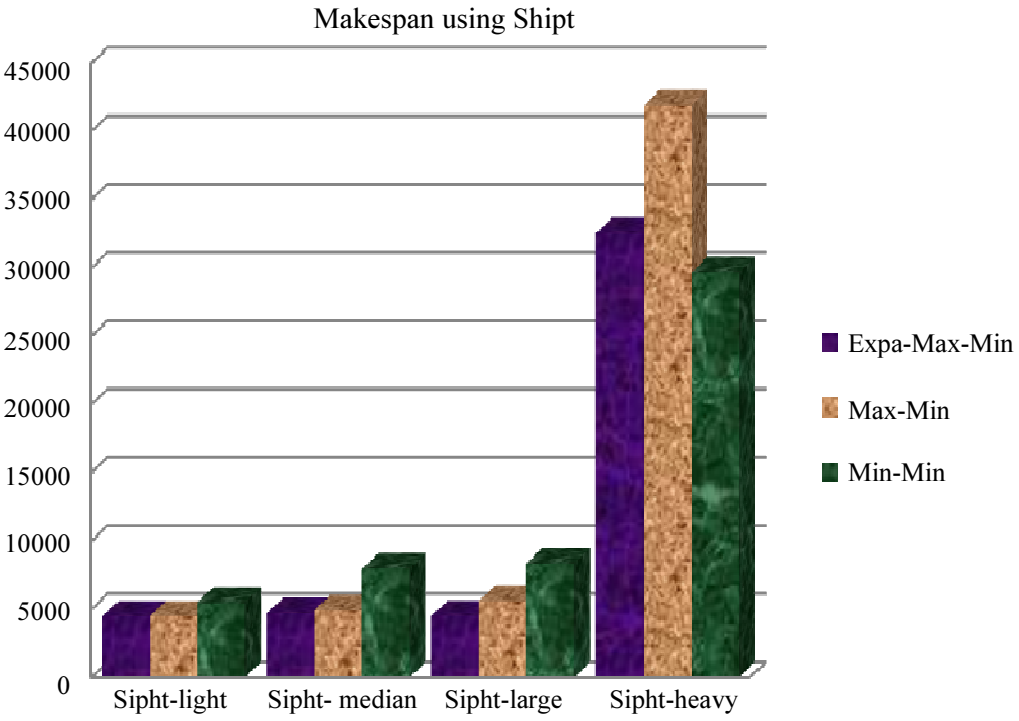


Fig. 15: Makespan of different algorithms using Sipht

Cost Metrics

Cost is referred to as the amount needed to be spent on scheduling a cloudlet in a cloud computing environment. The cost takes in to consideration of the Job size, cost of network, allocation size of the job, cost of executing a cloudlet, bandwidth, resource scheduling speed and cost of a data center as simplified in Table 6. In (Kamarajapandian and Chitra, 2016), the cost of scheduling in cloud is calculating using Equation 3 to 7:

i. Network Transmission Time

$$NTT_{ij} = \frac{J_S}{B_{Wi}} \quad (3)$$

ii. Expected Time to Complete

$$ETC_{ij} = \frac{J_S(MI)}{RSS_i\ (MIPS)} \quad (4)$$

iii. Expected Waiting Time

$$EWT_{ij} = \sum \frac{A_{si}}{RSS_i} \quad (5)$$

iv. Overall Cloudlet Completion Time

$$OCCT = NTT + BTC + BWT \quad (6)$$

v. Overall Cost for Cloudlet Completion

$$OCCC_{ij=}\left(CN_i \times NTT_{ij}\right)+\left(CE_{Ci} \times ETC_{ij}\right)+\left(CDC_i \times EWT_{ij}\right) \quad (7)$$

Table 6: Cost computation variables description

Variable	Description	Inspiral	CyberShake	Sipht
J_s	Job Size	1180	1180	1190
CN_i	Cost of network			
A_{si}	Allocated Size of job *i*			
CE_{ci}	Cost of execution cloudlet *i*			
B_{wi}	Bandwidth			
RSS_i	Resource scheduling speed			
CDC_i	Cost of Data Center			

Figures 16, 17 and 18 depicts the performance analysis of average executional cost with respect to the job size of 1180 workflows of Inspiral and CyberShake and 1190 workflows of Sipht. The results in the figures shows that Expa-Max-Min (proposed algorithm) has proven beyond reasonable doubt that it is more cost effective in scheduling cloudlets in all the three workflows used as compared to Max-Min and Min-Min algorithms, this is because Expa-Max-Min is able to boost up cloud scheduling processes by simultaneously selecting and assigning cloudlets with both maximum and minimum execution time concurrently at a reduced cost. On the other hand, Max-Min, one of the benchmarked algorithms has also performed better than the standard Min-Min algorithm at almost all the three workflows used except at CyberShake that Min-Min performed better than Max-Min algorithm.

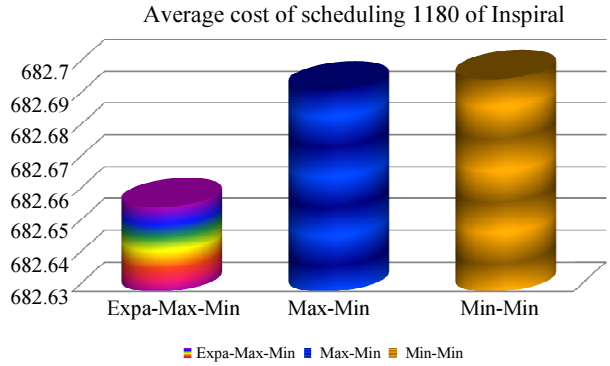


Fig. 16: Comparison of average executional cost of different algorithms using Inspiral

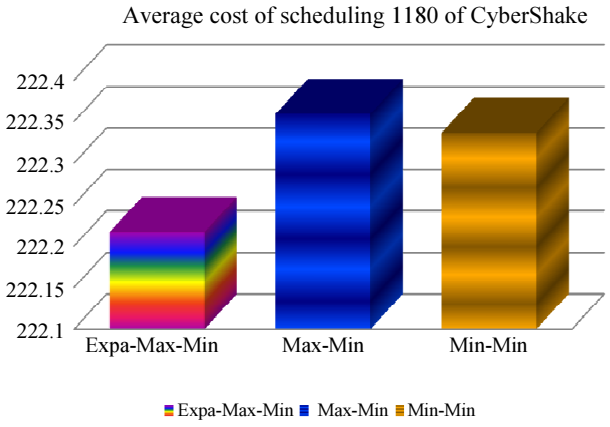


Fig. 17: Comparison of average executional cost of different algorithms using CyberShake

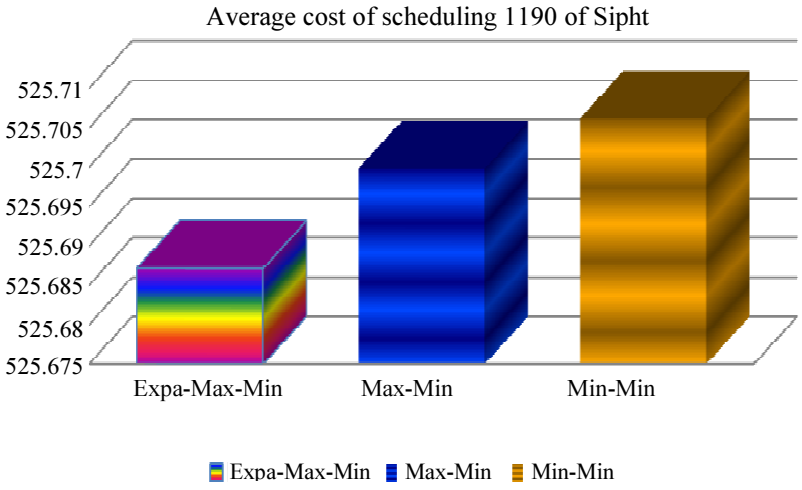


Fig. 18: Comparison of average executional Cost of Different Algorithms using Sipht

Conclusion

In this study, we identified three major challenges in the standard Max-Min algorithm, which includes high cost of scheduling workflows, high makespan and high priority given to workflows with maximum Execution time. In order to tackle these problems, we presented a state of the arc algorithm known as an Expanded Max-Min (Expa-Max-Min) Algorithm.

Our algorithm (Expa-Max-Min) was designed and simulated on cloud simulation platform called cloudsim for the purpose of minimising the makespan value, cost of scheduling scientific workflow and to balance load effectively in a dynamic cloud computing environment.

The proposed algorithm is able to produce good quality solutions for all the cases, schedule both workflow cloudlets with maximum and minimum execution time concurrently without giving priority to workflow cloudlets with maximum execution time as always in the case of the standard Max-Min algorithm. To further prove the effectiveness of the proposed Expa-Max-Min algorithm, we experimented the results on a cloudsim simulator. The simulation results shows that, our proposed algorithm has the ability to balance load fairly, producing better makespan and average executional cost when compared with the standard Max-Min and Min-Min algorithms.

Acknowledgement

This research was partially supported by the Universiti Putra Malaysia [Grant No: GP/2017/9588500].

Author's Contribution

James Kok Konjaang: Contributed in all experimentation, simulation design, data-analysis and writing part of the article.

Fahrul Hakim Ayob: Contributed in the organization of the study, methodology and writing part of the article.

Abdullah Muhammed: Contributed in the coding of the algorithm, experimentation, result presentation and the designing of the research plan.

Ethics

This research is our original contribution and no ethical issues are involved.

References

Agarwal, A. and R. Rastogi, 2014. Round robin approach for vm load balancing algorithm in cloud computing environment. Int. J. Adv. Res. Comput. Sci. Softw. Eng., 4: 34-39.

Anousha, S. and M. Ahmadi, 2013. An improved min-min task scheduling algorithm in grid computing. Proceedings of the International Conference on Grid and Pervasive Computing, (GPC'13), Springer-Verlag Berlin Heidelberg, pp: 103-113. DOI: 10.1007/978-3-642-38027-3_11

Berriman, G.B., E. Deelman, J.C. Good, J.C. Jacob and D.S. Katz *et al*., 2004. Montage: A grid-enabled engine for delivering custom science-grade mosaics on demand. SPIE Astronomical Telescopes + Instrumentation, International Society for Optics and Photonics, pp: 221-232.

Bhoi, U. and P.N. Ramanuj, 2013. Enhanced max-min task scheduling algorithm in cloud computing. Int. J. Applic. Innovat. Eng. Manage., 2: 259-264.

Brar, S.S. and S. Rao, 2015. Optimizing workflow scheduling using max-min algorithm in cloud environment. Int. J. Comput. Applic., 124: 44-49.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A. De Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Prac. Experience, 41: 23-50. DOI: 10.1002/spe.995

Cao, J., D.P. Spooner, S.A. Jarvis and G.R. Nudd, 2005. Grid load balancing using intelligent agents. Future Generat. Comput. Syst., 21: 135-149. DOI: 10.1016/j.future.2004.09.032

Cpedia, S.C.E., 2016. CyberShake. https://scec.usc.edu/scecpedia/CyberShake

Dehkordi, S.T. and V.K. Bardsiri, 2015. TASA: A new task scheduling algorithm in cloud computing. J. Advan. Comput. Eng. Technol., 1: 26-32.

Duan, R., R. Prodan and X. Li, 2014. Multi-objective game theoretic scheduling of bag of-tasks workflows on hybrid clouds. Cloud Comput., IEEE Trans., 2: 29-42. DOI: 10.1109/TCC.2014.2303077

El-Kenawy, E.S.T., A.I. El-Desoky and M.F. Al-Rahamawy, 2012. Extended max-min scheduling using petri net and load balancing. Int. J. Soft Comput. Eng.

Etminani, K. and M. Naghibzadeh, 2007. A min-min max-min selective algorihtm for grid task scheduling. Proceedings of the 3rd International Conference in Central Asia, Sep. 26-28, IEEE Xplore Press, Tashkent, Uzbekistan. DOI: 10.1109/CANET.2007.4401694

Gannon, D., E. Deelman, I. Taylor and M. Shields, 2007. Workflows in e-Science.

Graves, R., T.H. Jordan, S. Callaghan, E. Deelman and E. Field *et al*., 2011. CyberShake: A physics-based seismic hazard model for southern California. Pure Geophys., 168: 367-381. DOI: 10.1007/s00024-010-0161-6

Hamdaqa, M. and L. Tahvildari, 2012. Cloud computing covered: A research landscape. Adv. Comput., 86: 41-85. DOI: 10.1016/B978-0-12-396535-6.00002-8

Hu, J., J. Gu, G. Sun and T. Zhao, 2010. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming, Dec. 18-20, IEEE Computer Society, Washington, DC, USA, pp: 89-96. DOI: 10.1109/PAAP.2010.65

Kamarajapandian, P. and P. Chitra, 2016. HJSA: A hierarchical job scheduling algorithm for cost optimization in cloud computing environment. Econom. Computat. Econom. Cybernet. Stud. Res., 50: 281-296.

Kanani, B. and B. Maniyar, 2015. Review on max-min task scheduling algorithm for cloud computing. J. Emerg. Technol. Innovat. Res., 2: 782-784.

Kaur, R. and N. Ghumman, 2014. Hybrid improved max min ant algorithm for load balancing in cloud. Proceedings of the International Conference on Communication, Computing and Systems (CCS'14), pp: 188-191.

Kaur, R. and P. Luthra, 2014. Load balancing in cloud system using max-min min and min algorithm. Proceedings of the National Conference on Emerging Trends in Computer Technology, (TCT'14), pp: 31-34.

Li, X., Y. Mao, X. Xiao and Y. Zhuang, 2014. An improved max-min task-scheduling algorithm for elastic cloud. Proceedings of the International Symposium on Computer, Consumer and Control, Jun. 10-12, IEEE Xplore Press, Taichung, Taiwan, pp: 340-343 DOI: 10.1109/IS3C.2014.95

Liu, G., J. Li and J. Xu, 2013. An improved min-min algorithm in cloud computing. Proceedings of the International Conference of Modern Computer Science and Applications, (CSA'13), Springer, Berlin, Heidelberg, pp: 47-52. DOI: 10.1007/978-3-642-33030-8_8

Livny, J., 2016. SIPHT. https://pegasus.isi.edu/application-showcase/sipht/

Maipan-uku, J.Y., J.K. Konjaang and A.I. Baba, 2016. New batch mode scheduling strategy for grid computing system. Int. J. Eng. Technol., 8: 1314-1323.

Mao, Y., X. Chen and X. Li, 2014. Max-Min task scheduling algorithm for load balance in cloud computing. Proceedings of the International Conference on Computer Science and Information Technology, Sep. 21-23, Kunming, China, pp: 457-465. DOI: 10.1007/978-81-322-1759-6_53

Marphatia, A., A. Muhnot, T. Sachdeva, E. Shukla and L. Kurup, 2013. Optimization of FCFS based resource provisioning algorithm for cloud computing.

Mehta and Gideon, 2014. Workflow examples. http://www.workflowsim.org/workflows.html

Mell, P. and T. Grance, 2011. The NIST definition of cloud computing. Technical Report, SP 800-145, National Institute of Standards and Technology, Gaithersburg, MD, United States.

OpenNebula, 2010. OpenNebula Software. http://www.opennebula.org

Prathibha, S., B. Latha and G. Sumathi, 2014. Monitoring the performance analysis of executing workflow applications with different resource types in a cloud environment. Proceedings of the 1st International Symposium on Big Data and Cloud Computing Challenges, (CCC'14), pp: 27-28.

Samarsinh, P.J. and P.R. Deshpande, 2014. Load balancing in cloud computing. Int. J. Sci. Res., 3: 2282-2285.

Santhosh, B. and D. Manjaiah, 2014. An improved task scheduling algorithm based on max-min for cloud computing. Proceedings of the International Conference on Advances in Computer and Communication Engineering, Apr. 21-22, Department of CSE and ISE, Vemana Institute of Technology, Bengaluru, India, pp: 84-88.

Saraswathi, A., Y. Kalaashri and S. Padmavathi, 2015. Dynamic resource allocation scheme in cloud computing. Proc. Comput. Sci., 47: 30-36. DOI: 10.1016/j.procs.2015.03.180

Sharma, N. and S. Tyagi, 2017. A comparative analysis of min-min and max-min algorithms based on the makespan parameter. Int. J. Adv. Res. Comput. Sci., 8: 1038-1041.

Sharma, S. and D. Pariha, 2014. A review on resource allocation in cloud computing. Int. J. Adv. Res. Ideas Innovat. Technol., 1: 1-7.

Silva, R.F.D., W. Chen, G. Juve, K. Vahi and E. Deelman, 2014. Community resources for enabling research in distributed scientific workflows. Proceedings of the 10th International Conference on e-Science, Oct. 20-24, IEEE Xplore Press, Washington, DC, USA, pp: 177-184. DOI: 10.1109/eScience.2014.4

Talia, D., 2013. Workflow systems for science: Concepts and tools. ISRN Softw. Eng., 2013: 1-15. DOI: 10.1155/2013/404525

Thomas, A.G. Krishnalal and V.P.J. Raj, 2014. Credit based scheduling algorithm in cloud computing environment. Proceedings of the International Conference on Information and Communication Technologies, Dec. 3-5, Bolgatty, Island Resort, Kochi, India.

Wei, G., A.V. Vasilakos, Y. Zheng and N. Xiong, 2010. A game-theoretic method of fair resource allocation for cloud computing services. J. Supercomput., 54: 252-269. DOI: 10.1007/s11227-009-0318-1