International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

An improved PBMM Algorithm by Reduce Cloud Data Traffic and Transfer Costs in Cloud of Things

Abdulrahman Mohammed Hussein Obaid, Santyyosh Kumar Pani, Prasant Kumar Pattnaik

Abstract*: Cloud computing has been widely studied over the recent years. Researchers have developed different algorithms for improving the performance and minimizing the cost. This paper proposes a new algorithm to improve and enhance the PBMM algorithm (Priority Based on Min-Min Algorithm). The proposed algorithm works with the aid of one of the Cloud of Things (CoT) services; this service is Sensing and Actuation as a Service (SAaaS). The proposed Algorithm works on third-party broker. However, it has two-phase: the first phase is Sensing: in this phase, the sensor observes the throughput for all tasks and compares it with the link capacity. The Second phase is Actuation: depending on the comparison in the first phase, the priority of all the takes will change depending on the link capacity, all tasks will have the same priority if the throughput is low (Green throughput). All tasks will have two priority levels (high, low) if the throughput medium (Yellow throughput), and finally, if the throughput is high (red throughput) all tasks will have a default priority which assigned to them when they are created. However, the efficiency and performance of the IPBMM algorithm depend on the capacity of the link. If capacity is high (traffic in the network is high), the performance is very good and the costly, but if the capacity is medium (traffic in the network is medium), the performance is good as well as the cost. While if the capacity is low (traffic in the network is low), the performance is good and the cost is free. Therefore, the outcomes of the proposed algorithm experiment given 30% better results than the PBMM algorithm and other state-of-the-art algorithms.

Keywords: Cloud Computing, Cloud of Things, Cloud Task Scheduling, PBMM algorithm, User-Priority.

I. INTRODUCTION

This section discusses the different characteristics and challenges and gives an overview of the cloud of things. In particular, the main concepts and terminology that used in this paper.

Revised Manuscript Received on November 30, 2019.
* Correspondence Author
Abdulrahman Mohammed Hussein Obaid*, School of Computer Science and Engineering in KIIT University, India, E-mail: obaid.eng@gmail.com, 1681108@kiit.ac.in
Santosh Kumar Pani, School of Computer Science and Engineering in KIIT University, India, E-mail: spanifcs@kiit.ac.in
Prasant Kumar Pattnaik, School of Computer Science and Engineering in KIIT University, India, E-mail: patnaikprasantfcs@kiit.ac.in

A. Background

Due to the scientific and technical development in the field of computer, especially in cloud computing and internet of things (IoT), the service providers became more competitive and offering many services. The quality of these services depends on the payments that are paid by customers "Pay-as-you-use" [13, 1]. By integrating both of cloud computing and internet of things (IoT) a new field emerged knows as Cloud of Things (CoT) [12]. Recently this field becomes popular because of the possibilities of processing a massive amount of data that came from the internet of things, and by giving these data a privilege of access to virtual resources and storage in Cloud. Cloud of Things (CoT) is a set of virtual computing resources such as networking, databases, storage, and software etc, over the Internet, accessed by smart devices such as smart watches, smart phones, barometric pressure sensor, etc [12]. Cloud of Things (CoT) is providing a set of smart Services such as Data Base as a Service (DBaaS), Sensing as a Service (SaaS), Sensor Event as a Service (SEaaS), Data as a Service (DaaS), and Sensing and Actuation as a Service (SAaaS) etc [12, 4].

There are different challenges related to the CoT as showing in figure 1, one of these challenges is performance and cost. In Cloud of Things, high Performance can be Costly because of the amount of the massive data that need to process and store in the same time.

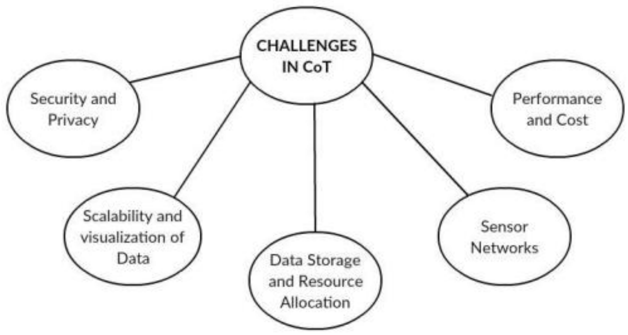


Fig. 1. Challenges in Cloud of Things

To minimize the cost with keeping the performance high there are many algorithms have been proposed especially in transfer and scheduling data from the customer to cloud, the responsible party for sending and scheduling data in the cloud is the third-party broker. The third-party broker is an entity used for negotiates and manages the data transfer from customers to cloud storage [13, 9].

There are a set of factors that can affect data transfer costs include: the type of Service, Data transfer outside Region, Service provider, quality of Service, and amount of data sent and received.

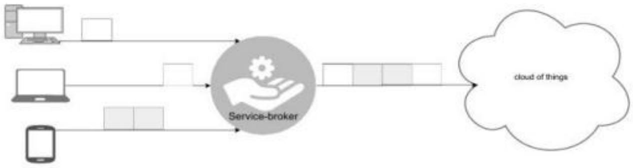


Fig. 2.Third-party broker

B. Terminology and Abbreviations

Table I: Terminology and Abbreviations

Term	Description
Capacity of Link	It is the maximum amount of data that moved successfully from one place to another over a network path in a given time period.
Data Transfer Time	It is taken time to transporter some data from one point to another. It is equivalent to the available bandwidth divided by the size of data.
Data Transfer Costs	It is a fee to transfer data from the customer to cloud storage through a third-party broker
Data Transmission Latency	It is the delay time that takes for the amount of data to get to, its destination across the network and back again (round trip delay); it is measured in milliseconds (ms).
PBMM	Priority Based on Min-Min Algorithm
Response Time	It is taken time between sending the request and receiving a response from an Internet application defined.
Service-Broker	It is a third-party that acts as an intermediary between the client (users) and service provider. Service-broker acts as an intermediary during negotiations between two or more parties.
Throughput	It is the amount of data that moved successfully from one place to another over a network path in a given time period. It is measured in bits per second (bps)

The remaining parts of this paper are organized as follows: in Section two, discusses the related and previous work which belongs to the same area of research. In Section three, presents the problem statement that makes us propose this paper. In Section four, the Proposed Architecture, we improved the PBMM algorithm by adding some techniques. Section five, performance Evaluation, evaluates the performance by using a simulation tool named cloudsim platform to prove the degree of efficiency of the proposed algorithm with PBMM algorithm and to present the results of the comparison by using a cloudsim platform. Finally, we conclude this paper in Section six.

II. RELATED WORKS

Recently, the Cloud of Things becomes popular not only because it deals with a large amount of information that came from millions of users and store in the cloud, but what we able to do with that information. There are many article papers have been published in cloud of things, by considering and reviewing a large number of these article papers we got the motivation behind this paper. Some of these papers as following:

Mohammed, Abdulrahman, et al. [12] introduced the "Towards on Cloud of Things: Survey, Challenges, Open Research Issues, and Tools" the authors discussed the concept of Cloud of Things in detail, and open research issues by exploring and focusing on the challenges in Cloud of Things, also discussed the different tools that can be used with Cloud of Things. The paper gives good ideas for researchers to find open research in the Cloud of Things.

Botta, Alessio, et al [5] In this paper, the authors focused on the integration of Cloud and IoT, and they called it the CloudIoT paradigm. This study offered literature surveyed on the integration of IoT and Cloud on their main properties, features, underlying technologies, challenges, and open research issues.

Abdelwahab, Sherif, et al. [2] they proposed cloud agents algorithmic, first they proposed a distributed sensing resource discovery algorithm and they proposed RADV distributed virtualization algorithm. These algorithms work efficiently by deploys Sensing as a Service on Cloud of Things which considered as a global architecture by using the computing platforms as cloud agents for discovering and virtualizing sensing resources of IoT devices. In this paper, the authors have described the design objectives and cloud agents technical challenges for virtualization and sensing resources discovery on virtual sensor networks and IoT devices that can dispatch to cloud users. They proposed a solution for sensing resource based on a gossip policy. Analysis and simulation showed the potential of RADV to reduce the communication overhead, employs devices virtualization with maximal benefit and low complexity.

Shanthan, BJ Hubert, et al. [18] proposed several scheduling algorithms to use for solved the resource management problem in Cloud-based IoT Environment. The main goal of this Proposed is to maximize utilization of resources with reduced cost estimation and less energy consumption. The authors classified the scheduling algorithm into two types: consideration of the request class and non-consideration of request class. The non-consideration of request classified into dynamic centralized, static centralized, distributed immediate, batch immediate, co-operative, and non-cooperative, while the consideration of request the classified into preemptive and non-preemptive. The non-preemptive is more suitable for IoT cloud because it has a low delay.

Zeng, L. et al. [20]. Provided the Security-aware and Budget-aware algorithm known as SABA algorithm. The authors designed this algorithm to schedule workflows in a multi-cloud environment. SABA algorithm has three main phases. First, the prioritization and clustering stage, in this phase the tasks and data are assigned to specific data centers based on the workflow of data sets. Then compute I/O costs on a baseline VM type. Based on this, priorities are assigned to tasks. The second phase is assigned tasks to VMs based on a cost ratio.

Finally, data is moved dynamically on location that is ready for execution of the process. The outcomes of the experiment indicate that the SABA algorithm has higher costs than expected when using the algorithm on a real cloud environment.

Abrishami, S. et al. [3]. Proposed two workflow scheduling algorithms for the cloud environment the first algorithm is called IaaS Cloud Partial Critical Paths or IC-PCP. The second algorithm which is called IaaS Cloud Partial Critical Paths with deadline distribution or IC-PCPD2. Both algorithms are suitable options for scheduling large workflows and the main difference between them that IC-PCP algorithm schedules the workflow in one phase, and IC-PCPD2 algorithm schedules the workflow in two-phase, the first phase is give all tasks subdeadline by distributed the deadline on the workflow tasks. Then each task scheduled based on its subdeadline. Depending in their work, the experiments and results show that IC-PCP algorithm outperforms of IC-PCPD2 algorithm as well as the computation time of the IC-PCP algorithm is very low.

Calheiros, R. N. et al. [7]. The authors proposed EIPR algorithm that considers the behavior of Cloud resources for tasks scheduling, and also by using the idle time of provisioning resources to increase the chance of meeting application deadlines. The EIPR algorithm has three distinct steps; the first step is combined task scheduling and provision of cloud resources. Second step data transfer-aware provisioning adjusts and final step task replication. The objective behind this algorithm is reducing the impact of poor performance of cloud resources. The simulation experiments show that the EIPR algorithm reduces the total execution time of applications and increases the probability of deadlines met with the use of task replication.

Byun, E. K., et al. [6]. Introduced Partitioned Balanced Time Scheduling or known as PBTS algorithm. This algorithm proposed architecture for the automatic execution of workflow on dynamically computing resources. The main objective of this algorithm is that PBTS fill the gap between the resource provisioning environments and workflow management system. And fit both parallel application models and elastic resource provisioning models. To minimize the resource cost, the PBTS algorithm estimate the resource capacity per time partition also the host requirements of tasks and time schedule, that lead to satisfying the deadlines. The results show the PBTS algorithm is better than the alternative approaches in terms of performance-cost. The performance is close to the theoretical low bound and it takes only a few seconds even for large workflows.

Zhou, A. C., et al. [21]. They are developed a scheduling algorithm called Dyna. Dyna is a workflow scheduling algorithm designed to minimize the expected cost of executing tasks in cloud. The authors developed an approach to find the solution this approach has a two-step, first, to minimize the cost and satisfying the deadline based on A*-based instance configuration, and also to select the on-demand for each task in the workflow. The Second approach is started from the on-demand instance configuration, by using a hybrid instance configuration of both spot instances and on-demand for executing a workflow for reducing cost. The authors deployed Dyna algorithm on both the simulator and Amazon EC2 to evaluate its effectiveness with three scientific workflow applications.

Patel, G., et al. [14]. Represent "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing", min-min algorithm is the simplest and common algorithm used in cloud, the authors improved and Enhanced min-min algorithm and they named Enhanced load balanced min-min algorithm (ELBMM) algorithm. They assumed an algorithm has two phases, phase one is selecting the minimum completion time by helping of min-min algorithm. The second phase is selecting the tasks with maximum completion time than assigns it to the appropriate resource. As the result, the ELBMM shows a better result for utilizing resources.

Raju, I. R. K., Varma, P. S., et al, [15] proposed "Deadline aware two stages scheduling algorithm in cloud computing". In this paper, the authors used two types of Virtual Machines (VM) for Scheduling n jobs by using deadline aware Two Stage scheduling algorithm. This model reduces the violation to deadlines and average waiting time by allocates Virtual Machines (VM's) as a resource to the requested jobs based on scheduling the jobs and processing time by considering deadlines with respect to waiting time and response time. The simulation environment find the proposed algorithm gives better performance when compared with SJF, FCFS scheduling algorithms.

Vasile, Mihaela-Andreea, et al, [19]. Proposed a hybrid scheduling algorithm considers hierarchical clustering for different types of application. The proposed algorithm is based on different traditional scheduling algorithms. These algorithms are: First and Earliest Deadline First algorithm for independent tasks, Earliest Time First and Modified Critical Path algorithm for DAGs, and Shortest Job. The proposed algorithm has two phases; first phases, tasks are assigned to groups of resources, and the second phase, will use the classical scheduling algorithm for each group of resources. The authors evaluated the performance and cost in a realistic setting of CloudSim tool. However, the results showed the proposed algorithm is suitable for Heterogeneous Distributed Computing.

Mohammed, Abdulrahman, et al. [13] proposed a new algorithm named PBMM algorithm in order to enhance system performance and task scheduling of the cloud computing. This algorithm has two phases: in the first phase will check the priority of the tasks and select the highest priority overall and execute it first. Then it will select the next highest priority and so on. If there is more than one task have the same priority the second phase will check the minimum execution time for all task and execute it first. Figure 3 illustrates the PBMM Scheduling Algorithm.

Algorithm 2 The PBMM Scheduling Algorithm

```
for all submitted tasks in the set; Ti do
    for all resources; Rj do
        Ctij = Etij + rtj;
    end for
end for
Do while tasks set is not empty
    Find task Ti that has maximum priority ; Pi
    if (more than one task have same priority)
        Find task Tk that give minimum execution time
        Assign task to resource Rj that gives maximum priority and minimum expected
completion time
    else
        Assign task to resource Rj.
    end if
    Delet task Tk from the set
    Updet ready time Rtj for theselected resources j
    Update Cij for all Ti
End Do
```

Fig. 3. The PBMM Scheduling Algorithm

III. PROBLEM STATEMENT

In cloud of things, there are millions of distribution devices (things such as phones, laptops, vehicles, smart watches, medical appliances ,remote sensing ,etc.) needs to store massive data in cloud storage [12, 16], the algorithm that responsible for scheduling tasks in the cloud should be effective and offering good performance to analyze and store all data in an effective manner. The cloud of things works under the concept “pay as you go” [12] that means there are different levels of performance that offer to the customers depending on the cost of services. The main challenge in Cloud of Things is the performance and the cost “performance can be costly” [12, 17, 10]. However, the PBMM algorithm came to solve the shortcomings of the Min -Min algorithm [13]. The PBMM algorithm works very well with customers who paying for good services, but for customers who paying less they will get low performance. For example, if there are 1000 tasks and the first task (T1) has the lowest priority over all tasks, T1 will wait until all tasks executed then it will execute. The problem is that T1 takes a long time to execute or may be lost. Meanwhile, assume there is a task (Tn) has the highest priority over all tasks, this task will execute first but it will be costly, so we need to improve the PBMM algorithm to get a good performance with low cost for all customers.

The Problems when using PBMM algorithm are: it executed only the highest priority tasks, then the next priority till all tasks executed, if there is a task has low priority must wait until all tasks executed then it will execute, or may it loss if the scheduler is too busy. Moreover, the main Problems are.

1) Cost performance is too high
2) It is too expensive for high priority users
3) Low priority tasks may wait for a long time to run or may loss
4) Sometimes resource utilization is not available for low priority user.
5) For low priority user, the makespan is too long, and the performance is too poor.

By proposing a new algorithm we will try to solve these Problems as showing in the next section.

IV. PROPOSED IPBMM MODEL

The part responsible for scheduling data is the third-party broker that decides which user-base should be served first. We proposed an algorithm named IPBMM (Improved PBMM Algorithm) works by aid of the Cloud of Things (CoT) service; this service is Sensing and Actuation as a Service (SAaaS) [12]. However, it has two-phase:

1) Sensing phase: In this phase, the sensor observes the capacity of the link (maximum throughput) and compares with total tasks throughput.
2) Actuation phase: Depending on the comparison in the first phase, the priority of the all takes will change depending on the capacity of the link. However, if the throughput is low (Green throughput) so all tasks will have the same priority, but if the throughput medium (Yellow throughput) all tasks have two priority levels (high, low), and if the throughput is high (red throughput) all tasks will have the default priority that gives when user created.

The way data is sent will be determined by the capacity of the link. This method works as described below:

A- The REQUEST messages routing:

1) When a User generates a set of tasks, it will specify the name of the User itself, the priority that has, and the ID for the intended application, these pieces of information used also for routing back the responses. Then the tasks are sent to the data center as a REQUEST.
2) Tasks received by the Service Broker from User, and seeing it tagged as a REQUEST. The Service Broker observes the link capacity by using a sensor, depending on the capacity of link, the tasks will separate into different levels of priority as following.
   - Green throughput:
     If the throughput is less than or equal the one-third of link capacity, that means the traffic in the network is low. Depending on this, all users will have the same level of priority (low level).
   - yellow throughput:
     If the throughput is greater than the one-third of link capacity or less than the two-third of link capacity that means the traffic in the network is Average, In this case, all users will separate into two levels of priority (high level, and low level).
   - Red throughput:
     If the throughput is equal or more than the two-third of link capacity, that means the traffic in the network is high, In this case, all users will have the same the priority that given to them when they were created.
3) After selecting the priority level, the Service Broker sends over the tasks to the Data Center.

B- The RESPONSE messages routing:

For RESPONSE messages, the routing is much simpler than REQUEST messages.

1) The DataCenter hands over the response message to the Service Broker and tagged as a RESPONSE.
2) The Service Broker receives the message from Data Center and seeing it tagged as a RESPONSE.
3) Finally, the Service Broker will use the originator field of the task to identify the destination and the right User-Base to send it over to him/her.

To calculate the throughput, The Service-broker maintains a list of the request processing time. Then it projects the response time by adding the appropriate network delay to the processing time. However, to find the Throughput it needs to calculate the size of the request message and the time of response as follows.

- **Calculate Data Transmission latency:**

To calculate the data transmission latency we will use the following formula:

$$DT_{latency} = D_{transfer} + N_{latency} \qquad (1)$$

Where $DT_{ransfer}$ is the time for transfer the size of data of a single request (D_{rq}) from a source location to a destination, And $N_{latency}$ is the network latency; it is taken from the latency matrix depending on the source location and destination.

- **Calculate user bandwidth:**

To calculate the user bandwidth we will use the following formula

For single user:

$$Bw_{peruser} - D_{rq} / DT_{latency} \qquad (2)$$

Where, $Bw_{peruser}$ is the bandwidth per user.

For all users:

$$Bw_{allusers} - \sum_{i=1}^{n} Bw_{peruser\ i} \qquad (3)$$

Where, $Bw_{allusers}$ is the total available bandwidth for all users.

- **Calculate the throughput bandwidth:**

Throughput is never exceeding the network bandwidth, but it can be equal or less than the network bandwidth. Throughput is equal $Bw_{allusers}$

$$TH = Bw_{peruser} \qquad (4)$$

$$TH_{Max} = NBw \qquad (5)$$

Where, TH is the throughput for all users, TH_{Max} is the maximum rate of data that can be sent through the link, it knew also as bandwidth, and NBw is the network bandwidth.

- **Calculate the levels of priority for proposed algorithm:**

The data is sent by determined the capacity of link, this method works as described below:

1. Green Throughput

$$\textit{If the } TH \leq (1/3) * TH_{Max} \qquad (6)$$

2. Yellow Throughput

$$\textit{If the } (TH < (1/3) * TH_{Max}) \ \&\& \ (TH > (2/3) * TH_{Max}) \qquad (7)$$

3. Red Throughput

$$\textit{If the } TH \geq (2/3) * TH_{Max} \qquad (8)$$

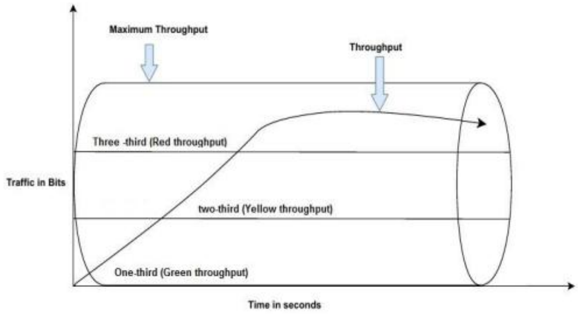


Fig. 4. The levels of priority for the proposed algorithm.

A. The Proposed IPBMM Model (Algorithm and Pseudocode)

The high priority lead to high performance and high cost as well, the proposed algorithm tries to minimizing the cost and waiting time for customers by minimize the priority levels and keeping high performance with low cost. To scheduling data, the proposed scheduling algorithm is presented in figure 5. This algorithm starts by calculates the completion time for all tasks, and the Throughput for all users. Then, compare Throughput with the capacity of the link. Depending on Throughput value (Green Throughput, Yellow Throughput, and Red Throughput); the priority level is determined for all tasks. The PBMM algorithm works by finding and executing the highest priority task first. Finally, if there is a set of tasks having a same priority, the algorithm executes the task with minimum execution time. Repeat this process until all tasks complete execution.

Algorithm 1 The cot-PBMM Scheduling Algorithm

```
for all submitted tasks in the set; Ti do
    for all resources; Rj do
        CTij = ETij + RTij ;
        THi = Dtransfer + Nlatency ;
        TH = TH + THi ;
    end for
end for
Do while tasks set is not empty
    if (TH ⩽ (1/3) * THMax )
        Assign all tasks set Ti in same priority
    else-if ( TH ⩾ (2/3) * THMax )
        Assign all tasks set Ti in default priority
    else
        Assign all tasks set Ti into two levels of priority
    end if
    Find task Ti that has maximum priority ; Pi
    if (more than one task have same priority)
        Find task Tk that give minimum execution time
        Assign task to resource Rj that gives maximum priority and minimum expected completion time
    else
        Assign task to resource Rj ,
    end if
    Delet task Tk from the set
    Updet ready time RTij
    Update CTij for all Ti
End Do
```

Fig. 5. The IPBMM Scheduling Algorithm.

Fig. 6. Chart for IPBMM Algorithm.

V. PERFORMANCE EVALUATIONS&

In order to evaluate the proposed scheduling algorithms we have used a simulation tool knows as Cloudsim, it uses to evaluate the hypothesis before deployment in the real environment [9]. In this section, we evaluate the performance of the proposed algorithm.

A. Simulation setup

Based on the properties and attributes of cloud computing, using a simulation tool named cloudsim platform and by changing some parameters and adding new ones in the cloudsim platform, the result will be as follows:

1) Cloudlets: In this experiment, all tasks generated randomly and dynamically. We propose set tasks for simulation, each task has 64KB lengths of instructions, 30 Mb input filesize, 30 Mb output filesize.
2) Every packet has a priority value given by a number between 0 and 3. The first priority value is free (0), it is the lowest priority value, the second priority value is silver (1), the third priority value is gold (2), and finally, the last priority value is platinum (3), it is the highest priority value, as shown in table II.
3)

Table II Priority Given to Tasks

Packs	Free	Silver	Gold	Platinum
Priority given	0	1	2	3

4) The running time for each task is 0.11 s.
5) Each host has one CPU, and the CPU performance is 1000 MIPS, 2048 MB of RAM, 1000000 MB of storage, and 1000 Kbits/s Bandwidth.
6) Virtual Machines (VM), for each VM requires one CPU with 1000 MIPS, 1000 MB of storage, 512 MB of RAM, 10 Megabytes/s Bandwidth, and Xen for VMM.

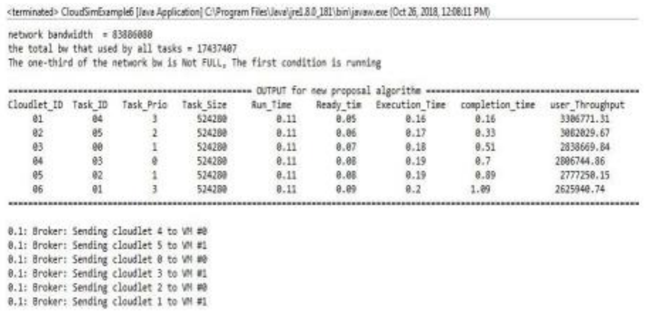
Based on these attributes and properties the following simulation environment was set.

B. Performance impact of the proposed algorithm

To evaluate the performance and the cost for the proposed algorithm we will examine every condition in the proposed algorithm.

- **Green throughput**

As mentioned in the previous section, Green throughput happened when the total throughput is equals or less than one-third of the maximum throughput TH ≤ (1/3) * THMax. To illustrate the Green throughput how it works, we simulated six tasks (Task0, Task1, Task2, Task3, Task4, and Task5) which are submitted by different users as showed in Figure.7.



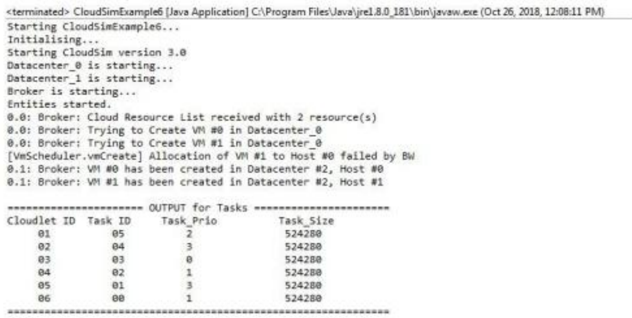
```
<terminated> CloudSimExample6 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Oct 26, 2018, 12:08:11 PM)
network bandwidth  = 83886080
the total bw that used by all tasks = 17437407
The one-third of the network bw is Not FULL, The first condition is running

================================================ OUTPUT for new proposal algorithm ================================================
Cloudlet_ID  Task_ID  Task_Prio  Task_Size  Run_Time  Ready_tim  Execution_Time  completion_time  user_Throughput
    01         04        3        524280      0.11      0.05        0.16            0.16           3306771.31
    02         05        2        524280      0.11      0.06        0.17            0.33           3082029.67
    03         00        1        524280      0.11      0.07        0.18            0.51           2838669.84
    04         03        0        524280      0.11      0.08        0.19            0.7            2806744.86
    05         02        1        524280      0.11      0.08        0.19            0.89           2777250.15
    06         01        3        524280      0.11      0.09        0.2             1.09           2625940.74
===================================================================================================================================

0.1: Broker: Sending cloudlet 4 to VM #0
0.1: Broker: Sending cloudlet 5 to VM #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 3 to VM #1
0.1: Broker: Sending cloudlet 2 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
```

Fig.7. tasks (Green throughput).

By applying the proposed algorithm on the previous tasks, algorithm will check the throughput for all users, and compare it with network bandwidth. Depending on this, all tasks will have the same priority because of the throughput of the total tasks is lass then one-third of the capacity of the link, the algorithm will find the task that has minimum execution time overall tasks and runs first, then the next task with minimum execution time and so on. The algorithm will repeat this process until all the tasks are scheduled, as showed in figure.8.



```
<terminated> CloudSimExample6 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Oct 26, 2018, 12:08:11 PM)
Starting CloudSimExample6...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by BW
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1

===================== OUTPUT for Tasks =====================
Cloudlet ID  Task ID   Task_Prio       Task_Size
     01         05         2             524280
     02         04         3             524280
     03         03         0             524280
     04         02         1             524280
     05         01         3             524280
     06         00         1             524280
============================================================
```

Fig. 8. Execution Tasks by Using Proposed Algorithm.

The total throughput for all users is 17437407 b/s and the maximum throughput is 83886060 b/s. The first condition is applied. In the previous figure, Task4 has the minimum execution time overall tasks so it will execute first, then task5, until the last task in set (task 1), because task1 has the longest execution time.

- **Yellow throughput**

In Yellow throughput the total throughput is greater than the one-third of maximum throughput or less than the two-third of the maximum throughput (TH < (1/3) * THMax) && (TH > (2/3) * THMax). To illustrate the Yellow Throughput how it works, we simulated twelve tasks (Task0, Task1, Task2, Task3... and Task11) which are submitted by different users as showed in figure.9.

```
<terminated> CloudSimExample6 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Oct 27, 2018, 11:40:55 AM)
Starting CloudSimExample6...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by BW
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1

====================== OUTPUT for Tasks ======================
Cloudlet ID  Task ID    Task_Prio        Task_Size
    01          11          0               524280
    02          10          3               524280
    03          09          3               524280
    04          08          1               524280
    05          07          3               524280
    06          06          2               524280
    07          05          1               524280
    08          04          3               524280
    09          03          3               524280
    10          02          1               524280
    11          01          0               524280
    12          00          3               524280
==============================================================
```

Fig. 9. Tasks (Yellow Throughput)

By applying the IPBMM algorithm on the previous tasks, the second condition will apply because the total throughput for all users is 40598587 b/s and the maximum throughput is 83886060 b/s.

The IPBMM algorithm will group the four levels priorities (packs) in to two groups as shown in table III.

Table III: Priority Given to Tasks (Yellow Throughput)

/	Low priority		High priority	
Packs	Free	Silver	Gold	Platinum
Priority given	0	1	2	3

The proposed algorithm will apply the PBMM algorithm with two levels of priority (Low priority, High priority) as showed in figure.10.

```
<terminated> CloudSimExample6 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Oct 27, 2018, 11:40:55 AM)
network bandwidth  = 83886080
the total bw that used by all tasks = 40598587
The two-third of the network bw is Not FULL, The second condition is running

============================================ OUTPUT for new proposal algorithm ============================================
Cloudlet_ID  Task_ID  Task_Prio  Task_Size   Run_Time   Ready_tim  Execution_Time  completion_time  user_Throughput
    01          04        3       524280       0.11       0.01        0.12            0.12            4406089.99
    02          06        2       524280       0.11       0.03        0.14            0.26            3835269.27
    03          09        3       524280       0.11       0.04        0.15            0.4           3585586.45
    04          03        3       524280       0.11       0.04        0.15            0.55            3559524.31
    05          10        3       524280       0.11       0.05        0.16            0.7           3377749.55
    06          00        3       524280       0.11       0.05        0.16            0.86            3373553.16
    07          07        3       524280       0.11       0.06        0.17            1.03            3145606.91
    08          11        0       524280       0.11       0.02        0.13            1.15            4148169.36
    09          02        1       524280       0.11       0.06        0.17            1.33            3084024.01
    10          05        1       524280       0.11       0.08        0.19            1.51            2827536.93
    11          01        0       524280       0.11       0.08        0.19            1.71            2706105.18
    12          08        1       524280       0.11       0.09        0.2           1.91            2629371.75
============================================================================================================================
```

Fig. 10. Execution Tasks by Using Proposed Algorithm.

In the previous figure, there are four different levels of priority (0, 1, 2, and 3). By applying the IPBMM algorithm, the priorities will group into two groups (Low priority, High priority). The algorithm will execute the first group that has the High priority, in this group there are many tasks, the task that has minimum execution time will execute first, then the next task with minimum execution time and so on. In the same way, the Low priority group will execute after the High priority group complete execution.

- **Red throughput**

In Red throughput the total throughput is equal or greater than the two-third of maximum throughput TH ≤ (1/3) * THMax.

To illustrate the Red throughput how it works, we simulated eighteen tasks (Task0, Task1, Task2, Task3….. and Task17) which are submitted by different users as showing in figure.11.

```
<terminated> CloudSimExample6 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Oct 27, 2018, 12:48:37 PM)
Starting CloudSimExample6...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by BW
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1

====================== OUTPUT for Tasks ======================
Cloudlet ID  Task ID    Task_Prio        Task_Size
    01          17          1               524280
    02          16          3               524280
    03          15          0               524280
    04          14          3               524280
    05          13          1               524280
    06          12          2               524280
    07          11          0               524280
    08          10          1               524280
    09          09          1               524280
    10          08          1               524280
    11          07          0               524280
    12          06          0               524280
    13          05          0               524280
    14          04          1               524280
    15          03          2               524280
    16          02          2               524280
    17          01          2               524280
    18          00          0               524280
==============================================================
```

Fig. 11. Tasks (Red Throughput).

By applying the IPBMM algorithm on the previous tasks, the third condition will apply, because the total throughput for all users is 64283311 b/s and the maximum throughput is 83886080 b/s. To apply the IPBMM algorithm: first, all tasks have the original priority level that given when created. Depending on this, the algorithm will find the task that has the highest priority overall tasks and executes it, then the next highest priority task and so on. Second, while the first step is executing if there are two tasks or more have the same priority an algorithm will execute the task that has minimum execution time. As showing in Figure.12.

```
<terminated> CloudSimExample6 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Oct 27, 2018, 12:48:37 PM)
network bandwidth  = 83886080
the total bw that used by all tasks = 64283311
 The two-thirds of the network bw is FULL, The third condition is running

============================================ OUTPUT for new proposal algorithm ============================================
Cloudlet_ID  Task_ID  Task_Prio  Task_Size   Run_Time   Ready_tim  Execution_Time  completion_time  user_Throughput
    01          16        3       524280       0.11       0.01        0.12            0.12            4531703.69
    02          14        3       524280       0.11       0.03        0.14            0.25            3773562.41
    03          12        2       524280       0.11       0.03        0.14            0.4           3681549.02
    04          02        2       524280       0.11       0.05        0.16            0.56            3224149.48
    05          01        2       524280       0.11       0.07        0.18            0.74            2909686.06
    06          03        2       524280       0.11       0.07        0.18            0.92            2876433.69
    07          09        1       524280       0.11       0.01        0.12            1.04            4555595.64
    08          10        1       524280       0.11       0.02        0.13            1.16            4121931.81
    09          08        1       524280       0.11       0.03        0.14            1.3           3772313.74
    10          04        1       524280       0.11       0.03        0.14            1.45            3683003.83
    11          17        1       524280       0.11       0.04        0.15            1.59            3544838.73
    12          13        1       524280       0.11       0.05        0.16            1.76            3213508.99
    13          07        0       524280       0.11       0.02        0.13            1.89            4084179.08
    14          06        0       524280       0.11       0.03        0.14            2.02            3819926.06
    15          11        0       524280       0.11       0.04        0.15            2.17            3604834.1
    16          00        0       524280       0.11       0.04        0.15            2.32            3432327.09
    17          05        0       524280       0.11       0.08        0.19            2.51            2764296.33
    18          15        0       524280       0.11       0.08        0.19            2.71            2689470.54
============================================================================================================================
```

Fig. 12. Execution Tasks by Using Proposed Algorithm.

VI. PERFORMANCE COMPARISON OF MIN -MIN, PBMM, AND IPBM ALGORITHMS

In scheduling, the performance is the amount of useful work accomplished by an algorithm. It is estimated in terms of speed execution, efficiency, and accuracy of the algorithm. When it comes to algorithm performance, the following metrics are involved:

1. Completion time:

Completion time is the length of time taken to execute the complete tasks for any particular user.

To calculate the completion time for any particular user, the following formula is used.

$$C\_time_{taski} = W\_time_{taski} + E\_time_{taski} \quad (9)$$

Where C_timetaski represents the completion time of the ith task, W_time represents the Waiting time and E_time represents Execution time.

$$TC\_time_{useri} = \sum_{i=1}^{n} C\_time_{taski} \quad (10)$$

Where TC_timeuseri represents the total time that takes to execute the user ith tasks.

2. Makespan

The Makespan can be defined as the total time difference between the beginning and end of the total workflow. The makespan is affected by the delay of the execution time task.

$$Makespan_{useri} = com_{time} - st_{time} \quad (11)$$

Where comtime represents the completion time of ith task and sttime represents the Starting time of ith task.

3. Cost

Cost refers to the total payment that paid to the cloud service providers for the utilization of resources by the cloud service consumers (users). To calculate the total cost of executing tasks, the cost varies from one another based on completion time, deadline, and service specification as specified by the cloud service providers. Therefore, Equation 1 show the cost of executing the task of a VM.

$$Cost = \sum_{i=1}^{n} task^{i}(Ti * Ci) \quad (12)$$

Where Ti is the completion time of ith task and Ci is the cost of ith VM.

A. Experimental Approach

As the main focus of this section is to demonstrate the benefit of using computational performance as a metric when comparing IPBMM, PBMM, and Min-Min algorithms. In order to evaluate the performance, we will use a virtual tool to examine all algorithm independently.

- **Simulation setup**

By having four users, and based on the attributes and properties that mentioned in section 5.1 the simulation environment was set.

- **Performance impact:**

As discussed in the previous sections the IPBMM algorithm has three different possibilities (Green, Yellow, and Red throughputs). To evaluate the performance impact of the three algorithms (IPBMM, PBMM, and Min-Min algorithms), three different examples will be presented.

Example 1 (Green throughput):

Assume that there are one thousand tasks, which are submitted by four users. By applying the three algorithms independently on the same data set as following:

1) Min-Min algorithms:

The Min-Min algorithm will find the task that has minimum execution time overall tasks and executes first. Then the next tasks with minimum execution time and so on, Algorithm will repeat this process until all the tasks are scheduled. The completion time is illustrated in figure.13.

2) PBMM algorithms:

By applying the previous tasks on PBMM algorithm. First, all tasks have different priority, depending on this, the algorithm will find the task that has the highest priority overall tasks and executes it, then the next highest priority task and so on. Second, while the first step is executing, and if there are two tasks or more have the same priority an algorithm will execute the task that has minimum execution time. As showing in figure.13.

3) IPBMM algorithms:

By applying the IPBMM algorithm on the previous tasks, an algorithm will check the throughput for all users, and compare it with network bandwidth. Depending on this, all tasks will have the same priority because of the throughput of the total tasks is less than one-third of the capacity of the link (the total throughput for all users is 3495775610 b/s and the maximum throughput is 10737418240 b/s,). The algorithm will find the task that has minimum execution time overall tasks and executes first, then the next task with minimum execution time and so on. The algorithm will repeat this process until all the tasks are scheduled, as showed in figure.13.

4) Performance Comparison

Depending on table 2, there are four different packs of costs (Free, Silver, Gold, platinum), each pack has a different cost, Such as free pack has free of cost, while the platinum pack has the highest cost. To evaluate the Performance of the previous algorithm we will use the Performance metrics as follows:

a- Completion time

As showed in Figure 13, the Min-Min algorithm executed the task that has minimum execution time, which gives the short completion time for the task that has a short execution time. Nevertheless, without taking the priority as a part of scheduling, the Min-Min algorithm leads to poor performance for the users paid for good service.

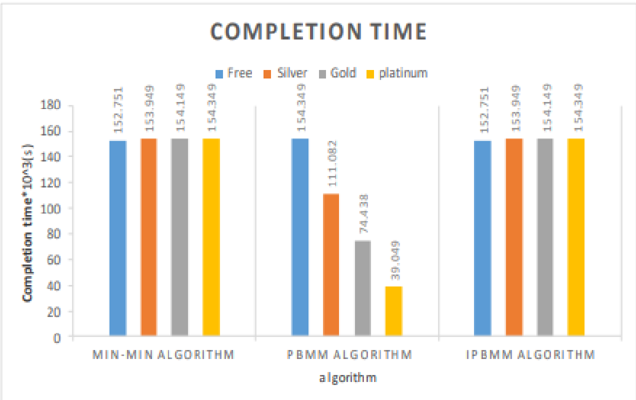


Fig. 13. The Completion Time of Three Algorithms

The Min-Min algorithm presented the worst performance in terms of completion time because the paid packs executed late based on execution time. The PBMM algorithm presented the good performance in terms of completion time because it executed the tasks that have a paid pack first based on their priority.

While the IPBMM algorithm executed the task that has minimum execution time, which gives the short completion time for the task that has a short execution time. The IPBMM algorithm presented the best performance in terms of Makespan because it is considered all packs as a free pack (no priority levels) by giving all tasks a free cost of charging (green throughput).

b- Makespan

Makespan is the total time difference between the beginning and end of the total workflow Figure 14 shown the Makespan of the three algorithms.

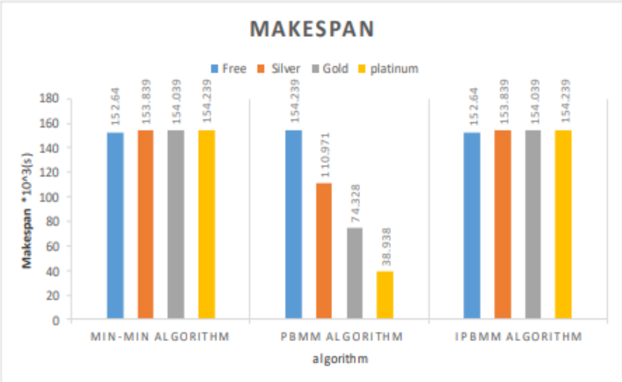


Fig. 14. The Makespan of Three Algorithms

As showed in Figure 14, the Min-Min algorithm presented the worst performance in terms of Makespan.

The PBMM algorithm presented the good performance in the terms of Makespan. While the IPBMM algorithm presented the best performance in terms of Makespan.

c- Cost

The second experiment verifies the costs of the three algorithms. The cost depends on the packs' subscription, and how fast the task is executed. Which means if the task takes a long time to execute the cost will increase. While the task takes less time to execute the cost will decrease. For calculating the cost of the three previous algorithms we will use equation 4, and each priority level in 2 has a different value of cost as follow: free pack $0.0, silver pack $0.12, clod pack $0.17, and finally platinum pack $0.48, per hour [11]. This experiment is considered the deadline without considering the budget cost constraints. The results are shown in Figure 15.

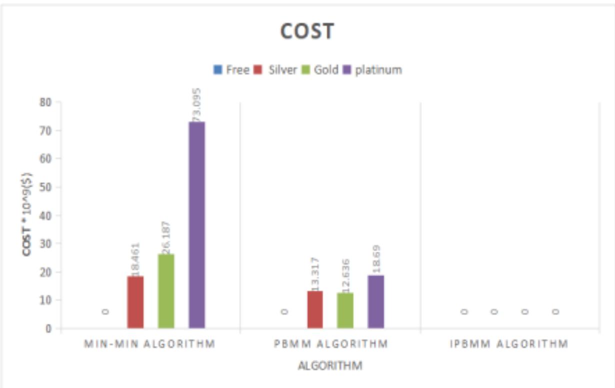


Fig. 15. The Cost of Three Algorithms

Example 2 (Yellow throughput):

Assume for now that there are two thousand tasks, which are submitted by four users. By applying the three algorithms independently on the same data set as following:

1) Min-Min algorithms:

By applying the Min-Min algorithm, with the same steps taken in example 1. The result is showed in figure.16.

2) PBMM algorithms:

By applying PBMM algorithm the result showed in igure.16.

3) IPBMM algorithms:

By applying the IPBMM algorithm on the previous tasks, the second condition will apply because the total throughput for all users is 6969023893 b/s and the maximum throughput is 10737418240 b/s. The algorithm will execute under the second condition. The IPBMM algorithm will group the four levels priorities (packs) in to two groups as showed in table III.

The result of applying the IPBMM algorithm showing in figure 16.

There are four different levels of priority (0, 1, 2, and 3). By applying the IPBMM algorithm, the priorities will group into two groups (Low priority, High priority). The algorithm will execute the first group that has the High priority, in this group there are many tasks, the task that has minimum execution time will execute first, then the next task with minimum execution time and so on. In the same way, the Low priority group will execute after the High priority group complete execution.

4) Performance Comparison

To evaluate the Performance of the previous algorithm we will use the Performance metrics as follows:

a- Completion time

By comparing the three previous algorithms, the Min-Min algorithm presented the worst performance in terms of completion time because the paid packs executed late based on execution time. The PBMM algorithm presented the good performance in terms of Makespan because it executed the tasks that have a paid pack first based on their priority. While the IPBMM algorithm presented the best performance in terms of cost because it considered all packs as two packs.

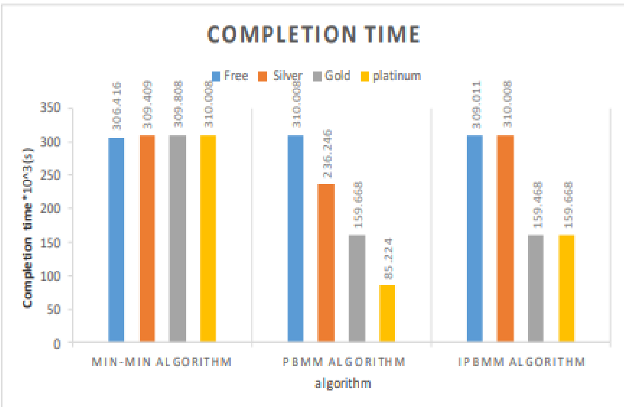


Fig. 16. The Completion Time of Three Algorithms

Retrieval Number: A4839119119/2019©BEIESP
DOI: 10.35940/ijitee.A4839.119119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

b- Makespan:

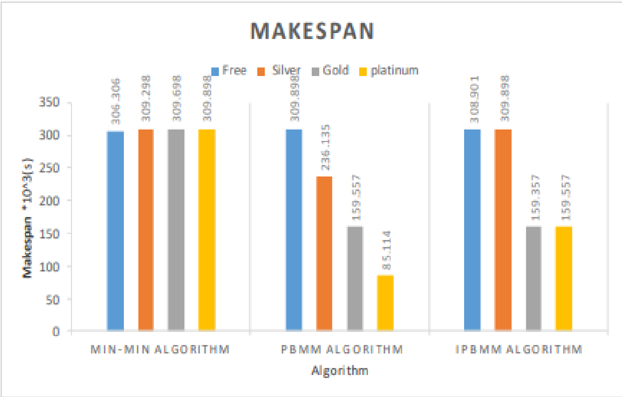


Fig. 17. The Makespan of Three Algorithms

As shown in the Figure 17, the Min-Min algorithm presented the worst performance in terms of Makespan. The PBMM algorithm presented the good performance in the terms of Makespan. While the IPBMM algorithm presented the best performance in terms of Makespan.

c- Cost

Depending on table III, there are two values of cost as follows: free pack $0.0 (Low priority), silver pack $0.12 (High priority), per hour. This experiment is considered the deadline without considering the budget cost constraints. The results are shown in Figure 18.

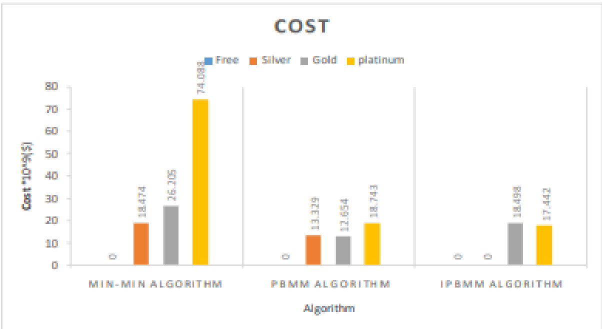


Fig. 18. The Cost of Three Algorithms

Example 3 (Red throughput):

Assume that there are three thousand tasks, which are submitted by four users. By applying the three algorithms independently on the same data set as following:

1) Min-Min algorithms:

By applying the Min-Min algorithm, the result showed in figure.19.

2) PBMM algorithms:

By applying the PBMM algorithm, the result showed in figure.19.

3) IPBMM algorithms:

By applying the IPBMM algorithm on the previous tasks, the third condition will apply, because the total throughput for all users is 10490165857 b/s and the maximum throughput is 10737418240 b/s, the algorithm will execute under the third condition. To apply the IPBMM algorithm: first, all tasks have the original priority level that given when created. Depending on this, the algorithm will find the task that has the highest priority overall tasks and executes it, then the next highest priority task and so on. Second, while the first step is executing if there are two tasks or more have the same priority an algorithm will execute the task that has minimum execution time. In this case, the IPBMM algorithm will work as PBMM works. As showed in figure.19.

4) Performance Comparison

To evaluate the Performance of the previous algorithm we will use the Performance metrics as follows:

a- Completion time

By comparing the performance of the Min-Min, PBMM, IPBMM algorithms on example 3 as showed in figure.19. The Min-Min algorithm presented a poor performance in terms of completion time because it does not take the priority as a part of scheduling. While the PBMM and IPBMM algorithms showed the best performance in terms of Makespan because it executes the tasks that have high priority first.

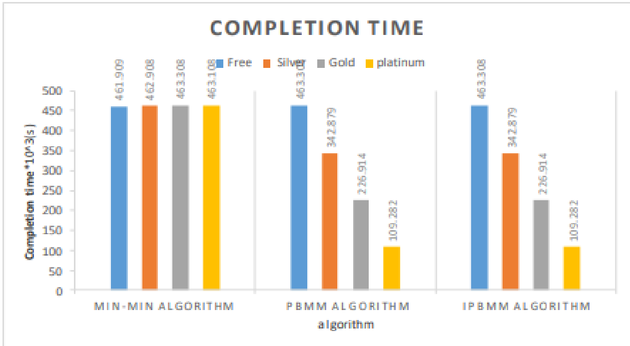


Fig. 19. The Completion Time of Three Algorithms

b- Makespan

As showed in figure 20, the Min-Min algorithm presented the worst performance in terms of Makespan. While The PBMM and IPBMM algorithms showed the best performance in terms of Makespan because it executes the tasks that have high priority first.

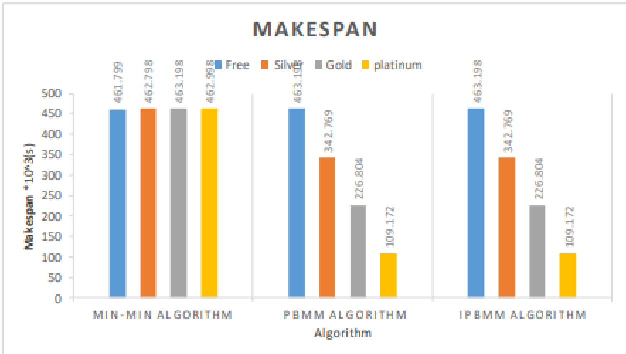


Fig. 20. The Makespan of Three Algorithms

c- Cost

The second experiment verifies the costs of the three algorithms. The min-min algorithm presented the highest cost, while PBMM IPBMM algorithm presented the lowest cost. The IPBMM algorithm is executed under the third condition, so all tasks had the original priority level that given when created as showed in figure.21.

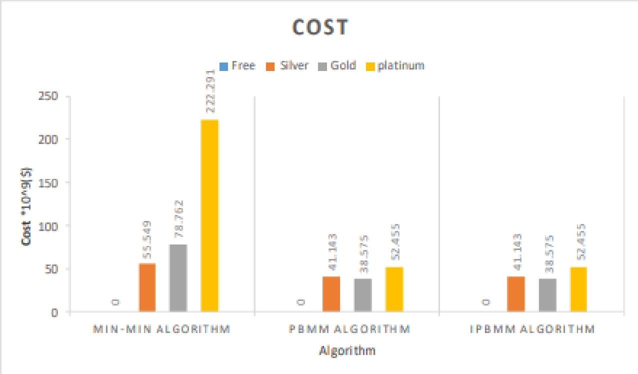


Fig. 21. The Cost of Three Algorithm

B. Results:

Testing was completed by executing several sequences of simulation, as showed in the previous section the makespan is different for different conditions, as well as the cost. It is clear that the performance of IPBMM algorithm is the best, the PBMM algorithm is good, while the Min-Min algorithm is worst in terms of performance for users who paid for good service. Tables IV shown the comparison of the previous examples.

Table IV Performance Comparison of Min-Min, PBMM, and IPBMM Algorithms

Algorithm	Performance Example1	Performance Example2	Performance Example3	Performance overall
Min-Min	Low	Low	Low	Low
PBMM	Good	Good	Good	Good
IPBMM	High	High	High	High

In general, the efficiency and performance of the IPBMM algorithm depend on the capacity of the link (carrier). If capacity is high (traffic in the network is high), the performance is good and the costly, but if the capacity is medium (traffic in the network is medium), the performance is good as well as the cost. While if the capacity is low (traffic in the network is low), the performance is good and the cost is free. The main properties and comparison of previous scheduling algorithms are illustrated in Table V.

Table V Properties and comparison of MIN-MIN, PBMM, and IPBMM algorithms.

Algorithm	Key Idea	Main Objective	Focus on	Performance criteria
Min-Min	It executed the task that has minimum execution time.	Reduce the Makespan and avoid long execution time.	Task Scheduling	Time
PBMM	Improving Min-Min algorithm	improve the performance and minimize the waiting time for customers who pay money to get a good service,	Workflow Scheduling	Makespan, Time
IPBMM	Improving PBMM algorithm	The objective behind this algorithm is to reduce the impact of high cost of PBMM algorithm and improve the performance for all users.	Workflow Scheduling	Makespan, Time, Cost

VII. CONCLUSION

In this paper, we proposed a new algorithm works on a cloud of things environment, which came from improving and enhancing the PBMM algorithm. The PBMM algorithm proposed solving the priority tasks execution problem in Cloud, but it still has many drawbacks. However, the proposed algorithm seeks to improve the performance and reduce the monetary cost in cloud. The proposed algorithm works on a third-party broker, which works by the aid of the Cloud of Things service, the proposed algorithm has two-phase, the first phase is sensing: in this phase, the sensor observes the throughput for all tasks and compares with link capacity. The Second phase is Actuation: depending on the comparison in the first phase, the priority of the all takes will change depending on the capacity of link. However, the outcomes of the experiment indicate that the proposed algorithm is giving better results and reduce data transfer costs and data traffic compare with the PBMM algorithm.

In future work, we will improve the method of priority level detection to reduce the calculation time that leads to reducing the execution algorithm time. In addition, we will apply the proposed algorithm on the actual Cloud of Thing environment as the case study for our future work.

REFERENCES

1. M. Aazam, I. Khan, A. A. Alsaffar, and E.-N. Huh, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved," in Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014, 2014, pp. 414-419.
2. S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Cloud of things for sensing as a service: sensing resource discovery and virtualization," in 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1-7.
3. S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," Future Generation Computer Systems, vol. 29, pp. 158-169, 2013.
4. A. Botta, W. De Donato, V. Persico, and A. Pescapé, "On the integration of cloud computing and internet of things," in 2014 International Conference on Future Internet of Things and Cloud, 2014, pp. 23-30.
5. A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: a survey," Future generation computer systems, vol. 56, pp. 684-700, 2016.
6. E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," Future Generation Computer Systems, vol. 27, pp. 1011-1026, 2011.
7. R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," IEEE Transactions on Parallel and Distributed Systems, vol. 25, pp. 1787-1796, 2013.
8. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and experience, vol. 41, pp. 23-50, 2011.
9. S. S. Chauhan, E. S. Pilli, R. Joshi, G. Singh, and M. Govil, "Brokering in interconnected cloud computing environments: A survey," Journal of Parallel and Distributed Computing, 2018.
10. M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing," Journal of Network and Computer applications, vol. 67, pp. 99-117, 2016.
11. S. H. H. Madni, M. S. A. Latiff, M. Abdullahi, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," PloS one, vol. 12, p. e0176321, 2017.

12. A. Mohammed, H. Obaid, P. K. Pattnaik, and S. K. Pani, "Towards on Cloud of Things: Survey, Challenges, Open Research Issues, and Tools," in 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 1458-1463.
13. A. M. H. Obaid, S. K. Pani, and P. K. Pattnaik, "A priority based on min-min algorithm for reducing make span task scheduling in cloud computing," 2018, vol. 7, p. 5, 2018-09-07 2018.
14. G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," Procedia Computer Science, vol. 57, pp. 545-553, 2015.
15. I. R. K. Raju, P. S. Varma, M. Sundari, and G. J. Moses, "Deadline aware two stage scheduling algorithm in cloud computing," Indian Journal of Science and Technology, vol. 9, pp. 1-10, 2016.
16. R. Ramakrishnan and L. Gaur, "Smart electricity distribution in residential areas: Internet of Things (IoT) based advanced metering infrastructure and cloud analytics," in 2016 International Conference on Internet of Things and Applications (IOTA), 2016, pp. 46-51.
17. R. Ranjan, L. Wang, P. P. Jayaraman, K. Mitra, and D. Georgakopoulos, "Special issue on Big Data and Cloud of Things (CoT)," Software: Practice and Experience, vol. 47, pp. 345-347, 2017.
18. B. Shanthan, A. D. V. Kumar, E. Govindrajan, and L. Arockian, "Scheduling for Internet of Things Applications on Cloud: A Review," Imperial Journal of Interdisciplinary Research, vol. 3, 2017.
19. M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," Future Generation Computer Systems, vol. 51, pp. 61-71, 2015.
20. L. Zeng, B. Veeravalli, and X. Li, "SABA: A security-aware and budget-aware workflow scheduling strategy in clouds," Journal of parallel and Distributed computing, vol. 75, pp. 141-151, 2015.
21. A. C. Zhou, B. He, and C. Liu, "Monetary cost optimizations for hosting workflow-as-a-service in IaaS clouds," IEEE Transactions on Cloud Computing, vol. 4, pp. 34-48, 2015.

AUTHORS PROFILE

Abdulrahman Mohammed Hussein Obaid received his B. Sc. computer and IT engineering from the Sana'a University, Yemen in 2011and M. tech. degrees in computer engineering from the JNTUH University, India in 2016. Currently, he is a Ph.D. scholar at school of computer engineering, KIIT University, India. His research interest covers Cloud Computer, Cloud of Things, Internet of Things and scheduling algorithm.

Dr.Santosh Kumar Pani is working as Associate Professor at School of Computer Engineering, KIIT (Deemed to be University), Bhubaneswar, India. He has over eighteen years of teaching experience and served in many key administrative positions in the University. His research areas include Program analysis, Internet of Things, Cloud Computing and Blockchain Technology.

Dr. Prasant Kumar Pattnaik is currently working as a Professor, in School of Computer Engineering, KIIT University, Bhubaneswar. He is a Senior Member of IEEE, Fellow Member of IETE, Senior member of IACSIT. His research area includes: Cloud Computing, Usability Engineering, Green Computing, MANET, Wireless Sensor Network