Optimized Task Scheduling for Efficient Resource Allocation in Cloud Computing

Project report by

**Abhishek Kumar**

**(M190667CA)**

*In partial fulfilment for the award of the degree of*

Master of Computer Applications

Under the guidance of

**Dr. T. Veni**

Assistant Professor



Department of Computer Science and Engineering

National Institute of Technology Calicut

Calicut, Kerala, India - 673601

May, 2022

**ACKNOWLEDGEMENT**

I would like take this opportunity and express my gratitude to everyone who helped me during the course of my final year project, **“Optimized task Scheduling for Efficient Resource Allocation in Cloud Computing”**. I am extremely grateful for their guidance, invaluable constructive criticism throughout the project and also the friendly advices given during whole duration of the project work. I am grateful to them for providing their illuminating and truthful views on several issues related to the project.

Firstly, I would like to express my gratitude to Dr. T. Veni, my supervisor, for her guidance, monitoring, and continuous encouragement during my final year project. I want to take this opportunity to thank Dr. Murali Krishnan (HOD of CSE Department) and also special thanks to our project coordinator Dr. Pournami PN for their constant attention, guidance and advice to help in the completion of this project.

Additionally, I would like to acknowledge the support and motivation I have received from my parents in achieving this goal. Furthermore, I would like to thank all of my friends and classmates for their valuable comments and suggestions, which inspired me to improve the project. Finally, I would like to thank everyone who helped me complete this project directly and indirectly.

**Date:** 26 May, 2022 ABHISHEK KUMAR

M190667CA

**DECLARATION**

*I, hereby declare that the project, titled* **"Optimized task Scheduling for Efficient Resource Allocation in Cloud Computing"***, is my original work and that, to the best of my knowledge, it does not contain anything published or written by another person, nor is it a part of anything contemplated for an award of any other degree or diploma from any university or other institution of higher learning, except where appropriate acknowledgment and reference have been made.*

**Signature**

**Date:** 26 May, 2022 Abhishek Kumar

**Place:** NIT Calicut M190667CA

**NATIONAL INSTITUTE OF TECHNOLOGY CALICUT**

**KERALA, INDIA – 673 601**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



**CERTIFICATE**

*Certified that this is a bonafide report of the project work titled*

**OPTIMIZED TASK SCHEDULING FOR EFFICIENT RESOURCE ALLOCATION IN CLOUD COMPUTING**

*done by*

**ABHISHEK KUMAR**

*of sixth semester MCA, during the winter semester 2021-’22, in partial*

*fulfilment of the requirements for the award of the degree of Master of*

*Computer Applications of the National Institute of Technology Calicut.*

****

**Date:** 26 May 2022 **Dr. T.Veni**

**Place:** NIT CalicutAssistant Professor

(Department of CSE)

**(Official Seal)**

**Abstract**

The problem attempted in this project is a heuristic task scheduling algorithm that is based on Ant-Colony Optimization (ACO) technique which will schedule the requested tasks to the VMs by evaluating the makespan and execution time of the task on the assigned virtual machine and also ensuring the efficient utilization of available resources. The allocation of available resources to incoming task requests is very challenging as tasks cane be of different nature size and type. Hence, an effective task scheduling technique is needed for assigning tasks to resources in dynamic cloud environment. There are various techniques and algorithms already available for task scheduling but with rapid growth of cloud system and their users every day, there is still the need of various improvements in existing approach for efficient utilization of cloud resources and minimization of makespan.

The method used in this project for task scheduling is a meta-heuristic approach based on Ant Colony Optimization (ACO) technique, where we are using natural phenomenon of ant colonies to find optimal allocation of tasks to available VMs with two main objective of reducing makespan and execution time while maintaining the load balance of virtual machine with the consideration load balance factor. The method is simulated using the Cloudsim toolkit and the results shows that proposed approach performs better than the traditional Ant Colony Optimization approach which only considers makespan while allocation of tasks. Comparative analysis also shows that it outperforms the other famous heuristic scheduling method called Genetic Algorithm (GA) both the parameters makespan and total execution time.

**Contents**

**Chapter 1.** [**INTRODUCTION**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#introduction) **1**

1.1 [Problem Definition](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#problem_definition) 1

1.2 [Background](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#background) 2

1.3 [Motivation](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#motivation) 3

**Chapter 2.** [**LITERATURE**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#introduction)**REVIEW4**

**Chapter 3. PROBLEM STATEMENT8**

**Chapter 4.** [**RELATED STUDY**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#introduction) **9**

4.1 ACO Algorithm Overview9

4.2 ACO Parameters 10

4.3 Scheduling Policies in Cloud 10

4.4 Architecture of scheduling in cloud 13

**Chapter 5.** [**METHODOLOGY**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#introduction) **15**

5.1 Proposed Work15

5.1.1 Definition of Performance Metrics 15

5.1.2 Assumptions made for Proposed Algorithm17

5.1.3 Algorithm I 18

5.1.4 Algorithm II 19

5.2 Flowchart of proposed system 21

**Chapter 6.** [**IMPLEMENTATION**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#introduction) **22**

6.1 Experiment Tool: Cloudsim 22

6.1.1 Components of Cloudsim 22

6.2 Stepwise procedure for Algorithm Implementation 24

6.3 Implementation flow diagram 27

6.4 Experimentation set Parameters 28

6.4.1 Experimentation set Parameters 28

6.4.2 VM and Task parameters 28

**Chapter 7.** [**RESULT**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#result)**S & COMPARISON 29**

7.1 Results29

7.2 Comparisons32

**Chapter 8.** [**CONCLUSION**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#conclusion)**34**

[**REFERENCES**](file:///C:\Users\User\Desktop\Final_Report%20(1).docx#references)**35**

**List of Figures**

1. Figure 3.1: Problem Representation 8

2. Figure 4.1: Ant Colony Optimization Example 9

3. Figure 4.2: Space Share policy for both VM and Task 11

4. Figure 4.3: Space shared policy VM and time-shared for Task 11

5. Figure 4.4: Time shared policy VM and space-shared for Task 12

6. Figure 4.5: Time shared policy for both VM and Task 12

7. Figure 4.6: Scheduling Architecture of Cloud 14

8. Figure 5.1: Flow-Chart of the Proposed Algorithm 21

9. Figure 6.1: Relationship between Cloudsim entities 24

10. Figure 6.1: Implementation flow diagram 27

11. Figure 7.1: Makespan Chart I 29

12. Figure 7.2: Makespan Chart II 30

13. Figure 7.3: Total Execution Time Chart I 31

14. Figure 7.4: Total Execution Time Chart II 31

15. Figure 7.5: Makespan Comparison Chart I 32

16. Figure 7.6: Makespan Comparison Chart I 32

17. Figure 7.7: Total Execution Time Comparison Chart I 33

18. Figure 7.8: Total Execution Time Comparison Chart 33

**List of Tables**

1. Table 3.1: Literature Survey Summarization 7

2. Figure 6.1: ACO set parameters 28

3. Figure 6.2: Task and VM set parameters 28

**Abbreviations**

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| VM | Virtual Machine |
| PM | Physical Machine |
| ACO | Ant Colony Optimization |
| GA | Genetic Algorithm |
| PSO | Particle Swarm optimization |
| RR | Round-Robin |
| MCT | Minimum Completion Time |
| MET | Minimum Execution Time |
| PE | Processing Element |
| CPU | Central Processing Unit |
| MIPS | Million Instructions Per Second |

**Chapter 1**

**INTRODUCTION**

This chapter is intended to give the problem introduction along with the context of the problem and justification of why solving this problem would be helpful for the users. This chapter also discusses the background of why the problem came to be, and the motivation behind the proposed solution.

**1.1 Problem Definition**

Task Scheduling in cloud is an discrete optimization problem and many static and dynamic heuristic algorithms have already been proposed to solve this particular problem. A good task scheduling system should be able adapt its scheduling strategy based on the type and nature of tasks. The major objective of this project is to design an algorithm for the proper scheduling of the on-demand tasks by the users in the cloud environment by doing the optimal resource allocation for the tasks on its size as in this project the nature of task is considered to be independent[[1].](https://www.researchgate.net/publication/350192028_QRAS_efficient_resource_allocation_for_task_scheduling_in_cloud_computing)

The main goal of this proposed algorithm is to optimize the makespan and execution time of the tasks on the entire system by maximizing the availability of virtual machines and assigning the resources to the tasks based on cost consumption required by that particular task which will maintain the load balance of virtual machines. In this paper we are going to implement a heuristic task scheduling algorithm which is based on Ant-Colony Optimization (ACO) technique for scheduling the requested tasks by evaluating the execution time and completion time of the particular task on the assigned virtual machine and also ensuring the efficient utilization of available resources. Further, this proposed algorithm is tested against traditional ACO approach and other heuristic method known as Genetic Algorithm, to evaluate its performance on the Cloudsim platform (which is a simulated cloud environment).

**1.2** **Background**

A cloud computing system is generally defined as a type of parallel and distributed system that consists of an interconnected collection of virtual computers that are dynamically provisioned and presented as one or more unified computing resources according to service level agreements (SLAs) between a service provider and end-user.

Virtualization is one of the latest technology, where all the available physical resources are virtualized. It was developed to share resources and increase the efficiency of data sharing. In virtualization, resources like memory, CPU, software(s) are virtualized in the cloud environment. In application level of cloud infrastructure, an application or a system is virtualized in form of virtual machines (VM), here the virtual layer sits on top of the operating system. Virtualization is mainly employed to map the incoming requests arising from different users and charge them as perthe use-case. Here, the main challenge is to provide the resources for all user requested tasks dynamically while maintain Service level Agreement (SLA) [[11]](https://doi.org/10.1109/TCC.2017.2648788).

Task Scheduling is mainly the process of allocating the requested tasks by the users in a certain manner so that the available resources are utilized properly. In the cloud, users can lease computing power in the form of virtual machines (VMs) using virtualization technology. The allocation of available resources to incoming task requests is very challenging as tasks cane be of different nature size and type. Hence, an effective task scheduling algorithm is need for assigning tasks to the virtual machines in dynamic cloud environment. Various techniques and algorithms are already available for task scheduling, but with a fast growth of cloud systemand its users every hour, there is still a space for various improvements in existing system for efficient utilization of cloud resources and minimization of execution time and total completion time (makespan).

.

**1.3** **Motivation**

In cloud environment each type of service requires users to submit requests to the service provider through the internet. In order to efficiently manage computing resources and schedule incoming requests (tasks), service providers need scheduling algorithms. The Scheduling and allocation of cloud computing resources are important factors in cloud computing performance. Because of this, many researchers have been interested in studies of various task scheduling algorithms in cloud computing. Due to the growing demand for cloud services, a very large number of requests are generated by users at a time. Scheduling systems which do not employ efficient scheduling may result in longer waiting times for tasks, and short-term tasks may be terminated due to the waiting period. Therefore for scheduling tasks we need to consider various factors such as the nature of the task, its size, its execution time, availability of resources, and the load on those resources [[9]](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-018-0105-8).

Advantage of proposed work:

* Due to an efficient approach of task scheduling resulting in increased utilization of available resources, the performance parameter such as overall makespan (which is maximum completion time) and total execution time of the allocation of the requested tasks to VMs is reduced.
* Also the service cost will be reduced.

.

**Chapter 2**

**LITERATURE REVIEW**

Several strategies for scheduling has already been proposed by various researchers all around the globe, in which they have discussed about their proposed method, how it is performing against already present methods in different scenarios and what are their future plans regarding the researched work.

* Sukhpal Singh et al. [[11]](https://doi.org/10.1109/TCC.2017.2648788) has proposed an autonomous resource management technique which main focus is to reduce the Service Level Agreement (SLA) violations made between service provider and the user for the optimal use of cloud services by fulfilling the user’s Quality-of-Service (QoS) requirements. Here the proposed technique considered various QoS parameters which mainly includes execution time, execution cost, deadline constraint availability of resources to analyze the impact of different Quality-of-Service parameters on SLA violation rate. In future work the authors in this paper aspires to also consider resource utilization and scalability factor as parameters for execution of workload to maintain low SLA violation rate.
* Through this literature Tahani Aladwani [[1]](https://www.intechopen.com/chapters/71902) has explained in detail about the levels of task scheduling in virtual machines, different task scheduling categories, various static task scheduling algorithms with their advantages and disadvantages of all algorithms in different scenarios. In this research work the author has mainly focused on static scheduling algorithms such as first come first serve (FCFS), shortest job first (SJF) and Max-Min algorithm. Here the performance of different static scheduling approaches is also shown for comparison purpose with the help of clousim simulation. The only con of this paper is that it doesn’t discuss about various dynamic scheduling algorithms which are more efficient in comparison to static algorithms.
* Rodrigo N. Calheiros et al. [[12]](https://www.academia.edu/3232847/CloudSim_a_toolkit_for_modeling_and_simulation_of_cloud_computing_environments_and_evaluation_of_resource_provisioning_algorithms) has done the research work on the ‘Cloudsim’ which is toolkit used for the simulation of cloud computing environments. Here it states that the cloudsim toolkit supports both the system and behavior modeling of various cloud system components such as data centers, virtual machines, resource allocation and task scheduling policies.In the cloudsim, generic application provisioning techniques are implemented that are extensible across various use-case scenarios with ease. Here the author has also described about components of the multi-layered cloudsim architecture. It also mentions about the modeling of cloud and its entities in virtual environment, different provisioning models provided for VM allocation and task scheduling. Further the design, implementation and workflow of different components, modules and entities are mentioned in this paper. Future aim of this research work is to incorporate models for database services like SQL, Quality of service parameters.
* In this research work Safwat A. Hamad et al. [[6]](https://www.researchgate.net/publication/358124000_AdPSO_Adaptive_PSO-Based_Task_Scheduling_Approach_for_Cloud_Computing) has proposed a task scheduling algorithm based on Genetic Algorithm for the allocation and allocation of tasks to the resources. Aim of the author is to optimize the completion time as well as the cost of tasks. Genetic Algorithm is mainly based on the biological concept used for generating the population. This approach works on the idea that after the each selection from a population, there will be a solution which makes the cut for good fitness function but it’s not directly selected for crossover process. The obtained solution is chosen and then added to the list of population before the next iteration. Hence after the max iteration the optimal solution is generated. The author has compared the GA algorithm with RR in terms of completion time and task cost. Future aim of this work is to add dynamic characteristics of VM and also include more parameters for better optimization.
* In this research work Said Nabi et al. [[7]](https://www.researchgate.net/publication/358124000_AdPSO_Adaptive_PSO-Based_Task_Scheduling_Approach_for_Cloud_Computing) has proposed an adaptive Particle Swarm Optimization (PSO) based load balanced task scheduling method. According to the article, Swarm intelligence-based heuristic algorithms are better suited for cloud computing. The optimization process of PSO approach consists of two major components called local and global search. In order to achieve an optimal mapping approach between incoming tasks and available resources, a balance between global and local search is crucial. In this case, scheduling the tasks in advance reduces the task execution time, increases throughput, and increases resource utilization. There are generations and particles in the swarm, where the generation shows the number of iterations that were required to reach the optimized solution, and each particle represents a single solution within a generation. In this paper the performance of the discussed approach has been compared with different inertia based PSO strategies.

**Summary of the Literature Survey**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Author’s | Discussed algorithms | Methodology | Parameters | Advantages | Disadvantages |
| Tahani Aladwani  [1] | FCFS | FIFO queue is used to schedule tasks, where the first tasks are executed first in the VM. | Arrival Time | Simple to implement | Ignorance of other criteria than arrival time. |
| Subham Mittal et al. [2] | RR | A particular task is allocated resources for a given time quantum, once the task is queued and the next chance for execution arises. | Arrival Time, Time Quantum | Good response time, load is balanced and less complex | Pre-emption causes the process out and time slice expires |
| Ibrahim Thiyeb et al. [3] | Min-Min and Max-Min | By using Min-Min, the smallest task is selected and assigned to VM, which gives it the shortest completion time.  Max-Min assigns the longest task to VM which gives the shortest completion time. | Makespan,  Expected completion time | Better makespan | Load imbalance and QoS parameters are not considered |
| Nigel Thomas et al. [4] | MET    MCT | This algorithm determines which resource will give the lowest execution time for a task.  Among the available VMs, MCT assigns the tasks based on expected minimum completion times. | Expected execution time.  Expected completion time & Load Balancing | Selects fastest resource for scheduling.  Load balancing is considered | Load  imbalance    No optimization in selection of best resource. |
| Safwat A. Hamad et al. [6] | Genetic Algorithm | Assigning tasks to resources is done here according to the fitness function value for each parameter of the task scheduling process. | Performance, Domain, Population | Needs fewer resources, fast execution. | Very difficult to make changes in parameter |
| Said Nabi et al. [7] | PSO | It uses population to find optimal minimum values to create correct order of task for suitable scheduling to resources. | Inertia weight, personal best, global best | Finds optimal solution, easy to implement | Parameter selection issue & slow convergence speed |
| Medhat A. Tawfeek et al. [5] | ACO | The ACO algorithm simulates the behavior of ant colonies. | Pheromone updation,  load  & makespan | Faster performance, high utilization of resources. | Time-consuming to update pheromone trail |

Table 2.1: Literature survey summarization

**Chapter 3**

**PROBLEM STATEMENT**

The major objective of this project is to propose an efficient scheduling algorithm which is based on ACO technique for the allocation of requested heterogeneous independent tasks to the most effective available virtual machines.

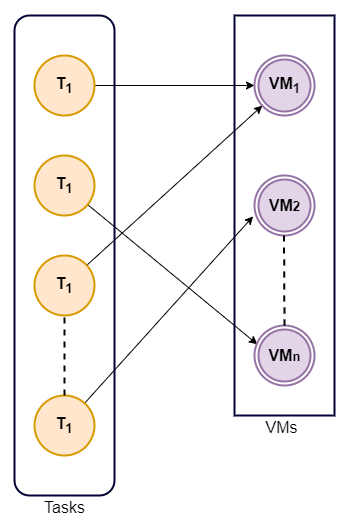
Objectives:

* Minimizing the completion time (makespan) while maintaining load balance.
* Minimizing the Execution time.

The problem can be represented in a form of graph like:

**G = (N,E)**

Where, the set of the nodes N represent the virtual machine and set of tasks, whereas set of edges E represents the connection between the task and VM. During the iteration, all ants are allocated randomly to the starting VMs, then they move from one VM to another for each task until they complete a tour (means all tasks are allocated). [5].



**Figure 3.1: Problem Representation**

**Chapter 4**

**RELATED STUDY**

**4.1 ACO Algorithm Overview**

Macro Dorigo in 1992 in his PhD thesis, which is a meta-heuristic algorithm, based on the real and intelligent behavior of ants for their food search. On the basis of various information (distance, amount of pheromone on the path) ants evaluate the quality of the path and choose one of them based on highest transition probability where the path which has probability means it has more amount of pheromone. These pheromones also get evaporated to help the ants to choose different paths and eliminate the longer distance paths. This algorithm is mostly useful for solving discrete optimization problems. A simple example of creation of pheromone trail for the search of optimal path is shown in figure 2. In figure (a) we see a path leading from the nest to food, in figure (b) an obstacle blocks the path, and soon ants spread on both sides of the obstacle because there is no clear trail to follow as shown in figure (c). However, when ants move around obstacles, they are able to detect the previous pheromone trail again, so a new trail forms. The shortest path will have a stronger trail than the longest path due to the presence of more pheromone in the shorter path [14].

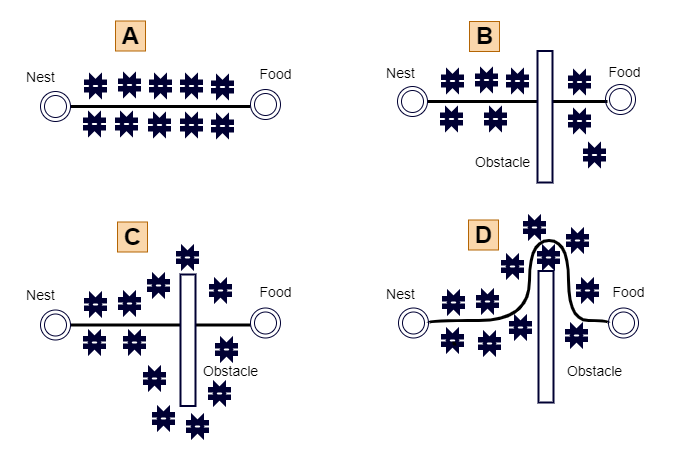


Figure 4.1: Ant Colony Optimization Example

**4.2 ACO Parameters**

To adapt the ACO algorithm into the scheduling problem, some of the most important parameters are defined:

1. Number of ants.
2. Evaporation factor (ρ) of pheromone (from the range 0:1).
3. α – the amount of pheromone level on the existing path
4. β – ‘quality’ factor which decide the next step to be taken .
5. γ – the load balancing factor for calculating the load on VMs.

Consideration while choosing these factors involves:

1. If number of ants is too big then the individual cycle of algorithm will last very long. Whereas, when ant size is too small, then most of the paths will remain uncovered and it may prevail from getting the optimal solution.
2. If the evaporation factor is too small then ants might easily forget the good solutions. In contrast, if the value is too high, the ants will continue to analyze the weak solutions, and the new solutions will not be implemented [15].

**4.3 Scheduling Policies in Cloud**

For scheduling in cloud systems, there are primarily two policies: time-sharing and space-sharing.

* Space -Shared policy: A VM or task can be executed on a host or VM at a given instance at a given time with this policy. It is also known as batch process scheduling.
* Time – Shared policy: This policy allows multiple VMs or tasks to do multitask and run simultaneously with a host or VM. Time is shared among VMS or tasks. It is also referred to as round-robin scheduling [[10]](http://paper.ijcsns.org/07_book/201802/20180214.pdf).

Four Different Scenarios of scheduling in cloud:

Consider two VMs each requiring two CPU cores on a host with two CPU cores [8].

1. **Space shared policy for both VM and Task:** Every listed task will get allocated inside the virtual machine for the execution. Firstly only two tasks will run and rest of the two will have to wait for the execution of first two tasks to be completed. Similar process will continue for tasks in VM2.



Figure 4.2: Space shared policy for both VM and Task

1. **Space-shared policy for VM & time-shared for task:** VM1 is going to be allocated first to the cores at a specific time & tasks will get assigned on virtual machines respectively.

Figure 4.3: Space shared policy VM and time-shared for Task

1. **Time-shared policy for VM & space-shared for task:** Here, VM1 and VM2 share a segment of time with each other. In this case, each slice will receive a single task at a time, while others will wait until each task is completed.

****

Figure 4.4: Time shared policy VM and space-shared for Task

1. **Time-shared policy for both VM and task:** The virtual machines in this case share the same slice of time on every core, and all tasks are scheduled at the same time.

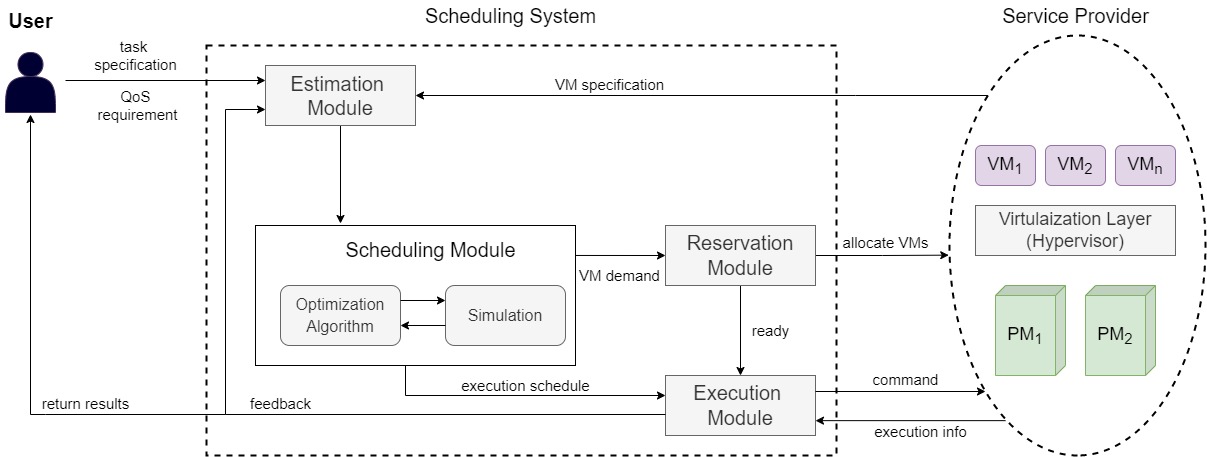
****

Figure 4.5: Time shared policy for both VM and Task

**4.4 Architecture of Scheduling in Cloud**

The steps to be followed to schedule the requested tasks by users in the cloud are:

* Firstly the estimation module will acquire the task specifications and QoS requirement (deadline constraint) from the user. It will also acquire VM specification from the cloud service provider.
* Estimation module estimates execution times for each task on each VM based on the collected information. The scheduling module then determines which tasks should be assigned to which VM based on the estimated execution time matrix. Scheduling module has two entities: optimization algorithm and simulation. Optimization algorithm will generate the schedules which will be given to the other part which is simulation component to check whether the generated schedules are satisfying the given constraints or not. The optimization algorithm will keep finding better schedules until the stop criteria (maximum number of iterations) are met.
* As soon as the scheduling module has determined an optimal schedule, it informs the reservation module of the amount and type of virtual machines required. The reservation modules now lease VMs from the service provider.
* When the runtime environment is ready, the execution module starts allocating the tasks to the respective VMs based on the given optimal schedule. In addition to collecting information about real execution times, it also gives information to the estimation module, so that when similar tasks are encountered, the estimated execution time will be more accurate. At last the execution module will return the final result to the user [10].

****

**Figure 4.6: Scheduling Architecture of Cloud**

**Chapter 5**

**METHODOLOGY**

In this work, an efficient scheduling algorithm is proposed based on the Ant-Colony-Optimization (ACO) approach for the allocation of incoming independent tasks to the VMs. As VMs may have different processing powers, so to these VMs the requested tasks are allocated based on the task size and load balance level of the VMs, to achieve our main goal minimization of total completion time (makespan) and total execution time [9].

**5.1 Proposed Work**

The problem of scheduling the independent tasks to the best available virtual machine is done in two divided parts:

1. Algorithm for Initialization and Assigning of Tasks based on given input parameters.
2. Algorithm for allocation of tasks to VMs based on load balanced factor ACO.

**5.1.1 Definition of Performance Metrics:**

* **Task:** Let T = {t1,t2,…tn} represent a set of total tasks, n = total number of tasks each have unique task id and length associated with it.
* **Resource:** Resources which are VM is defined as VM = {M1, M2, M3,...Mn}

and m = total no of virtual machines each consisting of VM id, No. of MIPS instructions (set at random), bandwidth etc.

* **Execution Time Calculation:** It is denoted by ET(ti,mj) which gives the estimated time for execution of task i on VM j.

ET = length of task(ti) / Computation cost of mj

- (1)

* **Completion Time:** It is denoted by CT and defined as the time taken from the arrival time of the task in addition with the execution time of that task on the particular VM.

CT = Arrival Time(Ti) + ET(ti,mj)

- (2)

* **Makespan:** Makespan is the maximum completion time of all tasks allocated to different VMs.

Makespan = max (CT(ti, VMj)

- (3)

* **Load Balancing Factor:** It is denoted by BLF and is used to choose the resource with currently minimum utilization for obtaining better load balancing which eventually helps in obtaining minimum makespan time.

LBF(mi) = Total Estimated ET of all VMs/ EET of mi

**-** (4)

* **Pheromone Initialization:** A small positive value 0.5 is set as the initial value of pheromone trail τ0 in the pheromone matrix, and as ants travel along the path, the trail is updated for the next ants.
* **Transition Probability Calculation:** After initialization of all parameters each ant k makes a tour by executing all the n tasks with the help of calculated probability.

Probability = ηβ \* τα / ∑i=0 to m ηβ  \* τα

- (5)

Where, η = heuristic information, α = total amount of pheromone value on the path & β= quality factor next step.

* **Pheromone Updating:** Once the tour is completed by ants, each ant k lay an amount of pheromone on the path calculated by: **-** (6)

∆τ = 1 /max(CT)

* **Local Pheromone Update:** Once the tasks are mapped to VMs based on the above probability, then the pheromone trail is updated by:

τik = (1-ρ) \* τik + ρ \* ∆τ0 , 0 < ∆ 1

**-** (7)

Where, ρ is the evaporation rate, τ is the local pheromone value and ∆τ0 is the change in localpheromone value.

* **Global Pheromone Updating:** Once all the ants have constructed their solution, pheromone values are increased by updating the ant's global pheromone rule on the edges. The obtained solution is mainly the most optimal one which is obtained through the path of best ants.

Pheromone𝑖𝑗𝑘 = ((1 − 𝜀) × (Pheromone𝑖𝑗𝑘 )) + (𝜀 × 𝜃)

- (8)

Where, 𝜀 is the global evaporation rate, and 𝜃 is a coefficient that determines the amount of pheromone to be deposited [13].

**5.1.2 Assumptions made for Proposed Algorithm**

Some of the critical assumptions made while implementing the proposed algorithm are:

* The proposed algorithm will comprise of only independent tasks.
* Due to independence task do not have any priority order associated with them.
* VM are also heterogeneous here, meaning no of MIPS instructions varies.
* Estimation of the execution time of tasks on each VM will be done initially.
* Each VM do the execution of only one task at a time.

**5.1.3 Algorithm I:**

Suppose there are m virtual machines in the system and each VM is assigned one separate task at a time. Tasks are executed for a specific time that is determined by the length of the task and the computation cost of VM to perform that task.

Procedure followed:

* Input the total no tasks and total number of resources as VMs. Assign the given tasks by user to a created list based on the arrival time of each task.
* Let m = number of available VMs and n denote the total number of tasks stored in task list.
* Check if number of task is greater than number of available VMs or not.
* If number of tasks is less than directly allocate all the tasks from stored list to m available VMs. Else if no of task is greater than available VMs, and then apply the proposed ACO procedure for the allocation of all the input tasks to the best possible resources
* **Pseudo code for Algorithm I**

Input: List of Tasks and Resources as VMs. **Procedure:**1. *Create a list to store given tasks*2. *Store the given tasks in the created list based on its arrival time*3. m = *no. of available VMs* n=|Task List (t1,t2,.. tn)| - //*total no. of tasks*4. Do5. If(n>m) *Run proposed algorithm on stored task and VMs*

6. Else *Give all n tasks from Task List to m VMs*7. End If8. While (*Task List != NULL or more incoming tasks*)

**5.1.4 Algorithm II:**

With the help of proposed approach, tasks are allocated to VMs in such a way that it reduces the overall makespan by keeping the load balancing factor among VMs thereby ensuring the efficient resource utilization. In order to achieve all objectives, the mapping of tasks to VMs is done using ACO [9].

The proposed ACO procedure works in following procedure way:

* Firstly we initialize the pheromone trail value, and set all the ACO parameters initial value and maximum number of iteration.
* Two matrices are created one is Expected execution time (EET) matrix and other is pheromone matrix, Start time of first task is initialized randomly and thereafter the EET is calculated with help of eq. (1).
* Initially all ants are placed in random order.
* Then all ants travel in a defined sequence. Each of the tasks on the task list is then mapped to one of the resources needed for that task.
* Once every task has been mapped to a particular resource, the solution for the ant comes to an end. Each task which is allocated to a particular VM finds the pheromone and heuristic information.
* Resources are chosen based on the high transition probability calculated by eq. (5). For successive ants coming one after another the initial pheromone value will be updated based on pheromone value of previous ants with help of eq (7).
* Local best solutions with makespan calculated with eq. (3) and load balancing factor calculated by eq. (4) will be obtained.
* At last of each iteration, the global solution will be updated based on the pheromone matrix using eq. (8) and best allocation schedule is obtained.
* The whole procedure will repeat till maximum no of iteration defined and finally the optimal solution is found in global solution.
* **Pseudo code for Algorithm II**

Step 1: Initialization: *Optimal solution = NULL, a= 1, B=5, Max no of iteration = 100 given initial pheromone = 0.5*

Step 2: *Evaluation of execution time based on task computing cost and VM computing capability as per eq. (1)*

Step 3: *Run the loop till max no of iterations*

Step 4: *Calculate probability with help of eq. (5)*

Step 5: *Assign task Ti to the VM based on high probability and update mapped task completion time according to equation. (2)*

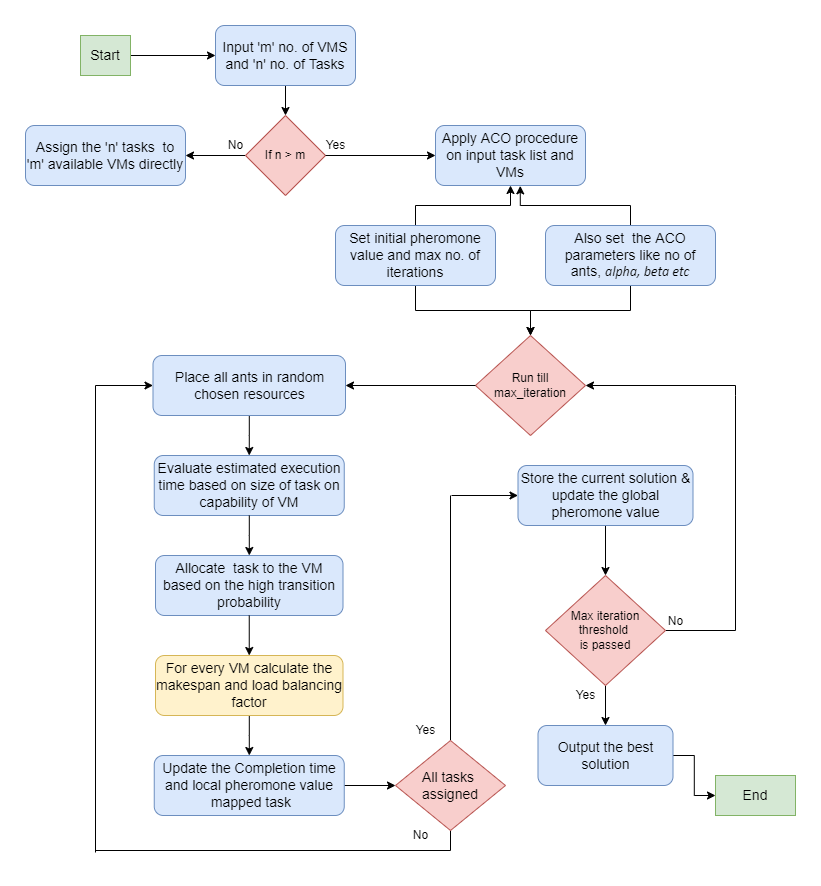
Step 6: *For each iteration makespan is calculated and load balancing factor as eq. (3) & (4)*

Step 7: *Update local pheromone value for allocated task based on eq. (6) & (7*)

Step 8: *Store the current solution and Update global pheromone value as per eq. (8)*

Step 9: *If the max iteration condition is completed then return the optmal solution of allocated tasks to respective VMs.*

**5.2 Flowchart of Proposed System**

 Figure 5.1: Flow chart of the proposed Algorithm

**Chapter 6**

**IMPLEMENTATION**

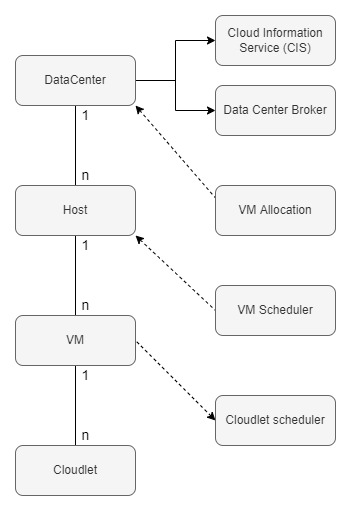
The proposed algorithm will be implemented with the help of CloudSim toolkit, which is used for modeling the cloud environment.

**6.1 Experiment Tool: CloudSim**

CloudSim is the most popular simulator for testing algorithms and techniques used in cloud computing. With this toolkit, users can seamlessly model, simulate, and experiment with cloud based infrastructures. This toolkit also helps to compare the performance of different approaches based pre-defined parameters [5].

**6.1.1 Components of Cloudsim**

* **Datacenter:** An entity that models the infrastructure-level services provided by cloud providers. Furthermore, every datacenter component serves as a central controller for generalized application provisioning that enforces a set of policies for the allocation of bandwidth, memory, and storage devices to hosts and VMs.
* **CIS:** Cloud information service is an entity which registers datacenter entity and check for resources.
* **Datacenter Broker:** Brokers are principally responsible for querying CISs to determine that the appropriate allocation of resources or services will fulfill the high quality of service specified by users.



* **Host:** Used to model a physical server. It contains information like how much memory is available, how much storage is available, how many CPU cores are available, and the type of allocation policy used.
* **VM:** This entity is used to simulate VM which is used for allocation of cloudlet. Every VM component stores parameters such as memory size, storage size, number of MIPS, bandwidth etc. IT also has access to VM internal provisioning policy.
* **VM Allocation:** As its primary purpose, it identifies the best available hosts within a data center that meet the VM mapping requirements and have adequate memory, storage and availability.
* **VM Scheduler:** In this class, host components implement space-sharing and time-sharing policies to allocate processor cores to VMs. It defines the rules for processor allocation to VMs.
* **Cloudlet:** This component model defines attributes of tasks such as size of task, input file size, output file size, no of processor required etc. Here the requested tasks are represented as cloudlet.
* **Cloudlet Scheduler:** It is also an abstract class which is responsible for sharing the total processing power needed between all the cloudlets assigned in that VM.

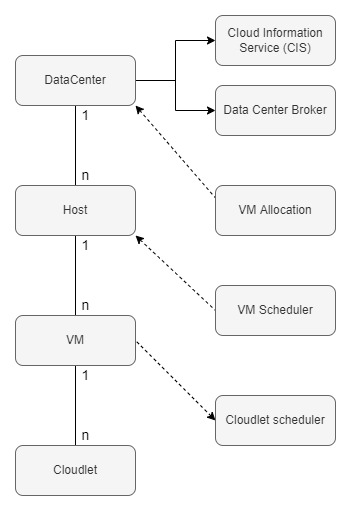
****

Figure 6.1: Relation between CloudSim Entities

**6.2 Stepwise procedure for algorithm implementation**

**Step 1:** Input the total number of heterogeneous tasks to be executed and number of virtual machines needed from the user.

**Step 2:** Initialize the cloudsim package which contains: no of users, calendar and a flag variable.

**Step 3:** Create datacenter entity which will be act as a resource provider (resource is VM in this case).

**Step 3.1:** First a list is created to store the host, which is an interface class used to implement the basic features of a virtual machine inside a datacenter.

**Step 3.2:** Create another list to store the PE. Processing element (PE) class here represents number of CPU core for a physical machine defined in terms of MIPS.

**Step 3.3:** Create the PE’s with already defined provision in cloudsim and store it into the created list of PE.

**Step 3.4:** Now the host is created with a unique id, list of PE/CPU core, ram, storage, and bandwidth. The used allocation policy to assign PE to the host is space-shared policy. Due to this only one CPU core is given to each host, hence as a result just one of the VM executes on each host at a particular time.

**Step 3.5:** Now the datacenter characteristics are defined which stores the properties of a datacenter. The characteristics include: architecture, OS, time zone, type of allocation policy and cost of processing.

**Step 4:** Create a datacenter broker, which gets the list of tasks and VMs from the users, creates it and then sends them to the datacenters to be executed. This broker will be used to allocate the tasks to the best VMs based on the proposed algorithm.

**Step 4.1:** We create a broker class with some set parameters of ACO algorithm. Broker class will accept parameters such as: name of broker, number of ants, initial pheromone value, value to compute the quality factor which will decide the next best route to be taken, value to compute the load balancing level of each particular resource at that particular time, and lastly the value of evaporation factor used for updating the pheromone trail.

**Step 4.2:** We will first process different events such creation of VM and task list and binding of those VM and task and sending those to the datacenter for the processing.

**Step 4.3:** In submit cloudlets method inside the broker class we have defined the implementation of proposed ACO algorithm. Here first we will create an object of ACO algorithm class to implement that particular algorithm based on the given ACO parameters. A hash map is created to store the list of tasks allocated to most appropriate VM based on the ACO algorithm with the help of defined maximum number of iterations for the ACO algorithm. Lastly we will submit the list of VM which contains the tasks allocated inside them to the datacenter.

**Step 5:** Create the VM with characteristics such as size, ram, bandwidth, MIPS (randomly to make VM power heterogeneous), and number of CPU. Similarly we will create Cloudlet with parameters such as length (given randomly to make size of task heterogeneous), input and output file size of cloudlet and utilization model.

**Step 6:** Submit the cloudlet list and VM list to the created broker for further execution.

**Step 7:** Stat the simulation of set architecture in cloudsim environment.

**Step 8:** Print the result when the simulation is over.

**Step 8.1:** Result will contain all log information such as starting of datacenter, creation of VMs in datacenters, allocation of the created VM to the defined host in datacenter, sending the cloudlets to broker, and allocation of those received cloudlets to the particular VM based on implemented ACO algorithm.

**Step 8.2:** Print the output with all the details.

**Step 9:** Stop the simulation.

**6.3 Implementation flow diagram**

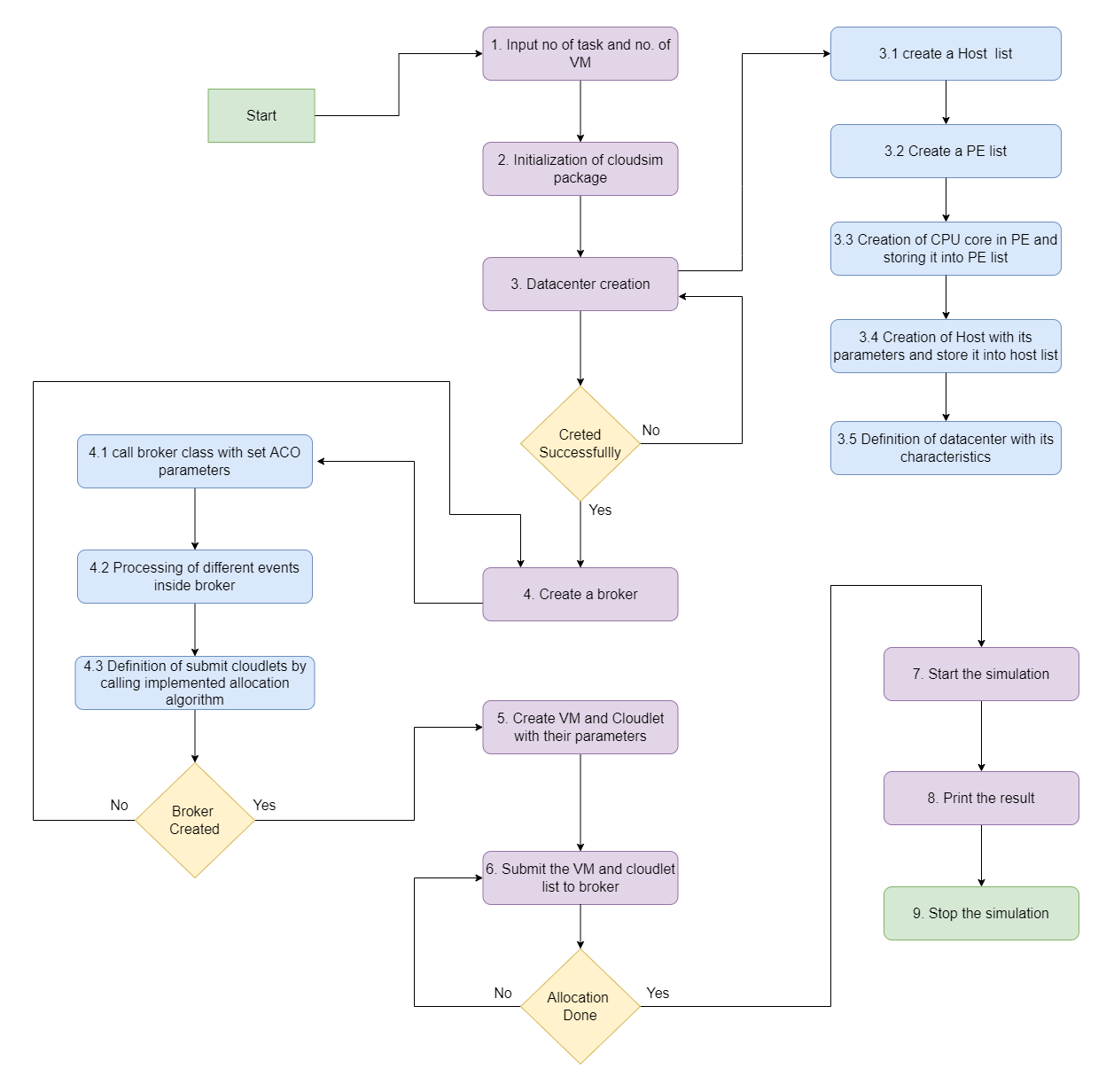
****

Figure 6.2: Implementation Flow Diagram

**6.4 Experimentation set parameters:**

The implemented task scheduling algorithm based on load balanced level ACO has been tested against tradition ACO and GA algorithm based on some set parameters defined below:

**6.4.1 ACO predefined values:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No. of**  **Ants** | **Initial Pheromone**  **Value** | **Total Pheromone Amount on path** | **Next best route selection**  **factor** | **Load balance level factor** | **Evaporation factor** |
| **m** | **Q** | **α** | **β** | **γ** | **ρ** |
| 30 - 50 | 0.5 | 2 | 1 | 4 | 0.1 |

Table 6.1: ACO set parameters

**6.4.2 VM and Task parameters:**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Parameter** | **Value** |
| **Task(as Cloudlet)** | **Task length** | **100 - 12000** |
| **Total no. of task** | **40 – 500** |
| **Virtual machine (VM)** | **MIPS** | **700 - 1500** |
| **No. of VMs** | **10 – 50** |

Table 6.2: VM and Task parameters

**Chapter 7**

**RESULTS & COMPARISION**

**7.1 Results**

As the main objective of our proposed scheduling algorithm is to optimize the execution time and makespan (total completion time of all tasks on all VMs). So we performed the execution on different set of inputs and predefined ACO parameters as per section 6.3.1.

* In Figure 7.1, it shows that the makespan is ranging from 27 to 120 when task size is in between 40 and 160. Here the no. of VM used for allocation used is 10. This chart shows that since the number of resource available (VM) is less in this case hence, the makespan is increasing sharply when task size is large.

Number of VMs = 10

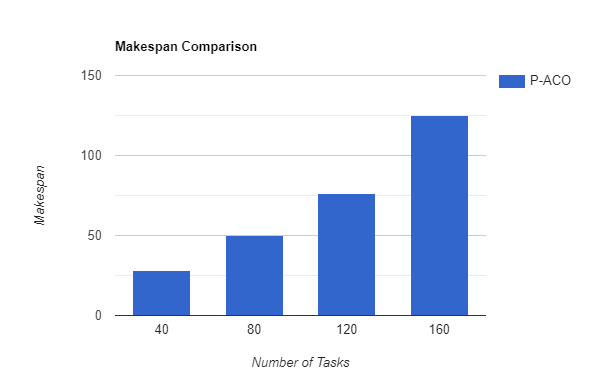


Figure 7.1: Makespan Chart I

* In Figure 7.2, it shows that the makespan is ranging from 22 to 92 when task size is in between 100 and 400. Here the no. of VM used for allocation used is 40, hence the resulting makespan time is less in comparison to the above chart as there is more number of available resource for the execution of tasks.

Number of VMs = 40

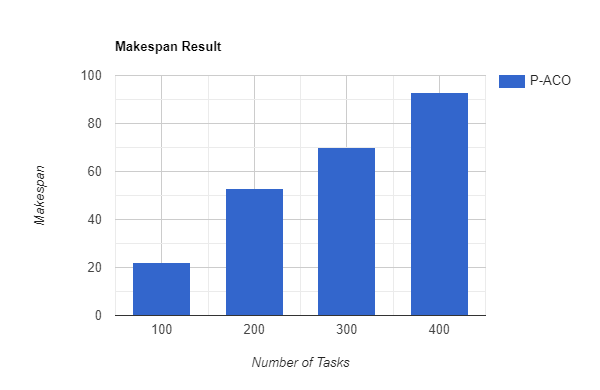


Figure 7.2: Makespan Chart II

* In Figure 7.3 shows the total execution cost of task execution is ranging from 200 to 810 when size of task is in between 40 and 160.

Number of VMs = 10

****

Figure 7.3: Total Execution Time Chart I

* In Figure 7.4 shows the total execution cost of task execution is ranging from 440 to 2010 when size of task is in between 100 and 400. This chart shows that the increase is total execution time is also very stable in our proposed algorithm.

Number of VMs = 40

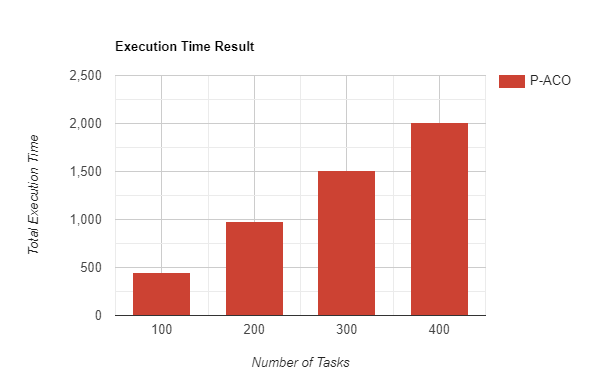
****

Figure 7.4: Total Execution Time Chart II

**7.2 Comparisons**

* Figure 7.5 shows how our proposed algorithm (P-ACO) is performing in the makespan category in comparison to the basic ACO and GA algorithms. The line chart clearly shows that the proposed algorithm clearly outperforms the GA algorithm and its makespan time is also slightly less in when paired up against the basic ACO algorithm when the task size is varying from 40 to 160.

Number of VMs = 10

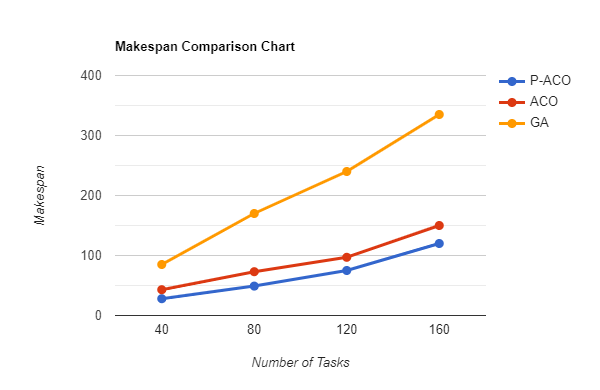
****

Figure 7.5: Makespan Comparison Chart I

* Figure 7.6 shows how our proposed algorithm is performing even more efficiently in comparison to basic ACO algorithm when the task size is large.

Number of VMs = 40

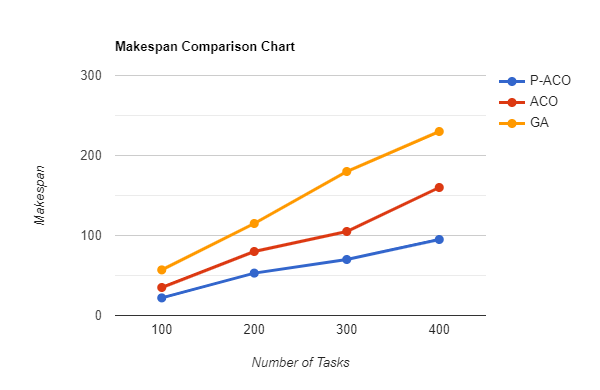
****

Figure 7.6: Makespan Comparison Chart II

* Figure 7.7 shows how the comparison of total execution time between the three algorithms. This chart is displaying that execution time most study and less in our proposed algorithm when compared with other two when the task size is ranging from 40 to 160.

Number of VMs = 10

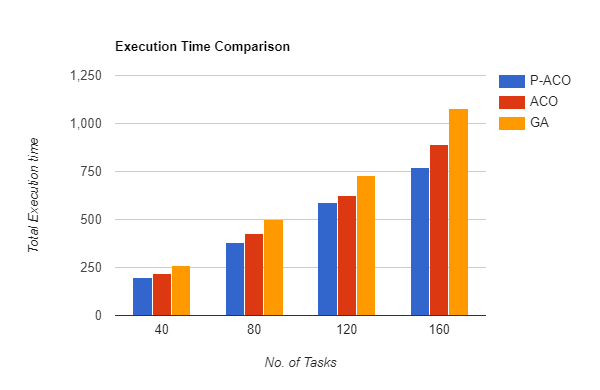
****

Figure 7.7: Total Execution Time Comparison Chart I

* Figure 7.8 displays the comparison of total execution time between the three approaches. This chart is to verify that total execution time is less than other algorithms even when the task size is large as in this case.

Number of VMs = 40

****

Figure 7.8: Total Execution Time Comparison Chart II

**Chapter 8**

**CONCLUSION**

In this project, an optimized task scheduling algorithm for the allocation of independent tasks to the best available resource has been proposed and implemented which mainly achieves the two major objectives of optimization of makespan and total execution time with the help of load balance factor. And with the minimization of makespan and checking of load balance level of VMs, we insure the efficient utilization of available resources. The proposed method uses the ACO technique to obtain the optimal solution for allocation of tasks to VMs while also considering the load balance level of VMs, which makes the proposed algorithm slightly different from the traditional ACO method where makespan is the only considering factor while allocating tasks.

The proposed algorithm is implemented using the Coudsim toolkit and later tested against the other heuristic scheduling approach like GA and basic ACO. In the comparison it is found that the proposed approach clearly outperforms the GA algorithm in both makespan and total execution time parameters. The comparison confirms that our proposed technique is performing better than the traditional ACO in different task and resource size scenarios.

Later, for the future work we intend to consider the dependent nature of tasks by optimizing the proposed method. It also aims to consider the QoS parameters of users during the scheduling as it is most important in real world scenarios. At last in future we also aim to test our algorithm with other optimized scheduling approach like PSO etc in even large use case scenarios for better performance analysis.

**REFERENCES**

[1] Aladwani, Tahani. "Types of task scheduling algorithms in cloud computing environment." Scheduling Problems-New Applications and Trends. IntechOpen, 2020.[2] Mittal, Shubham, Swati Singh, and Ramandeep Kaur. "Enhanced round robin technique for task scheduling in cloud computing environment." (2017).[3] Syed, Ibrahim. "HAMM: A Hybrid Algorithm of Min-Min and Max-Min Task Scheduling Algorithms in Cloud Computing." International Journal of Recent Technology and Engineering (IJRTE) 9 (2020): 209-218.[4] Llwaah, F., N. Thomas, and J. Cala. "Improving MCT scheduling algorithm to reduce the makespan and cost of workflow execution in the cloud." UK Performance Engineering Workshop. Newcastle University, 2015.[5] Tawfeek, Medhat A., et al. "Cloud task scheduling based on ant colony optimization." 2013 8th international conference on computer engineering & systems (ICCES). IEEE, 2013.

[6] Hamad, Safwat A., and Fatma A. Omara. "Genetic-based task scheduling algorithm in cloud computing environment." International Journal of Advanced Computer Science and Applications 7.4 (2016): 550-556.

[7] Nabi, S., Ahmad, M., Ibrahim, M. and Hamam, H., 2022. AdPSO: adaptive PSO- based task scheduling approach for cloud computing. *Sensors*, *22*(3), p.920.

[8] Almezeini, Nora, and Alaaeldin Hafez. "Review on scheduling in cloud computing." *Int J Comput Sci Netw Secur* 18 (2018).

[9] Gupta, Ashish, and Ritu Garg. "Load balancing based task scheduling with ACO in cloud computing." 2017 International Conference on Computer and Applications (ICCA). IEEE, 2017.[10] Jia, Y.H., Chen, W.N., Yuan, H., Gu, T., Zhang, H., Gao, Y. and Zhang, J., 2018. An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(1), pp.634-649.

[11] Singh, S., Chana, I. and Buyya, R., 2017. STAR: SLA-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing*, *8*(4), pp.1040-1053.

[12] Buyya, R., Ranjan, R. and Calheiros, R.N., 2009, June. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *2009 international conference on high performance computing & simulation* (pp. 1-11). IEEE.

[13] Ghazipour, F., Mirabedini, S.J. and Harounabadi, A., Presenting a New Ant Colony Optimization Algorithm (ACO) for Efficient Job Scheduling in Grid Environment.

[14] Yoo, Myungryun, and Takanori Yokoyama. "ACO Based on EDF for Real Time Task Scheduling." (2019).

[15] Drabowski, Mieczysław, and Edward Wantuch. "Scheduling in Manufacturing Systems–Ant Colony Approach." In *Ant Colony Optimization-Techniques and Applications*. IntechOpen, 2013