ISSN 2278-3091
Volume 8, No.1.4, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse3281.42019.pdf
https://doi.org/10.30534/ijatcse/2019/3281.42019

An Efficient Virtual Machine Scheduling Algorithm To Minimize Makespan And Maximize Profit Using Hyper Heuristic Approach

Ahmad Alauddin Ariffin[1], Aws Fadhil Ibrahim[2], Sazlinah Hasan[3], Rohaya Latip[4]
UPM Faculty of Computer Science and Information Technology, Malaysia
[1]alauddin@upm.edu.my
[2]awsfadhili91@gmail.com
[3]sazlinah@upm.edu.my
[4]rohayalt@upm.edu.my

ABSTRACT

Virtual machines are assigned to hosts, depending on its current resource usage and not considering their overall utilization. It is one of the main problems in cloud computing that can reduces the system performance. The scheduling is used to schedule tasks for better utilization of resources by allocating certain tasks to particular resources at a particular time. The purpose of scheduling is to select the most excellent and suitable resource available to execute the tasks or to assign computer machines to execute tasks with minimal completion time is but still feasible. An efficient task scheduling algorithm is needed for improve the system performance. In this paper, the focus is on improving the virtual machines scheduling performance for makespan and cost. The proposed process of scheduling includes three main processes. The first process is the Clustering Formation based on the characteristics such as Processor, Memory and Bandwidth. The second process is known as the Hyper Analytical Task Scheduling Algorithm, and based on the scheduled tasks, the Policy-based Profit Maximization Algorithm was proposed in the final process. The performance comparison of the proposed work is analyzed through some empirical results. The result shows that the proposed work significantly reduces the makespan of task scheduling and gives high profit compared with the other scheduling algorithms.

KEY WORDS: Virtual Machine, Makespan, Policy-based Profit Maximization, Hyper Analytical Task Scheduling, VM Cluster Formation

1. INTRODUCTION

Cloud computing is a model where the configurable computing resources are being shared, which are based on on-demand usage, that can be minimally managed or needed interaction for service providers[1].

Cloud computing is categorized into three types of service models known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In IaaS, computing resources are considered as services. In any IaaS, services provided by service providers are similar to facilitating hardware resources to users. Cloud computing can be considered as an extension to Grid Computing. One of the main features of this type of Cloud Computing is Virtualization. With virtualization, virtual machines are created on physical machines, which are limited in numbers. In PaaS model, providers deliver computing platform software [2]. For SaaS model, applications are provided as services that runs on infrastructure created and maintained by the providers [3].

Depending on the location of cloud providers, the virtual machines are placed inside physical machines along with the distributed data centers. These computing resources are also being shared between users according to the user needs over the resources. These are the factors that requires a proper resource allocation that can give huge impact on the system performance [4].

In order to satisfy user's requests, these virtual machines created in data centers in cloud needs to be efficiently scheduled [5]. The scheduling process of virtual machines is to find the way in which physical machine resources are allocated in order to create a virtual machine object [6]. The requests to create virtual machines are sent the Physical Machines (PM) at a suitable Data Center (DC), which are then scheduled in order to fulfill the needed resources such as RAM, Memory, Central Processing Unit.

Today, cloud computing is the most preferred way over handling tasks because of the flexibility and reliability. It also scalable as physical machines can be added easily. The tasks sent to these cloud systems are mostly based on NP-Hard and NP-Complete, and some scheduling algorithms, which are based on rules such as deterministic algorithms are used as they are simple and easy to be build. However, according to [19, 21], these algorithms are not suitable as the tasks requirements varies in demands.

As tasks begins to change to more complex demands, scheduling of these resources also changes. Heuristic algorithms are being introduced to improve the scheduling process in cloud computing, which also attracts cross domain researchers. Some of the later research even proposed of combining more than two heuristic algorithms to improve scheduling process, which are also known as hybrid heuristic scheduling algorithms.

There are a few basic heuristic algorithms which are used to create hybrid heuristic scheduling algorithms. The main objective of combining these basic algorithms is to find better results of heuristic algorithms such as Artificial Bee Colony, Invasive Weed Optimization, Particle Swarm Optimization, and Ant Colony Optimization.

The main contribution of this work is to group available PMs into number of clusters before their allocation. Provided with efficient VM scheduling algorithm, this could minimize the makespan of the tasks and finally find the maximum profit for provides based on the policy.

2. RELATED RESEARCH

Schedulers can be classified into several types, and listed above are the most essential ones:

- **Optimal or non-optimal:** An optimal scheduler is able to schedule a specific set of tasks if it is schedulable by schedulers.
- **Preemptive or non -preemptive:** A preemptive scheduler is able to make a decision on the suspension of a particular task before the execution is completed, and resume the process at a later time, as a higher priority tasks are ready. However, non-preemptive schedulers are not able to suspend the tasks in a similar manner, as the tasks that are already in progress will be able to be balanced unwillingly.
- **Static or dynamic:** Static scheduler works by computing the tasks implementation sequence preceding the run-time. The task characteristics information is essential for this scheduler, but it also generates small run-time overhead. This type however, is not able to handle non-predicted or periodic events. On the other hand, dynamic schedulers decides during the the system run-time. While this permits in creating a system that is more flexible, it however, also produces some overhead.

Over the years, a number of traditional scheduling algorithms have been presented, namely earliest due date (EDD) [7], critical path method (CPM) [8], project evaluation and review technique (PRET) [9], dynamic programming [10], and branch-and-bound [11]. As these methods are constructed based on singular or a small number of rules in arranging and managing the tasks, they are mostly simpler in implementation. While several traditional methods guarantee in providing the scheduling problem with the best solution, they all however, fell short in terms of solving the problem in reasonable time frame, mainly on a larger or complex problem.

The two main disadvantages mentioned earlier have inspired researchers to identify ways to modify the search strategy of these methods. Firstly, the time consumption of a majority of full search algorithms, namely dynamic programming and branch-and-bound are usually high, mainly due to the large number of checks they have to perform. Secondly, although deterministic algorithms are easy to implement and substantially fast, they do come with one major drawback, which is easily falling into local optima. As discussed earlier, although the traditional algorithms are efficient enough in generating optimal solution for small scheduling problems in a timely manner [12], but the same cannot be said for larger scheduling problems [13].

The main cloud scheduling problem is that the resources and jobs have to be assigned and scheduled in a manner that can ease the users in multiple areas, namely job completion in minimal time, and boost the throughput Cloud Resource Provider and user satisfaction.

The major distinction between scheduling in a traditional single processor and cloud environment is that, the latter consists of several processors. In addition, the cost and execution time in carrying out similar tasks in different processor units might differ due to the fact that the processor units might consist of different resources in a cloud. Generally, scheduling in a cloud computing system or a multi-core processor can be divided into two parts, namely the execution order of the tasks determination, and also assigning tasks to suitable processor unit. The most crucial problem in multi-processor systems is task mapping and scheduling with various sets of limitations, which is an NP-complete problem. Task queues in a system are divided into three parts, namely high priority, medium priority, and low priority tasks [27]. Upon the arrival of requests, the algorithm then measures the priority level based on the cost of the requests on each processor, before sending them to task queues that match them best. After all the steps have been completed, the tasks are executed by the system based on their priorities. Lin and Lu [28] proposed SHEFT method, which works by selecting the task with the highest priority, to measure the time completion for every processor units, before sending the it to the processor with the shortest time completion.

Besides the methods discussed earlier, heuristics algorithms have also been utilized in recent studies for the purpose of scheduling in cloud computing systems. Meta-heuristics is mainly utilized to acquire a possible solution in a short time frame [29]. In order to identify possible directions or combinations from huge space of solution, this method utilizes strategic guesses, and discovers better possible solutions through repeated calculations. This method consists of three major operators, which are transition, evaluation, and determination. The first in the category, the transition operator, works by modifying the current solution to the later state. The two usual methods of transition for combinatorial problems are constructive and perturbative. This operator's complexity level also differs based on the meta-heuristics design. The second operator, which is the evaluation operator, measures the value of the problem's objection function, which includes scheduling problem's makespan. When measuring the solutions, a number of metaheuristics do not utilize an objection function directly. Rather, other measurement instruments are employed in determining the decision space value. Thus, the value might not be representative of what the real quality of the optimization problem solution, as the value can be measured through either the decision or objective space. The third operator, which is the determination operator, guides the search. It decides both the directions, and also the search's diversification or intensification, which might manipulate the speed of convergence.

A majority of the studies which employed meta-heuristics in scheduling have been utilized on job scheduling problems and grid computing, namely tabu searches, genetic algorithms, particle swarm optimization (PSO), and ant colony optimization (ACO), and also large distributed systems. Cloud computing systems scheduling is a job shop scheduling problem variant. Optimization algorithm has been employed by numerous studies to solve problems in scheduling. In this work the new hyper scheduling algorithm contains, Ant Colony Optimization (ACO), Particle Object Swarm (PSO), Artificial Bee Colony (ABC), and Invasive Weed Optimization (IWO). These scheduling algorithms are selected based on their priority.

Resource-Aware Scheduling Algorithm (RASA), a new task scheduling algorithm has been proposed by Saeed Parsa and Reza Entezari-Maleki [14]. RASA mainly consists of two traditional algorithms, namely Min-min and Max-min. This algorithm uses the benefits of Min-min and Max-min and hides their shortcomings. This algorithm takes into account the tasks' incoming rate, task's limit, and task execution expenditure on every resource, but does not consider the cost of the transmission. Another reliable scheduling algorithm has been proposed by Arash Ghorbannia et al., [15], which divides major job into sub jobs. The acknowledge and request time are determined separately to balance the jobs.

Another new algorithm which is based on the effect of RASA has been proposed by El-Sayed T. El-kenawy et al, [16]. They have improved Max-min algorithm by changing its selection basis from total time into predictable implementation time. In order to display the disseminated systems' simultaneous performance, petri nets are employed. It has been shown that Max-min achieves agendas that are equivalent to the lower makespan before unique Max-min and RASA. SLA Based Resource Provisioning for Hosted Software as-a-Service Applications in Cloud Computing Environments Cloud computing has been proposed by Linlin Wu, and Saurabh Kumar Greg [17], which is a solution to address problems on namely, distribution, configuration, licensing, and enterprise applications operation that are linked to the traditional IT infrastructure, deployment models, and software sales. By moving to the cloud model from a traditional model has a number of advantages, namely providing revenue that is ongoing for SaaS providers, and lowering the cost for enterprise clients and complexity of maintenance. In order to measure the Quality of Service (QoS), a Service Level Agreement (SLA) needs to be established.

A Stochastic model has been introduced by Dario Bruneo [18] to identify the performance of data center and QoS in IaaS cloud computing systems by Dario Bruno [18]. As a result of the various strategies available, which ranges from the association with other clouds to the placement of VM, the management of cloud datacenter has been the main issue. To comply with the users that do not have a sufficient readily available compute resources, W.R. Helen [19] has introduced Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS, which provides Infrastructure as a Service (IaaS). IaaS works by multiplexing to attain the economy of scale, thus encounters a tasks scheduling challenge, in order to achieve the peak demand, whilst still factoring in the QoS preservation. A few solutions have been proposed by previous studies to solve this issue, namely, cloud federation, and proactive machine purchasing. Nevertheless, cloud federation is currently still hardly achievable, and the proactive machine purchasing is not economic. In this study, the researcher has proposed source allocation framework which is beneficial when the demand is higher that the resources, as it can be utilized by the IaaS provider to transfer the tasks to External Clouds (ECs). Besides that, no formal agreement on inter cloud is needed by this architecture for the cloud federation. However, the main challenge of this proposed way is on how the allocation of users tasks can be done in such a way that it will be beneficial in boosting the IaaS provider profit, while still factoring in the QoS. Integer programming model (IP) is then used to formulate this problem, and which then utilizes self-adaptive learning particle swarm optimization based scheduling approach (SLPSO) to solve it. SLPSO consists of four strategies of updating, which provide the robustness and diversity of each particle by adaptively updating its velocity.

In a separate study, done by Jianying Luo, Lei Rao, and Xue Liu[20], the Temporal Load Balancing with Service Delay Guarantees has been proposed for Data Center Energy Cost Optimization Cloud computing services, as they have been integrated in our daily lives. Internet data center (IDC) is an infrastructure that is used to support these services, and the amount of IDCs energy consumption soars with the rising demand for cloud computing services. The issue of IDCs energy management has brought great attention from various sectors, namely the industry and academia. Energy cost minimizing in deregulated electricity for IDCs is the main focal point of this study. They have proposed Energy Cost Optimization-IDC algorithm (eco-IDC) and a two-stage design, in order to schedule workload dynamically, before implementing it on IDC servers through an input queue, and to manipulate the electricity price's temporal diversity. A real life enterprise production data center workload traces and the price of electricity are used for experiment purposes. The study has identified that the energy cost of IDCs has been significantly reduced through the selected approach, which assures a service delay bound, and when the size of the service delay bound is huge, it helps to alleviate the workload drop.

Besides that, in boosting performance and lowering the cost, Haitao Yuan, Jing Bi [21] have also proposed Cost-Aware Workload Scheduling and Admission Control (CAWSAC) to be implemented on Distributed Cloud Data Centers Multiple heterogeneous applications which concurrently run in distributed cloud data centers (CDCs). A market with a diverse geographical range of energy cost and bandwidth may cause a challenging problem, especially on the ways of minimizing CDC's provider's total cost. Based on the problem encountered, this study proposed two approaches, which firstly, a revenue-based workload admission control method, that considers various factors, namely revenue, expected response time, and priority, before admitting the requests. Secondly, a cost-aware workload scheduling method, which optimizes the internet service providers' selection pool for CDC, and also the total figure of servers that are active in each CDC. From the results of the trace driven simulation, they have discovered that when compared with other existing methods, the two methods proposed earlier not only can reduce the total cost tremendously, but also boosts CDCs provider's throughput. Wenhong Tian [22] developed an essential element for large-scale Cloud applications, which is a toolkit modeling and simulation of real-time VM allocation in a Cloud Data Center Resource scheduling in infrastructure as a service (IaaS). As the two main elements, which are the network infrastructure, conditions, and environment might be beyond developers' control, it is highly challenging to conduct extensive research regarding real environment issues.

In their study, Ferrandi et al. [30] have utilized ACO in scheduling both the tasks and also the communications between each tasks. In the effort of reducing exploration's execution time, they have proposed a multi-stage decision process. Besides that, they have also added a forgetting factor to increase the diversity by reserving the worst solutions. In another study, Wang et al. [31] have introduced CPACO as a solution for multi-agent task allocation problem. As the factor of efficiency, this study utilized agent capacities, while factoring in the cost of communication simultaneously. However, Bai et al. [32] have identified that the application of single colony system in ACO is usually dependent on the pheromones' positive feedback, leading to the fall of system into local optima. Thus, to achieve load balancing for the task scheduling, the study has introduced MACO approach. MACO works by considering the feedbacks from both positive and negative angles from the information shared among multiple colonies, to prevent idle situations. Another solution for this problem has been proposed by Lu and Gu [33] through an ACO based model, which is a load-adaptive cloud resource scheduling model. This model works by monitoring the utilization of CPU and memory, computing node bandwidth, and identifies the hotspot node through a cluster controller. A node that turns into a hotspot releases ants which help to swiftly locate the stagnant nodes nearby, and transfers a part of the load to the stagnant node.

In boosting the service providers in an IaaS environment's profit, Zuo et al. [34] have introduced a self-adaptive learning particle swarm optimization (SLPSO) method. This method consists of four updating strategies that are utilized in updating the particle's velocity and boosting the diversity of the search. PSO has also been employed on traffic lights programming in a study by Garcia-Nieto et al. [35]. The researchers have managed to identify traffic light cycle programs that are successful through PSO. In a separate research done by Wen at al. [36], PSO and ACO are combined to develop a hybrid algorithm. The hybrid resources scheduling algorithm is employed on global and local optimal solution, and also the solutions obtained through ACO. In a similar study, Xie and Wu [37] have experimented on the ACO parameters optimal combination. In order to identify a more efficient parameters combination for ACO performance boosting, the study has utilized PSO.

In addition, honey bee behavior inspired load balancing (HBB-LB), is another algorithm that has been proposed by Babu et al. [40]. By identifying VMs that are overloaded, the algorithm then sends the tasks of these VMs to other VMs with a low load. This algorithm considers the priority level of each tasks removed from the VMs that were overloaded. Besides that, the Artificial Bee Colony (ABC) [38] has been employed for the VM machine scheduling optimization on Cloud computing. This study aims to both analyze the VM load balancing algorithm difference, and reduce data processing time makespan. Heuristic task scheduling with Artificial Bee Colony (HABC) algorithm has also been proposed [39] to be utilized on VMs in heterogeneous Cloud computing. The main objective of the study is the introduction of the current task scheduling and load balancing algorithm, HABC, in minimizing the system's makespan. Multi-objective Artificial Bee Colony Algorithm (TA-ABC) has been proposed through another study [41] to attain task scheduling efficiency. This algorithm works by optimizing a number of areas, namely cost, energy, processing time, and resource utilization of the cloud environment.

Cloud computing is extremely useful to universal users, as it is able to efficiently present the computing resources above the network that are on-demand. In general, users receive the computing resources in cloud data centers through the pay-as-you-go pricing model. Furthermore, the number of companies that are attracted to set up their applications in cloud data centers are growing due to the scale economy introduced through cloud computing. Providing cost-effective services to the users while simultaneously able to guarantee the specified QoS are the main aims of a private cloud provider. Thus, this also means that a private cloud provider's vital objective is to be able to maximize the profit margin.

Yeo and Buyya [23] have proposed a pricing function method that is dependent on both utilization and base pricing rate. Profit maximization can be achieved through this method, as clients are inclined on paying higher price range for resources when the utilization is high, as it boosts the demand for the sources. The opposite will happen if for lower utilization, as providers are inclined to offer lower prices to gain the clients' attention.

Dynamic pricing strategies has also been discussed by Maglaras and Meissner [24], where a fixed amount of a resource is owned by a company, and then utilized for the delivery of various products. The study has demonstrated that the problem can be modified into a simpler form, which gives the company the ability to control the aggregate rate where resources have been used. Another study, done by Eren and Maglaras [25] has examined the the scenario of monopoly pricing for sellers that are equipped with market information that is limited, by taking sellers' various learning capabilities demand into account. They have found that the particular policies will perform better when the pricing policies are not updated and re-optimized constantly. Thus, this will be a suitable option for models that are active, as they tend to demand assumptions that are more rigid.

H. Chen. et al. [42] have done a comparative study between the performance of traditional heuristic algorithms for scheduling, which includes Opportunistic Load Balancing (OLB), FCFS, Minimum Completion Time (MCT), Minimum Execution Time (MET), Max-min, Min-min, and others. Besides thats, as a solution to the makespan and boost the utilization of resources, they have also employed Min-min algorithm in their study. From their findings, they have discovered that, compared to other algorithms, they are able to achieve a better schedule that aids the makespan minimization through Min-min algorithm.

S. Devipriya et al. [43] proposed an alternative of complete time through simple changes in Max-min algorithm based on expected execution time. Through these small changes, the improved Max-min can aid the tasks within cloud in a acquiring lower makespan, when compared with the previous version of Max-min. Z. Zheng et al. [44] in their study, have introduced Parallel Genetic Algorithm (PGA), which is able to accomplish the given tasks 1.5 times faster with two threads, and 2.7 times faster with the usage of four threads, when compared to the performance of the original GA. The rate of resource utilization is significantly higher when it is compared to Round Robin algorithm and the first fit, thus making it easily applied on numerous types of parallel mainframes and computer networks.

K. Dasgupta et al. [45] have conducted an experiment on using GA for the load balancing of resource while scheduling the tasks on the cloud. Experimental outcomes shows that GA manage to reduce 25 to 26% of the response time. M. Shojafar et al. [46] presented a combination of fuzzy idea and GA (FUGE) which is predicted to lower the cost, makespan, imbalance level in cloud while conducting the scheduling of tasks. Fuzzy hypothesis works by analyzing the chromosomes' fitness estimation and crossover operation. The addition of GA helps to improve the performance by lowering the cost of execution and makespan. A 45% of improvement has been demonstrated on the cost of execution, and 50% of makespan improvement over the traditional GA.

Besides that, a study by M. A. Tawfeek et al. [47] also concentrates on the makespan minimization through objective function. In this study, the heuristic function is based on two elements, which are the expected task transfer time, and the execution time. Each ant is also constrained to visiting each VM once. The experiment is employed on Cloudsim simulator, with the tasks number ranging from 100 to 1000. The findings of simulations have demonstrated that when compared with FCFS and RR algorithms, ACO requires lesser time with the increasing number of tasks. The findings have shown that for 1000 tasks, there is a reduction of around 29 to 32% in makespan.

W. N. Chen et al. [48] proposed an improved workflow scheduling algorithm that is based on Ant Colony System (ACS) algorithm. The algorithm has been designed to lower the cost, while still keeping to deadline. To ensure the success of solving the problem, the study has employed two variants of well-defined pheromone, which function as firstly, cost minimizer, and also also makespan minimizer. In order to control the ants in identifying the search directions, the study has divided heuristic information into three types. Each ant utilizes one type of pheromone and one type of heuristics in every cycle to view the probabilities that are controlled adaptively, and balances parameters in the algorithm.

3. METHODOLOGY

To solve the problem an efficient VM scheduling algorithm is needed to provide a better VM scheduling in cloud to reduce the make-span. This work introduces novel policy based service model, which aim at maximizing the profit of cloud providers.

The VM scheduling problem can be defined as follows:

- Find an optimal solution to schedule a given set of VMs $VM=\{VM_1,VM_2,\cdots,VM_N\}$ to a given set of physical machines $PM = \{PM_1, PM_2, \cdots, PM_M\}$. There are M heterogeneous physical machines for dispatching the VM requests.

- A VM request set $VM_{Req} = \{R_1, R_2, \cdots, R_r\}$ is the set of input requests. The VM request represents $R_i = (C, M, B)$ where C is the number of cores, M is the memory and B is the Bandwidth of VM.

In order to accomplish VM scheduling process, this research first group the physical machine (PM) based on their characteristic. Each PM has own number of CPU, Memory and Bandwidth. Using K-Means Clustering Algorithm, PMs are clustered into groups. Next, the hyper task scheduling algorithm is executed to allocate the VM to a suitable cluster that minimize the make-span. Based on the allocated VM, the next phase is the profit value calculation.

The existing system uses hyper heuristic algorithm [26]. This algorithm leverages strengths of heuristic algorithm such as simulated annealing, genetic algorithm, particle swarm optimization, and ant colony optimization by integrating them into a single algorithm. By using this algorithm, the make-span was reduced. But it has some limitations; it uses random selection mechanism to select the low level heuristic. It leads to increase in computational cost for the resource allocation which in turn becomes a burden for both service provider and consumer. The service provider needs to reallocate resources again. In this work the new hyper scheduling algorithm contains, Particle Object Swarm (PSO), Ant Colony Optimization (ACO), Invasive Weed Optimization (IWO) and Artificial Bee Colony (ABC). These scheduling algorithms are selected based on their priority. For maximizing the profit, productive estimation (based on the scheduled VM) and basic First in First Out policy is used to find the maximum profit. Figure 1 shows the system architecture of the VM Scheduling.

4. PROPOSED ALGORITHM

4.1 VM Cluster Formation

The K-means clustering algorithm is used to cluster the available PM into K-Groups. Steps for K-means clustering:

- Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of data point and $V = \{v_1, v_2, \ldots, v_c\}$ be the set of centers.
- Step1: Randomly select c cluster centers.
- Step 2: Calculate the distance between each data point and cluster centers.
- Step 3: Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
- Step 4: Recalculate the new cluster center using:

$$v_i = \frac{1}{c_i} \sum_{j=1}^{c_i} x_i$$

- Step 5: Recalculate the distance between each data point and new obtained cluster centers.
- Step 6: If no data point was reassigned then stop, otherwise repeat from step 3.

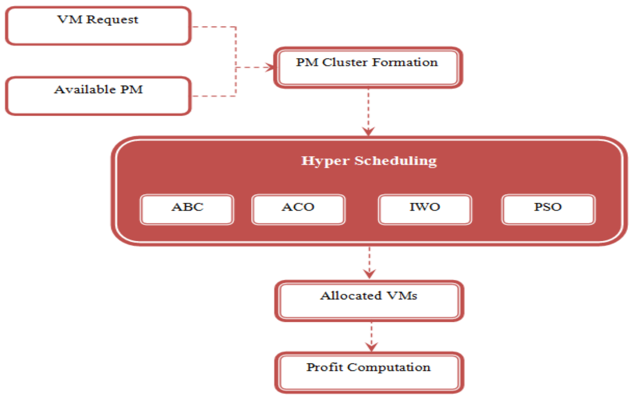

Figure 1 : System Architecture

4.2 Task Sscheduling Algorithm

After grouping the PM into number of clusters, the hyper analytical task scheduling algorithm is applied to allocate the task to the particular cluster that minimize the makespan of tasks.

A hyper-heuristic is a methodology for selecting or generating heuristics to solve hard computational search problems [26]. The basic idea of the algorithm is to use two operator diversity detection and improvement detection to balance the intensity and diversification in the search of the solutions during the process. The algorithm is below

Input: Cloudlet
Output: Makespan

- Step 1: *Input* Task and PM Details
- Step 2: Initialize the population Z= {z_1, z_2,…, z_n}. (Based on the Task)
- Step 3: Is it the First Execution?
- Step 4: If Yes =>
  - o Step 4.1: Randomly select a one algorithm. (ABC, ACO, IWO and PSO)
- Step 5: If No =>
  - o Step 5.1: Select the algorithm with highest priority for scheduling.
- Step 6: While the termination criterion is not met.
  - o Step 6.1: Update the population of solutions Z by using the selected algorithm.
- Step 7: Compute F1 = Improvement Detection (Z).
- Step 8: Compute F2= Diversity Detection (Z).
- Step 9: Compute F3=Pertub(Z)
- Step 10: Compute If Pertub(Z) is true
  - o Step 10.1: Assign “Lowest Priority” to the current algorithm.
  - o Step 10.2: Select another algorithm with the “Highest Priority”.

- Step 11: Repeat the scheduling process until the maximum number of iteration is reached.

Output the Best Makespan as the final solution.

Improvement Detection (Z) The improvement detection operator is used to decide whether to change algorithm or not. This can be done by comparing the makespan obtained using H_i with the current makespan. If the makespan obtained in the H_i is greater than current makespan it results "true". This operator returns false for the following conditions.

- If the current makespan is not improved after ϕ_{ni} iterations
- When maximum number of iteration is reached
- When stop condition is reached.

Improvement Detection function can be depicted as follows

$$F_1 = \begin{cases} false;\ \text{current makespan is not improved after } \phi_{ni} \text{ iteration} \\ \text{true; otherwise} \end{cases}$$

Diversity Detection (Z) This operator is used to decide when to change new algorithm to perform scheduling. The working of this algorithm is as follows. Initially it assigns the solution obtained in the first iteration as the threshold value (ω), ie $\omega = D(Z_0)$. And compares values with the current solution. If the current value is greater than ω, it returns true, else it returns false.

Diversity Detection Function can be depicted as follows

$$F_2 = \begin{cases} true\ ;D(Z_0 > \omega) \\ false\ ;otherwise \end{cases}$$

Perturbation Operator (Z) Perturbation operator is used to select a new low level algorithm for the process of effective scheduling. This operator returns false when improvement detection operator and diversity detection operator returns true, else it returns true. At once it returns true,

1. It assigns "Lowest Priority" to the current algorithm
2. It selects the algorithm with the highest priority for the next schedule.

$$F_3 = \begin{cases} false, F_1 = true\ and\ F_2 = true \\ true, \text{otherwise} \end{cases}$$

4.3 Profit Maximization Algorithm

Productive estimation and basic First in First Out policy is used to find the maximum profit. Procedure:

```
While (Task List is not empty)
{
    • First it takes the first task (i.e first task that allocated to VM)
    • Find the utilization of resources (i.e how much it use the resources (cpu, memory and bandwidth)
    • Compute the price
    • Remove the first task
}
```

5. SIMULATION IMPLEMENTATION

This work was done using CloudSim. It is a simulator; hence, it doesn't run any actual software. It can be defined as 'running a model of an environment in a model of hardware', where technology-specific details are abstracted. Figure 2 illustrated the Implementation Workflow of our work.

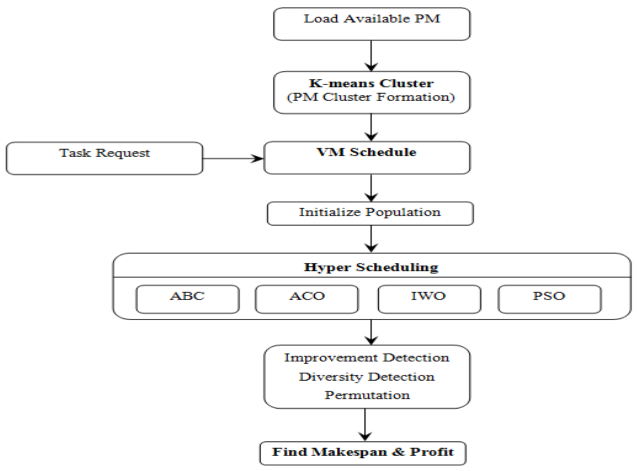

Figure 2 : Implementation Workflow

6. RESULTS AND DISCUSSION

In running the simulation, there are 54 Physical Machines (PM) are used. Each PM has three important properties, which are CPU, Memory and Bandwidth. After applying the K-means Clustering Algorithm, the PMs are categorized into 4 clusters groups, which has different number of PMs. The Instance Type of Each cluster group are Low, Medium, High and Very High. Table 1 shows the Clustered PM that was created after applying K-means Clustering Algorithm. Table 2 shows the requested VM. Each VM contains VM ID, duration (time for accessing the resources), CPU, Memory and Bandwidth.

Table 1 : Clustered PM

Cluster Id	Instance Type	No of PM
1	Low	20
2	Medium	16
3	High	9
4	Very High	9

Table 3 and Figure 4 show makespan calculated by each algorithm for the different algorithm implemented in this research. Hyper-heuristic algorithm showing much less makespan than other and it is much better than other algorithms which are PSO, ACO, ABC, and IWO. From the result, it is clear that hyper-heuristic is the makespan reducing high-performance algorithm than other heuristic algorithms. (Proposed < ABC < PSO < IWO < ACO).

Table 2 : VM Request

VM Id	Duration	CPU	Memory	Bandwidth
v1	4	1	5	200
v2	8	2	8	300
v3	2	4	10	600
v4	10	5	12	600
v5	4	6	15	800
v6	5	8	32	800
v7	12	10	20	1000
v8	10	12	25	2500

Table 3 : Scheduling Time

Algorithm	Makespan
ACO	20
PSO	17
ABC	16
IWO	18
Proposed	14

Figure 3 shows the scheduled VM. In the graph, x-axis represent VMID and y-axis represent cluster ID. Table 4 and Figure 5 show the profit comparison for the different algorithms implemented in this research. Hyper-heuristic algorithm showing a much better profit than other algorithms which are PSO, ACO, ABC, and IWO. From the result, the proposed work maximizes profit compared to other heuristic algorithms. (Proposed > IWO > PSO > ACO > ABC).

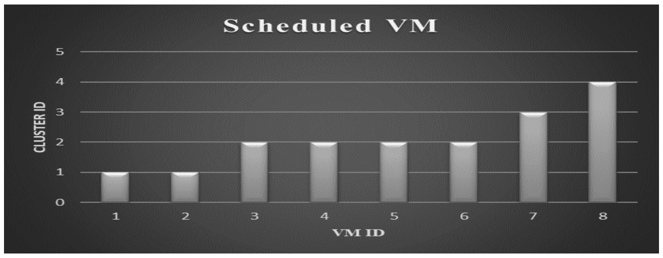

Figure 3 : Scheduled VM

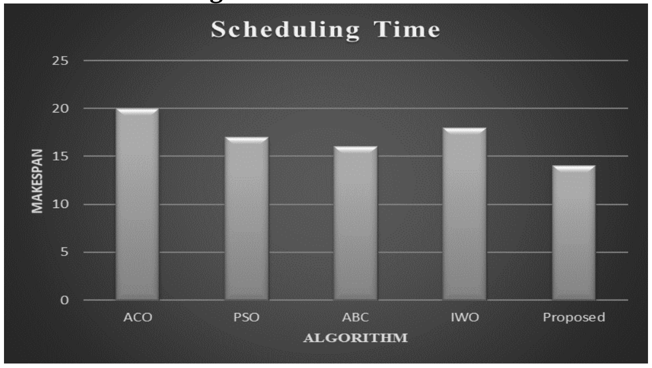

Figure 4 : Makespan Comparison

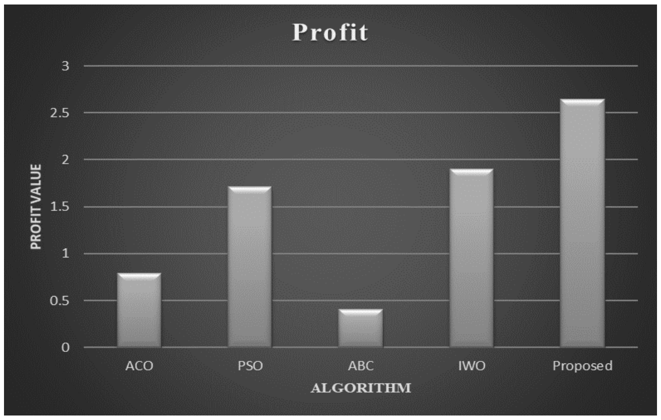

Figure 5: Profit Comparison

Table 4 : Profit Value

Algorithm	Profit
ACO	0.788
PSO	1.711
ABC	0.406
IWO	1.905
Proposed	2.642

7. CONCLUSION

The research provides the novel solution in order to minimize the makespan and maximize the profit of the task in cloud environment. This research is supported by various ideas that are discussed in the literature review supporting the different views towards task scheduling in cloud computing.

Effective scheduling plays a significant role in performance provided by the cloud environment. This thesis analyzed various scheduling algorithm and found the shortcomings of the existing system. And modify the hyper scheduling algorithm based on their priority which could overcome the shortcomings of the existing work. This research also proposed the policy based profit maximization. Further the proposed algorithm to be implemented in a simulator (Cloudsim).

REFERENCES

1. P. Mell and T. Grance. **The NIST Definition of Cloud Computing**, National Institute of Standards and Technology, Information Technology Laboratory, Technical Report Version 15, 2009.
2. Bazarbayev, S., Hiltunen, M., Joshi, K., Sanders, W.H., Schlichting, R. **Content-based scheduling of virtual machines (VMs) in the cloud**. In: Proceedings of 33rd IEEE International Conference on Distributed Computing Systems (ICDCS 2013), pp. 93–101. IEEE, 2013. https://doi.org/10.1109/ICDCS.2013.15
3. Archana Pawar, Deepak Kapgate. **A Review on Virtual Machine Scheduling in Cloud Computing**, IJCSMC, Vol 3, Apr 2014, pg 928-933, 2014.
4. Karan D. Prajapati, Pushpak Raval, Miren Karamt and Potdar. **Comparison of Virtual Machine Scheduling Algorithms in Cloud Computing**, International Journal of Computer Applications (0975 – 8887) Volume 83 – No 15, December 2013, 2013. https://doi.org/10.5120/14523-2914
5. Varma, M.K., Choi, E. **Study and comparison of virtual machine scheduling algorithms in open source clouds**. In: The 11th International Conference on Future Information Technology, 20–22 April 2016, 2016.
6. S.Sotiriadis, N.Bessis, C.Amza, R.Buyya. **Vertical and horizontal elasticity for dynamic virtual machine reconfiguration**, IEEE Trans. Serv. Comput. PP (99) (2016), doi: 10.1109/TSC.2016.2634024. 1–1, 2016. https://doi.org/10.1109/TSC.2016.2634024
7. K. M. Elsayed and A. K. Khattab. **Channel-aware earliest deadline due fair scheduling for wireless multimedia networks**, Wireless Personal Commun., vol. 38, no. 2, pp. 233–252, 2006. https://doi.org/10.1007/s11277-006-9013-1
8. W. H. Kohler. **A preliminary evaluation of the critical path method for scheduling tasks on multiprocessor systems**, IEEE Trans. Comput., vol. C-24, no. 12, pp. 1235–1238, Dec. 1975, 1975. https://doi.org/10.1109/T-C.1975.224171
9. R. Ingalls and D. Morrice. **PERT scheduling with resources using qualitative simulation graphs**, in Proc. Simul. Conf., 2000, vol. 1, pp. 362–370, 2000.
10. H.-J. Sch€utz and R. Kolisch. **Approximate dynamic programming for capacity allocation in the service industry**, Eur. J. Oper. Res., vol. 218, no. 1, pp. 239–250, 2012. https://doi.org/10.1016/j.ejor.2011.09.007
11. D. Shi and T. Chen. **Optimal periodic scheduling of sensor networks: A branch and bound approach**, Syst. Control Lett., vol. 62, no. 9, pp. 732–738, 2013. https://doi.org/10.1016/j.sysconle.2013.04.012
12. S. M. Johnson. **Optimal two- and three-stage production schedules with setup times included**, Naval Res. Logistics Quart., vol. 1, no. 1, pp. 61–68, 1954.
13. M. R. Garey, D. S. Johnson, and R. Sethi. **The complexity of flowshop and jobshop scheduling**, Math. Oper. Res., vol. 1, no. 2, pp. 117–129, 1976.
14. Saeed Parsa and Reza Entezari-Maleki. **RASA: A New Task Scheduling Algorithm in Grid Environment**. World Applied Sciences Journal 7 (Special Issue of Computer & IT): 152-160, 2009. https://doi.org/10.4156/jdcta.vol3.issue4.10
15. Arash Ghorbannia Delavar, Mahdi Javanmard, Mehrdad Barzegar Shabestari and Marjan Khosravi Talebi,(2012) **RSDC (Reliable Scheduling Distributed In Cloud Computing)**. in International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012, 2012. https://doi.org/10.5121/ijcsea.2012.2301
16. El-Sayed T.El-kenawy, Ali Ibraheem El-Desoky, Mohamed F. Al-rahamawy. **Extended Max-Min Scheduling Using Petri Net and Load Balancing**, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-4, September 2012, 2012.
17. Linlin Wu, Saurabh Kumar Greg. **SLA based Resource Provisioning for Hosted Softwareas-a-Service Applications**, IEEE Trans. Services Computer., vol. 7, no. 3, pp. 465–485, Jul. 2014, 2014. https://doi.org/10.1109/TSC.2013.49
18. Dario Bruneo. **A Stochastic Model to Investigate Data Center Performance and QoS**, IaaS Cloud Computing Systems Proc. 32nd IEEE Int.Conf. Comput. Commun., 2013, pp. 2148–2156, 2013.
19. A. Shahina Banu and W. R. Helen. **Self Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud**, IEEE Trans. Autom. Sci. Eng., vol. 12,no. 1, pp. 309–323, Jan. 2014, 2014.
20. Jianying Luo, Lei Rao, and Xue Liu. **Temporal Load Balancing with Service Delay Guarantees for Data Center Energy Cost Optimization**, IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 3, pp. 775–784, March 2014, 2014. https://doi.org/10.1109/TPDS.2013.69

21. Haitao Yuan, Jing Bi. **CAWSAC: CostAware Workload Scheduling and Admission Control for Distributed Cloud**, Data Centers vol. 2, no. 1, January-March 2014, 2014.
22. Wenhong Tian. **A Toolkit for Modeling and Simulation of Real-time Virtual Machine Allocation in a Cloud**, Data Center Resource scheduling in infrastructure as a service (IaaS). Future Gener. Comp. Sy., vol. 25, no. 6, pp. 599–616, 2009.
23. Yeo, C.S.,& Buyya, R. **Pricing for utility-driven resource management and allocation in clusters**, Proceedings of the ADCOM(pp.32–41). Ahmedabad, India, 2004.
24. Maglaras, C.,& Meissner, J. **Dynamic pricing strategies for multiproduct revenue management problems**, Manufacturing & Service Operations Management,8(2),136–148.10.1287/msom.1060.0105, 2009. https://doi.org/10.1287/msom.1060.0105
25. Eren,S.S.,& Maglaras, C. **Monopoly pricing with limited demand information**, Journal of Revenue and Pricing Management, 9(1-2), 23–48, 2009. https://doi.org/10.1057/rpm.2009.41
26. C. Tsai et al. **A hyper-heuristic scheduling algorithm for cloud**, IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 236–250, Feb. 2014. https://doi.org/10.1109/TCC.2014.2315797
27. Selvarani S, Sadhasivam G. **Improved cost-based algorithm for task scheduling in cloud computing**, Proceedings of the IEEE international conference on computational intelligence and computing research, pp 1–5, 2010. https://doi.org/10.1109/ICCIC.2010.5705847
28. Lin C, Lu S. **Scheduling scientific workflows elastically for cloud computing,** Proceedings of the IEEE international conference on cloud computing, pp 746–747, 2011.
29. Blum C, Roli A. **Metaheuristics in combinatorial optimization: overview and conceptual comparison**, ACM Comput Surv35(3):268–308, 2003.
30. Ferrandi F, Lanzi PL, Pilato C, Sciuto D, Tumeo A. **Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems,** IEEE Trans Comput Aided Des Integr Circuits Syst 29(6):911–924, 2010.
31. Lu W, Zhiliang W, Siquan H, Lei L. **Ant colony optimization for task allocation in multi-agent systems**, China Commun 10(3):125–132, 2013.
32. Bai L, Hu YL, Lao S, Zhang WM (2010). **Task scheduling with load balancing using multiple ant colonies optimization in grid computing**. In: Proceedings of the international conference on natural computation, pp 2715–2719, 2010.
33. Lu X, Gu Z. **A load-adapative cloud resource scheduling model based on ant colony algorithm**. In: Proceedings of the IEEE international conference on cloud computing and intelligence systems, pp 296–300, 2010. https://doi.org/10.1109/CCIS.2011.6045078
34. Zuo X, Zhang G, Tan W. **Self-adaptive learning PSO based deadline constrained task scheduling for hybrid IaaS cloud**, IEEE Trans Autom Sci Eng 11(2):564–573, 2014.
35. Garcia-Nieto J, Olivera A, Alba E. **Optimal cycle program of traffic lights with particle swarm optimization**. IEEE Trans Evol Comput 17(6):823–839, 2013.
36. Wen X, Huang M, Shi J. **Study on resources scheduling based on ACO allgorithm and PSO algorithm in cloud computing**. In: Proceedings of the international symposium on distributed computing and applications to business, engineering science, pp 219–222, 2012.
37. Xie X, Wu P. **Research on the optimal combination of ACO parameters based on PSO.** In: Proceedings of the international conference on networking and digital Society, vol 1, pp 94–97, 2010.
38. Kruekaew B, Kimpan W. **Virtual machine scheduling management on cloud computing using artificial bee colony**, Proceedings of the International MultiConference of engineers and computer scientists, vol 1, pp 12–14, 2014.
39. Kimpan, W., & Kruekaew, B. **Heuristic Task Scheduling with Artificial Bee Colony Algorithm for Virtual Machines**. In Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems, 2016 Joint 8th International Conference on (pp. 281-286). IEEE, 2016. https://doi.org/10.1109/SCIS-ISIS.2016.0067
40. L.D. Dhinesh Babu,P.Venkata Krishna. **Honey bee behavior inspired load balancing of tasks in cloud computing environments**, Applied Soft Computing, Vol. 13,2013;pp.2292-2303, 2013.
41. R. Jena. **Task scheduling in cloud environment: A multi-objective ABC framework**, Journal of Information and Optimization Sciences, vol. 38, no. 1, pp. 1-19, 2017, 2017.
42. Chen, H., Wang, F., Helian, N., & Akanmu, G. **User-priority guided min-min scheduling algorithm for load balancing in cloud computing**, 2013 National Conference on Parallel Computing Technologies, PARCOMPTECH 2013, 2013.
43. Devipriya, S., & Ramesh, C. **Improved max-min heuristic model for task scheduling in cloud**, Proceedings of the 2013 International Conference on Green Computing, Communication and Conservation of Energy, ICGCE 2013, 883–888, 2013.
44. Zheng, Z., Wang, R., Zhong, H., & Zhang, X. **An approach for cloud resource scheduling based on parallel genetic algorithm**. ICCRD2011 - 2011 3rd International Conference on Computer Research and Development, 2, 444–447, 2011.

45. Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S. **A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing**. Procedia Technology, 10, 340–347, 2013. https://doi.org/10.1016/j.protcy.2013.12.369
46. Shojafar, M., Javanmardi, S., Abolfazli, S., & Cordeschi, N. **FUGE: A joint meta heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method**. Cluster Computing, 18(2), 829–844, 2015.
47. Tawfeek, M., El-Sisi, A., Keshk, A., & Torkey, F. **Cloud task scheduling based on ant colony optimization.** International Arab Journal of Information Technology, 12(2), 129–137, 2015
48. Wkh, P., Ri, E., Vr, D., Lqihulru, W., Zloo, F., Ljqruhg, E. H., … Wkh, R. (n.d.). * , ,1752'8&7,21.
49. Wei, L., Zhang, X., Li, Y. Y., & Li, Y. Y. **An Improved Ant Algorithm for Grid Task Scheduling Strategy**. Physics Procedia, 24, Part C(0), 1974–1981, 2012. https://doi.org/10.1016/j.phpro.2012.02.290