VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Investigations on Opulence Split Algorithm and Particle Swarm Optimization Algorithm for Task Scheduling in a Virtualized Cloud Environment

A J-Component Report

submitted by

Faraz Ahmad

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

Professor Jothi Kr

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

APRIL 2019

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project entitled **"Certain Investigations on Opulence Split Algorithm and Particle Swarm Optimization Algorithm for Task Scheduling in a Virtualized Cloud Environment"** submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Vellore-14 towards the partial fulfilment of the requirements for the course CSE4011- Virtualization is a record of bonafide work carried out by me under the supervision of Professor Jothi Kr**.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for any other course or purpose of this institute or of any other institute or university.

Faraz Ahmad

Faraz Ahmad

16BCE0920

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled "**Certain Investigations on Opulence Split Algorithm and Particle Swarm Optimization Algorithm for Task Scheduling in a Virtualized Cloud Environment**" is prepared and submitted by Faraz Ahmad, has been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the course CSE4011-Virtualization in Vellore Institute of Technology, Vellore-14, India.

Jothi KR

Guide

(Name & Signature)

ACKNOWLEDGEMENT

The project "**Investigations on Opulence Split Algorithm and Particle Swarm Optimization Algorithm for Task Scheduling in a Virtualized Cloud Environment"** was made possible because of inestimable inputs from everyone involved, directly or indirectly. I would first like to thank my guide, **Prof. Jothi Kr,** who was highly instrumental in providing not only a required and innovative base for the project but also crucial and constructive inputs that helped make my final product. My guide has helped me perform research in the specified area and improve my understanding in the area of web development and internet of things and I am very thankful for her support all throughout the project.

I would also like to acknowledge the role of the HOD, **Prof Santhi V,** who was instrumental in keeping me updated with all necessary formalities and posting all the required formats and document templates through the mail, which I was glad to have had.

It would be no exaggeration to say that the Dean of SCOPE, **Prof Saravanan R**, was always available to clarify any queries and clear the doubts I had during the course of my project.

Finally, I would like to thank **Vellore Institute of Technology**, for providing me with a flexible choice and execution of the project and for supporting my research and execution related to the project.

TABLE OF CONTENT

LIST OF TABLES

TITLE **PAGE**

LIST OF FIGURES

TITLE **PAGE**

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
VM	Virtual Machine
VMS	Virtual Machine Simulator
CPU	Central Processing Unit
RTS	Real Time Scheduler

1. ABSTRACT

With the remarkable development in high-speed Internet technologies, the concept of virtualization and cloud computing has become more popular. Cloud provides convenient and on demand network access for computing resources available over internet. Individuals and organizations can access the software and hardware such as network, storage, server and applications which are located remotely easily with the help of Cloud Service. The tasks/jobs that are submitted to this cloud environment needs to be executed on time using the resources available so as to achieve proper resource utilization, efficiency and lesser makespan. Our proposed work surveys a series of currently existing task scheduling algorithms with respect to their characteristics and come up with a new method of real time task scheduling algorithm based on equal resource split.

Keywords—cloud platform; job scheduling; efficiency; makespan, resource

2. INTRODUCTION

Cloud computing is the current buzzword in the Information Technology industry. With its characteristic behaviour, several features and growing popularity the world seems to be currently shifting towards the cloud platform. The demand for cloud services continues to grow with the passage of time. Cloud Computing alludes to application and services that are executed on distributed networks utilizing virtual assets, regular web protocols and system administration models. In cloud computing every entity from power to infrastructure is provided to the user as a service. The services are classified under three models as follows [1]:-

1. Infrastructure as a service(IaaS) : Delivery of technology as an on demand scalabale service. Provides access to fundamental resources like physical and virtual machines as well as storage.

2. Platform as a service (PaaS): Provides runtime environment for application development and deployment tools. It has a highly scalable multitier architecture. It provides all applications and life cycles from the internet.

3. Software as a service (Saas): Allows the end user to use software applications as a service. Software delivery methodology to provide multi-tenant access to software and its function is remotely a web service.

The other model in cloud computing is the deployment model which can be stated as follows [2]: -

1. Public- This is the most used model of cloud computing. This is used by general public cloud consumers. The provider has its own policies on which various users agree upon and use the service. Examples from day to day life are Google Drive, AWS, Azure, etc.

2. Private- This is the kind of cloud infrastructure which is operated within a single organization for its sole purpose. This kind of cloud model is beneficial for an organization in many ways. More number of in-house resources, cutting the cost for management of it by some third party and full control and access of data are some of them

3. Hybrid- This type of cloud infrastructure is combination of two or more types of cloud deployment models. This increases data and application portability. [3]

4. Community- Many small organizations come together to and create their own cloud infrastructure on settling down to policies applied after each of them agrees to all of them.

2.1.PROBLEM STATEMENT

The success of virtualization and cloud computing makes an increasing number of real-time applications such as signal processing and weather forecasting run in the cloud. With the increasing popularity of virtualization, the number of tasks occurring in the virtualized environments have increased at an enormous amount. The proposed work surveys a series of currently existing task scheduling and resource allocation algorithms with respect to their

characteristics and performance in order to establish a paradigm for efficient task management in a virtual cluster environment. The proposed work includes analysis and survey of various currently existing task scheduling algorithms. The various algorithms will be analyzed for scheduling criteria which are later used to find out the most efficient algorithm. The aim is to successfully implement the task scheduling algorithms and come up with a new improvements algorithm. There is a paradigm shift by using technology as a utility by end users. Cloud computing paradigm is following the characteristics of utility based computing. Under the category of distributed computing, utility computing includes grid and cloud computing. This research work highlights the efficient task scheduling at the virtual machine level. When tasks are scheduled over the virtual machine users need to allocate the tasks to the respective virtual machines.

2.2.LITERATURE SURVEY

2.2.1. BACKGROUND

High speed internet is giving world many facilities which were almost impossible earlier and today, they have proven to be of utmost importance in the daily life. One of them is cloud computing. It gives access to computing resources available over internet. One of its major benefits is that individuals and organizations can access the software and hardware such as network, storage, server and applications which are located remotely easily with the help of Cloud Service. The tasks/jobs submitted to this cloud environment needs to be executed on time using the resources available so as to achieve proper resource utilization, efficiency and lesser makespan which in turn requires efficient task scheduling algorithm for proper task allocation. Cloud task scheduling is a NP complete problem. In the process of task scheduling, the users submit their jobs to the cloud scheduler. The cloud scheduler inquires the cloud information service for getting the status of available resources and their properties and hence allocating the various tasks on different resources as per the task requirements. Cloud scheduler will assign multiple user tasks to multiple virtual machines. In our project, we plan to survey the existing Optimized Task Scheduling Algorithm which adapts the advantages of various other existing algorithms according to the situation while considering the distribution and scalability characteristics of cloud resources. At present there is not a uniform standard for job scheduling in cloud computing. Resource management and job scheduling are the key technologies of cloud computing that plays a vital role in an efficient cloud resource management. Cloud computing makes collaboration simpler and can reduce platform-incompatibility problems. Since our research for the project aims at the methods of task scheduling in cloud computing, special emphasis will be paid to the scheduling of tasks in the cloud based environment. Cloud Computing is an essential ingredient of advanced computing systems. Computing concepts, technology and architectures have developed and consolidated in the last decades. Many aspects are subject to technological evolution and revolution. Cloud Computing is a computing technology that is rapidly consolidating itself as the next step in the development and deployment of increasing the number of distributed application. To gain the maximum

benefit from cloud computing, developers must design mechanisms that optimize the use of architectural and deployment paradigms. The role of Virtual Machine's (VMs) has emerged as an important issue because, through virtualization technology, it makes cloud computing infrastructures to be scalable. Therefore developing on optimal scheduling of virtual machines is an important issue. The architecture of cloud computing can be broken into three layers of models as stated above. The performance and efficiency of cloud computing services always depends upon the performance of the user tasks submitted to the cloud system. Scheduling of the user tasks plays significant role in improving performance of the cloud services. Task scheduling is one of the main types of scheduling performed. [4]

2.2.2. Existing Task Scheduling Algorithms and Related Work

A Scheduling of tasks along with the allocation of resources are the most important features of the cloud computing environment which directly affect the performance of a system. In order to achieve high throughput, various task scheduling algorithms have been designed and implemented by various researchers for scheduling and scaling of resources in a cloud environment. The adaption of the appropriate algorithm decides the performance of the system. Another concept that has to be kept in mind while designing a scheduling algorithm is the makespan. Makespan refers to the overall time taken by tasks to execute. Some existing task scheduling algorithms can be classified as follows

1. **Min-Min** It is a heuristic algorithm that starts with a set of all unmapped tasks and work by first finding the minimum expected time of all tasks. The task having the minimum expected completion time is selected and assigned to the corresponding resource.

2. **Max-Min** This is commonly used algorithm in distributed environment which is somewhat similar to the Min-min algorithm. In this algorithm, the expected completion time of each task as per the available resource is calculated. A task with overall maximum time of completion is scheduled over a resource with overall minimum execution time. This step is repeated until meta-task is not empty. In this algorithm the waiting time of larger task is reduced.

3. **RASA** A relatively new algorithm named RASA followed the Max-min and Min-min strategy alternatively in order to assign task over available resource. It can be used to implement the advantage of both the existing algorithm, it supports synchronized execution of large and small tasks which avoids delay in execution of larger task.

In cloud computing, job scheduling is done by adapting the existing scheduling algorithms. These algorithms when scheduling tasks follow the same principal in each situation sometimes leading to increase in makespan of the processes. In our project the algorithm which we will implement will try to adapt the advantages of various existing task scheduling algorithms and applies one of them on the basis of the situation and calculations performed. Since high throughput and load balancing is equally important,

the distribution of the tasks over available resources is done in order to achieve lesser makespan. The Total Execution Time over all available resource is calculated for all processes; it tells about the total time taken by individual resource to execute all tasks. Thereafter, subtraction of Execution Time of individual tasks from Total Execution Time is calculated only for fastest resource. This gives us the estimate of the total time taken by the Resource to schedule all task, excluding the task whose execution time has been subtracted. This gives us the task which needs to be migrated from one Resource to another so that makespan can be reduced. **[5]**

Apart from the above basic scheduling processes a number of

optimised algorithms and scheduling policies were implemented and researched upon.

1. Ant Colony optimization: This is an approach for balancing the load between the different nodes of a cloud system. Many modifications have been made in this algorithm by different computer scientists to gain nothing but efficiency in the cloud computing environment. **[6]**

2. Berger model: For the first time, Baomin Xu with assistants proposed an algorithm on considering commercialization and the virtualization characteristics of cloud computing. It classifies the tasks by ordering them in preference order of QoS. It establishes a general expectation function to restrain the fairness of the resources in selection process. **[7]**

3. Round Robin algorithm: The Round Robin algorithm is one with its main focus towards distributing the load equally to all the resources. Using this algorithm, the broker in a cloud platform allocates one VM to a node in a cyclic manner. The round robin scheduling in the cloud computing is very similar to the round robin scheduling used in the process scheduling. The scheduler starts with a node and moves on to the next node, after a VM is assigned to that node. This is repeated until all the nodes have been allocated at least one VM and then the scheduler returns to the first node again. Hence, in this case, the scheduler does not wait for the exhaustion of the resources of a node before moving on to the next. **[8]**

A Survey on Different Scheduling Algorithms in Cloud Computing led us to the establishment of basic ground definitions related to the field of cloud. A basic introduction of Cloud Computing defines it as ‘Computing services delivered to the user over the internet.’ It is basically sharing of resources between people who need to access data, software and even hardware from a browser’s window from different locations at any point of time. Cloud Computing is pay per use service and many major players which provide such services are Google, Amazon and Microsoft. These are called cloud service providers. Cloud architecture has two components, Frontend and Backend. These two components are connected in a network using internet. Front end components are those elements which provide the application and interface to access the cloud platform. For example, A Web Browser.

The two main classification of scheduling techniques which currently exist are as follows:-

1. **Static Scheduling:** All information is known to scheduler about tasks and resources before execution. It has less runtime overhead.

2. **Dynamic Scheduling:** Information about the task components is not known before execution. Task execution time may not be known and is acquired after it is completed. It has more runtime overhead. These algorithms, take into account two major parameters, Task Length and Deadline. Some of those are mentioned below.

1. **Earliest Feasible Deadline:** The task with shortest deadline gets scheduled. It is a dynamic scheduling technique. After completion of one process, the queue is searched for the task which is closest to the deadline and is scheduled next.

2. **Priority Based Job Scheduling Algorithm**: Mathematical calculations are involved. For scheduling, priority is considered and each job request for resources with some priority.

3. **Greedy Based Job scheduling:** The main focus of this algorithm is to reduce completion time and to give faster solution to scheduling problem. This algorithm has proven to be very efficient when cloud computing is used in business purposes.

As the cloud computing technology is changing day by day a lot of new challenges are emerging. One of them is the task scheduling in a cloud computing environment. The main objective of the scheduling is to maximize utilization of resources and to reduce makespan. **[9]**

Cloud computing is an world wide web based development and use of technology and its one among the advanced and future latest computing model wherever applications and knowledge services are make available over the net **[10]**. Cloud Computing is often outlined as a new method of computing within which dynamically scalable and some time virtualized resources are provided as a services over the web. Job scheduling algorithms is one of the most difficult theoretical problems in the cloud computing area. The Job management is that the primary idea of cloud computing systems task scheduling problems are main that relates to the efficiency of the complete cloud computing system. Job scheduling may be a mapping mechanism from users' tasks to the correct choice of resources and its execution. Job scheduling is flexible and convenient. Job scheduling is global centralized. As cloud computing is a computing model that offer the centralized resource by the mirror service to multiple distributed applications. Job scheduling can be dynamically self adaptive - Increasing and shrinking applications within the cloud also be necessary depend on the need. The virtual computing resources in cloud system may additionally expand or shrink at the identical time. The resources are perpetually dynamical, some resources might fail; new resources may take part the clouds or restart.

The set of job scheduling - Task scheduling is split into two parts: one is employed as a unified resource pool scheduling, and primarily chargeable for the scheduling of applications and cloud API; the other is for the unified port resource scheduling within the cloud. Thus in the end it all boils down to make the most out of all the resources at hand and achieve efficiency.

3. OVERVIEW OF CLOUDSIM TOOLKIT

CloudSim provides a generalised and extensible simulation framework that enables seamless modelling and simulation of app performance. By using CloudSim, developers can focus on specific systems design issues that they want to investigate, without getting concerned about details related to cloud-based infrastructures and services. Advances in computing have opened up many possibilities. Hitherto, the main concern of application developers was the deployment and hosting of applications, keeping in mind the acquisition of resources with a fixed capacity to handle the expected traffic due to the demand for the application, as well as the installation, configuration and maintenance of the whole supporting stack. With the advent of the cloud, application deployment and hosting has become flexible, easier and less costly because of the pay-per-use chargeback model offered by cloud service providers. Cloud computing is a best-fit for applications where users have heterogeneous, dynamic, and competing quality of service (QoS) requirements. Different applications have different performance levels, workloads and dynamic application scaling requirements, but these characteristics, service models and deployment models create a vague situation when we use the cloud to host applications. The cloud creates complex provisioning, deployment, and configuration requirements. Why simulation is important for the cloud environment? Cloud service providers offer elastic, on-demand, and measured infrastructure, platforms and software services. In the public cloud, tenants have control over the OS, storage and deployed applications. Resources are provisioned in different geographic regions. In the public cloud deployment model, the performance of an application deployed in multiple regions is a matter of concern for organisations. Proof of concepts in the public cloud environment give a better understanding, but cost a lot in terms of capacity building and resource usage even in the pay-per-use model. CloudSim, which is -a toolkit for the modelling and simulation of Cloud computing environments- comes to the rescue. It provides system and behavioural modelling of the Cloud computing components. Simulation of cloud environments and applications to evaluate performance can provide useful insights to explore such dynamic, massively distributed, and scalable environments. The principal advantages of simulation are:

i. Flexibility of defining configurations

ii. Ease of use and customisation

iii. Cost benefits: First designing, developing, testing, and then redesigning, rebuilding, and retesting any application on the cloud can be expensive.

iv. Simulations take the building and rebuilding phase out of the loop by using the model already created in the design phase.

CloudSim is a toolkit for modelling and simulating cloud environments and to assess resource provisioning algorithms. CloudSim is a simulation tool that allows cloud developers to test the performance of their provisioning policies in a repeatable and controllable environment, free of cost. It helps tune the bottlenecks before real-world deployment. 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.

CloudSim is a library for the simulation of cloud scenarios. It provides essential classes for describing data centres, computational resources, virtual machines, applications, users, and policies for the management of various parts of the system such as scheduling and provisioning. Using these components, it is easy to evaluate new strategies governing the use of clouds, while considering policies, scheduling algorithms, load balancing policies, etc. It can also be used to assess the competence of strategies from various perspectives such as cost, application execution time, etc. It also supports the evaluation of Green IT policies. It can be used as a building block for a simulated cloud environment and can add new policies for scheduling, load balancing and new scenarios. It is flexible enough to be used as a library that allows you to add a desired scenario by writing a Java program. By using CloudSim, organisations, R&D centres and industry-based developers can test the performance of a newly developed application in a controlled and easy to set-up environment.

Architecture: The CloudSim layer provides support for modelling and simulation of cloud environments including dedicated management interfaces for memory, storage, bandwidth and VMs. It also provisions host to VMs, application execution management and dynamic system state monitoring. A cloud service provider can implement customised strategies at this layer to study the efficiency of different policies in VM provisioning. The user code layer exposes basic entities such as the number of machines, their specifications, etc, as well as applications, VMs, number of users, application types and scheduling policies.

Regions: It models geographical regions in which cloud service providers allocate resources to their customers. In cloud analysis, there are six regions that correspond to six continents in the world.

Data centres: It models the infrastructure services provided by various cloud service providers. It encapsulates a set of computing hosts or servers that are either heterogeneous or homogeneous in nature, based on their hardware configurations.

Data centre characteristics: It models information regarding data centre resource configurations.

Hosts: It models physical resources (compute or storage).

The user base: It models a group of users considered as a single unit in the simulation, and its main responsibility is to generate traffic for the simulation.

Cloudlet: It specifies the set of user requests. It contains the application ID, name of the user base that is the originator to which the responses have to be routed back, as well as the size of the request execution commands, and input and output files. It models the cloud-based application services. CloudSim categorises the complexity of an application in terms of its computational requirements. Each application service has a pre-assigned instruction length and data transfer overhead that it needs to carry out during its life cycle.

Service broker: The service broker decides which data centre should be selected to provide the services to the requests from the user base.

VMM allocation policy: It models provisioning policies on how to allocate VMs to hosts.

VM scheduler: It models the time or space shared, scheduling a policy to allocate processor cores to VMs.

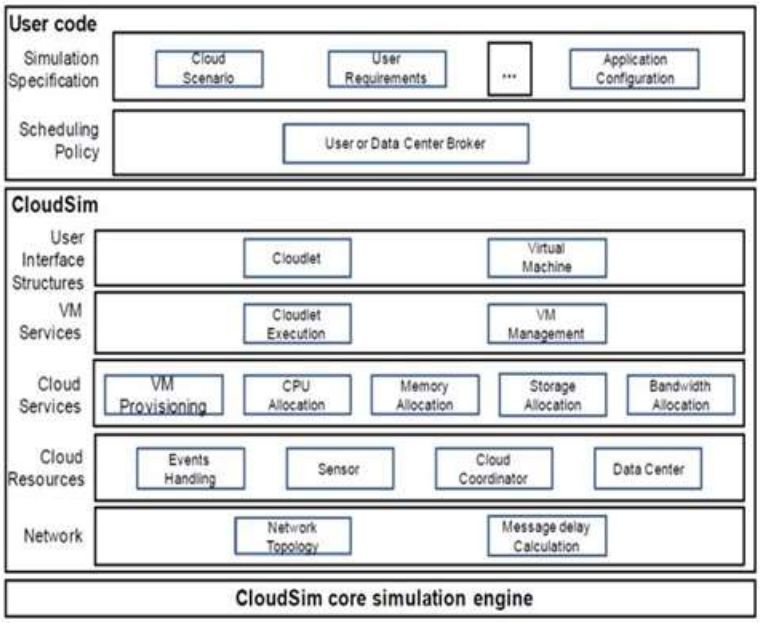

Figure 1: The CloudSim Architecture [11]

3.1.PROPOSED SYSTEM DESIGN AND ARCHITECTURE

The proposed work includes analysis and survey of various currently existing task scheduling algorithms. The various algorithms have been analyzed for various scheduling criteria which are later used to find out the most efficient algorithm. We have successfully implemented the task scheduling algorithms and came up with a new workforce opulence split algorithm. Other algorithms analyzed include First Come First Serve, Shortest Job First and Max Min Scheduling. The simulation of cloud platform has taken place on cloudsim integrated by eclipse, a java based integrated development environment.

Initial releases of CloudSim used SimJava as the discrete event simulation engine that supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation clock. However in the current release, the SimJava layer has been removed in order to allow some advanced operations that are not supported by it. We provide finer discussion on these advanced operations in the next section. The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. The fundamental issues, such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic

system state, are handled by this layer. A Cloud provider, who wants to study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), would need to implement his strategies at this layer. Such implementation can be done by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer related to provisioning of hosts to VMs. A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider's defined QoS levels. This layer also exposes the functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study. The top-most layer in the CloudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. By extending the basic entities given at this layer, a Cloud application developer can perform the following activities:

(i) Generate a mix of workload request distributions, application configurations

(ii) Model Cloud availability scenarios and perform robust tests based on the custom configurations

(iii) Implement custom application provisioning techniques for clouds and their federation. As Cloud computing is still an emerging paradigm for distributed computing, there is a lack of defined standards, tools, and methods that can efficiently tackle the infrastructure and applicationlevel complexities. Hence, in the near future there will be a number of research efforts both in the academia and industry toward defining core algorithms, policies, and application benchmarking based on execution contexts. By extending the basic functionalities already exposed to CloudSim, researchers will be able to perform tests based on specific scenarios and configurations, thereby allowing the development of best practices in all the critical aspects related to Cloud Computing.

3.1.1. MODELLING THE CLOUD

The infrastructure-level services (IaaS) related to the clouds can be simulated by extending the data center entity of CloudSim. The data center entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider. Here, the VM policy stands for the operations control policies related to VM life cycle such as: provisioning of a host to a VM, VM creation, VM destruction, and VM migration. Similarly, one or more application services can be provisioned within a single VM instance, referred to as application provisioning in the context of Cloud computing. In the context of CloudSim, an entity is an instance of a component. A CloudSim component can be a class (abstract or complete) or set of classes that represent one CloudSim model (data center, host). A data center can manage several hosts that in turn manages VMs during their life cycles. Host is a CloudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in

millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs.

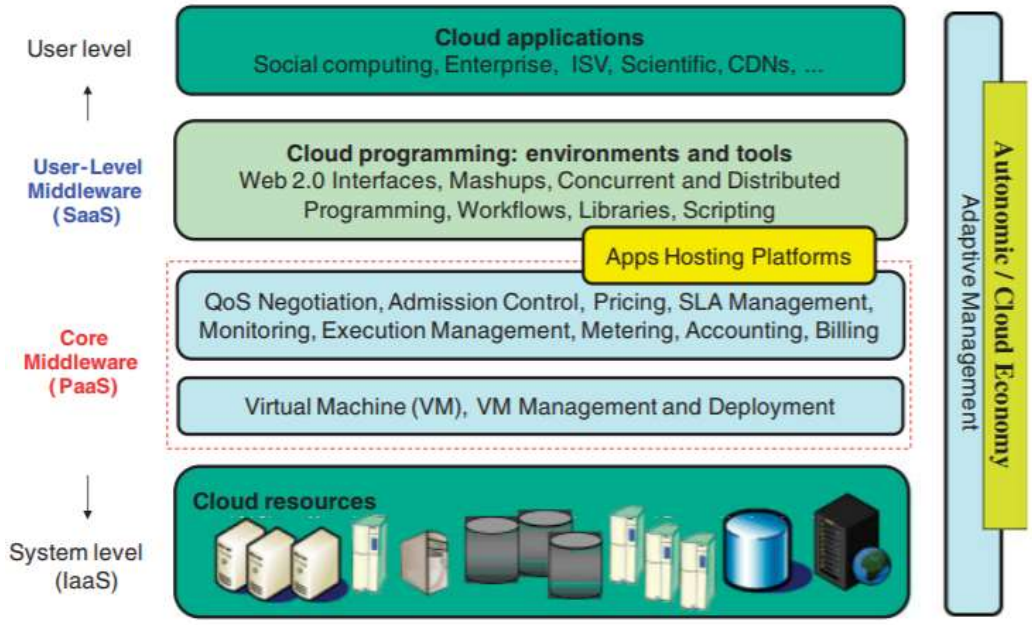

Figure 2: Virtualized Cloud Layered Architecture [12]

The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes. VM allocation (provisioning) is the process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider. CloudSim supports the development of custom application service models that can be deployed within a VM instance and its users are required to extend the core Cloudlet object for implementing their application services. Furthermore, CloudSim does not enforce any limitation on the service models or provisioning techniques that developers want to implement and perform tests with. Once an application service is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service-specific allocation policy. Allocation of application-specific VMs to hosts in a Cloud-based data center is the responsibility of a VM Allocation controller component (called VmAllocationPolicy). This component exposes a number of custom methods for researchers and developers who aid in the implementation of new policies based on optimization goals (user centric, system centric, or both). By default, VmAllocationPolicy implements a straightforward policy that allocates VMs to the Host on a First-Come-First-Serve (FCFS) basis. Hardware requirements, such as the number of processing cores, memory, and storage, form the basis for such provisioning. Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in CloudSim. However, policies used by public Cloud providers (Amazon EC2, Microsoft Azure) are not publicly

available, and thus a pre-implemented version of these algorithms is not provided with CloudSim. For each Host component, the allocation of processing cores to VMs is done based on a host allocation policy. This policy takes into account several hardware characteristics, such as number of CPU cores, CPU share, and amount of memory (physical and secondary), that are allocated to a given VM instance. Hence, CloudSim supports simulation scenarios that assign specific CPU cores to specific VMs (a space-shared policy), dynamically distribute the capacity of a core among VMs (time-shared policy), or assign cores to VMs on demand. Each host component also instantiates a VM scheduler component , which can either implement the space-shared or the time-shared policy for allocating cores to VMs. Cloud system/application developers and researchers can further extend the VM scheduler component for experimenting with custom allocation policies. In the next section, the finer-level details related to the timeshared and space-shared policies are described. Fundamental software and hardware configuration parameters related to VMs are defined in the VM class. Currently, it supports modeling of several VM configurations offered by Cloud providers such as the Amazon EC2.

3.1.2. MODELLING VIRTUAL MACHINE ALLOCATION

One of the key aspects that make a Cloud computing infrastructure different from a Grid computing infrastructure is the massive deployment of virtualization tools and technologies. Hence, as against Grids, Clouds contain an extra layer (the virtualization layer) that acts as an execution, management, and hosting environment for application services. Hence, traditional application provisioning models that assign individual application elements to computing nodes do not accurately represent the computational abstraction, which is commonly associated with Cloud resources. For example, consider a Cloud host that has a single processing core. There is a requirement of concurrently instantiating two VMs on that host. Although in practice VMs are contextually (physical and secondary memory space) isolated, still they need to share the processing cores and system bus. Hence, the amount of hardware resources available to each VM is constrained by the total processing power and system bandwidth available within the host. This critical factor must be considered during the VM provisioning process, to avoid creation of a VM that demands more processing power than is available within the host. In order to allow simulation of different provisioning policies under varying levels of performance isolation, CloudSim supports VM provisioning at two levels: first, at the host level and second, at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core will be assigned to each VM. At the VM level, the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine. For the purpose of this paper, we consider a task unit as a finer abstraction of an application service being hosted in the VM. At each level, CloudSim implements the time-shared and space-shared provisioning policies. In the architecture figure, a host with two CPU cores receives request for hosting two VMs, such that each one requires two cores and plans to host four tasks' units. More

specifically, tasks t1, t2, t3, and t4 to be hosted in VM1, whereas t5, t6, t7, and t8 to be hosted in VM2.

3.1.3. MODELLING NETWORK BEHAVIOUR

Modeling comprehensive network topologies to connect simulated Cloud computing entities (hosts, storage, end-users) is an important consideration because latency messages directly affect the overall service satisfaction experience. An end-user or a SaaS provider consumer who is not satisfied with the delivered QoS is likely to switch his/her Cloud provider; hence, it is a very important requirement that Cloud system simulation frameworks provide facilities for modeling realistic networking topologies and models. Inter-networking of Cloud entities (data centers, hosts, SaaS providers, and end-users) in CloudSim is based on a conceptual networking abstraction. In this model, there are no actual entities available for simulating network entities, such as routers or switches. Instead, network latency that a message can experience on its path from one CloudSim entity (host) to another (Cloud Broker) is simulated based on the information stored in the latency matrix (see Table I). For example, Table I shows a latency matrix involving five CloudSim entities. At any instance of time, the CloudSim environment maintains an m×n size matrix for all CloudSim entities currently active in the simulation context. An entry eij in the matrix represents the delay that a message will undergo when it is being transferred from entity i to entity j over the network. Recall, that CloudSim is an event-based simulation, where different system models/entities communicate via sending events. The event management engine of CloudSim utilizes the inter-entity network latency information for inducing delays in transmitting message to entities. This delay is expressed in simulation time units such as milliseconds. It means that an event from entity i to j will only be forwarded by the event management engine when the total simulation time reaches the t +d value, where t is the simulation time when the message was originally sent, and d is the network latency between entities i and j. The transition diagram representing such an interaction is depicted in cloudsim. This method of simulating network latencies gives us a realistic yet simple way of modelling practical networking architecture for a simulation environment. Further, this approach is much easier and cleaner to implement, manage, and simulate than modelling complex networking components such as routers, switches etc.

3.1.4. Modelling Power Consumption

Cloud computing environments are built upon an inter-connected network of a large number (hundreds-of-thousands) of computing and storage hosts for delivering on-demand services (IaaS, PaaS, and SaaS). Such infrastructures in conjunction with a cooling system may consume enormous amount of electrical power resulting in high operational costs. Lack of energy-conscious provisioning techniques may lead to overheating of Cloud resources (compute and storage servers) in case of high loads. This in turn may result in reduced system reliability and lifespan of devices. Another related issue is the carbon dioxide (CO2) emission that is detrimental to the physical

environment due to its contribution in the greenhouse effect. All these problems require the development of efficient energy-conscious provisioning policies at resource, VM, and application level. To this end, the CloudSim framework provides basic models and entities to validate and evaluate energy-conscious provisioning of techniques/algorithms. We have made a number of extensions to CloudSim for facilitating the above, such as extending the PE object to include an additional Power Model object for managing power consumption on a per Cloud host basis. To support modeling and simulation of different power consumption models and power management techniques such as Dynamic Voltage and Frequency Scaling (DVFS), we provide an abstract implementation called PowerModel. This abstract class should be extended for simulating custom power consumption model of a PE. CloudSim users need to override the method getPower() of this class, whose input parameter is the current utilization metric for Cloud host and return parameter is the current power consumption value. This capability enables the creation of energy-conscious provisioning policies that require real-time knowledge of power consumption by Cloud system components. Furthermore, it enables the accounting of the total energy consumed by the system during the simulation period.

4. IMPLEMENTATION AND ANALYSIS

4.1.BRIEF INTRODUCTION

The proposed work includes analysis and survey of various currently existing task scheduling algorithms. The various algorithms have been analysed for various scheduling criteria which are later used to find out the most efficient algorithm. This involves successful implementation of the task scheduling algorithms in order to come up with a new workforce dynamic opulence split algorithm. Other algorithms analysed include First Come First Serve, Shortest Job First and Max Min Scheduling, Round Robin etc. The simulation of cloud platform has taken place on cloudsim integrated by eclipse, a java based integrated development environment.

4.2.REQUIREMNT ANALYSIS

4.2.1. SOFTWARE REQUIREMENTS

- Java (JDK) 3.7 or above
- Windows 7 or Above
- Apache Common Maths 3.1
- Libraries- CloudSim Toolkit for Java, PSwarmlib

4.2.2. HARDWARE REQUIREMENTS

- 4GB RAM (8GB Recommended)
- 20 MB Memory

4.3.DETAILED DESIGNS

4.3.1. CLASS DIAGRAM

CloudSim provides a generalised and extensible simulation framework that enables seamless modelling and simulation of app performance. By using CloudSim, developers can focus on specific systems design issues that they want to investigate, without getting concerned about details related to cloud-based infrastructures and services.

BwProvisioner: This is an abstract class that models the policy for provisioning of bandwidth to VMs. The main role of this component is to undertake the allocation of network bandwidths to a set of competing VMs that are deployed across the data center. Cloud system developers and researchers can extend this class with their own policies

(priority, QoS) to reflect the needs of their applications. The BwProvisioningSimple allows a VM to reserve as much bandwidth as required; however, this is constrained by the total available bandwidth of the host. CloudCoordinator: This abstract class extends a Cloud-based data center to the federation. It is responsible for periodically monitoring the internal state of data center resources and based on that it undertakes dynamic load-shredding decisions. Concrete implementation of this component includes the specific sensors and the policy that should be followed during load-shredding. Monitoring of data center resources is performed by the updateDatacenter() method by sending queries Sensors. Service/Resource Discovery is realized in the setDatacenter()abstract method that can be extended for implementing custom protocols and mechanisms (multicast, broadcast, peer-to-peer). Further, this component can also be extended for simulating Cloud-based services such as the Amazon EC2 Load-Balancer. Developers aiming to deploy their application services across multiple clouds can extend this class for implementing their custom inter-cloud provisioning policies.

Cloudlet: This class models the Cloud-based application services (such as content delivery, social networking, and business workflow). CloudSim orchestrates the complexity of an application in terms of its computational requirements. Every application service has a pre-assigned instruction length and data transfer (both pre and post fetches) overhead that it needs to undertake during its life cycle. This class can also be extended to support modeling of other performance and composition metrics for applications such as transactions in database-oriented applications.

CloudletScheduler: This abstract class is extended by the implementation of different policies that determine the share of processing power among Cloudlets in a VM. As described previously, two types of provisioning policies are offered: space-shared (CloudetSchedulerSpaceShared) and time-shared (CloudletSchedulerTimeShared).

Datacenter: This class models the core infrastructure-level services (hardware) that are offered by Cloud providers (Amazon, Azure, App Engine). It encapsulates a set of compute hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations (memory, cores, capacity, and storage). Furthermore, every Datacenter component instantiates a generalized application provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices to hosts and VMs.

DatacenterBroker or Cloud Broker: This class models a broker, which is responsible for mediating negotiations between SaaS and Cloud providers; and such negotiations are driven by

QoS requirements. The broker acts on behalf of SaaS providers. It discovers suitable Cloud service providers by querying the CIS and undertakes online negotiations for allocation of resources/services that can meet the application's QoS needs. Researchers and system developers must extend this class for evaluating and testing custom

brokering policies. The difference between the broker and the CloudCoordinator is that the former represents the customer (i.e. decisions of

these components are made in order to increase user-related performance metrics), whereas the latter acts on behalf of the data center, i.e. it tries to maximize the overall performance of the data center, without considering the needs of specific customers.

DatacenterCharacteristics: This class contains configuration information of data center resources.

Host: This class models a physical resource such as a compute or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (to represent a multi-core machine), an allocation of policy for sharing the processing power among VMs, and policies for provisioning memory and bandwidth to the VMs.

NetworkTopology: This class contains the information for inducing network behavior (latencies) in the simulation. It stores the topology information, which is generated using the BRITE topology generator.

RamProvisioner: This is an abstract class that represents the provisioning policy for allocating primary memory (RAM) to the VMs. The execution and deployment of VM on a host is feasible only if the RamProvisioner component approves that the host has the required amount of free memory. The RamProvisionerSimple does not enforce any limitation on the amount of memory that a VM may request. However, if the request is beyond the available memory capacity, then it is simply rejected.

SanStorage: This class models a storage area network that is commonly ambient in Cloud-based data centers for storing large chunks of data (such as Amazon S3, Azure blob storage). SanStorage implements a simple interface that can be used to simulate storage and retrieval of any amount of data, subject to the availability of network bandwidth. Accessing files in a SAN at run-time incurs additional delays for task unit execution; this is due to the additional latencies that are incurred in transferring the data files through the data center internal network.

Sensor: This interface must be implemented to instantiate a sensor component that can be used by a CloudCoordinator for monitoring specific performance parameters (energy-consumption, resource utilization). Recall that, CloudCoordinator utilizes the dynamic performance information for undertaking load-balancing decisions. The methods defined by this interface are: (i) set the minimum and maximum thresholds for performance parameter and (ii) periodically update the measurement. This class can be used to model the real-world services offered by leading Cloud providers such as Amazon's CloudWatch and Microsoft Azure's FabricController. One data center may instantiate one or more Sensors, each one responsible for monitoring a specific data center performance parameter.

Vm: This class models a VM, which is managed and hosted by a Cloud host component.

VmmAllocationPolicy: This abstract class represents a provisioning policy that a VM Monitor utilizes for allocating VMs to hosts. The chief functionality of the VmmAllocationPolicy is to select the available host in a data center that meets the memory, storage, and availability requirement for a VM deployment.

VmScheduler: This is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processor cores to VMs. The functionalities of this class can easily be overridden to accommodate application-specific processor sharing policies.

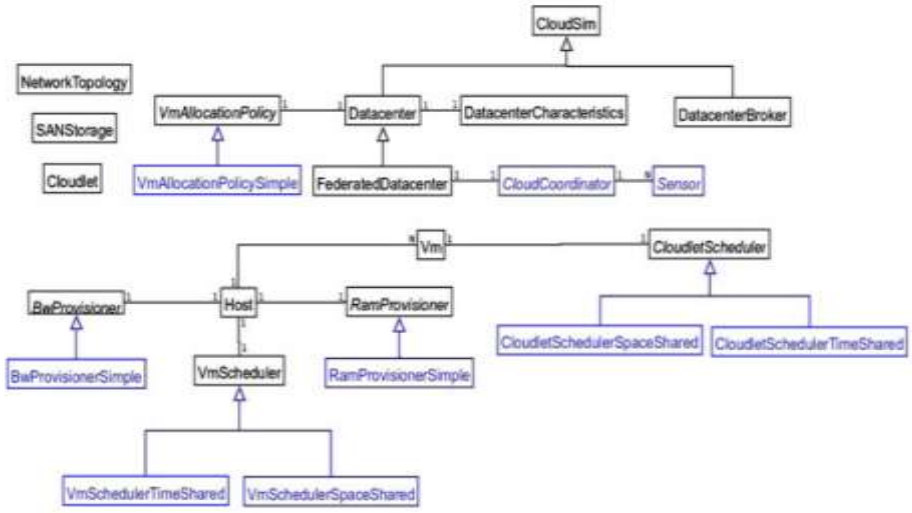

Figure 3: The Task Scheduling Class Diagram [13]

4.3.2. FLOWCHART FOR EXECUTION

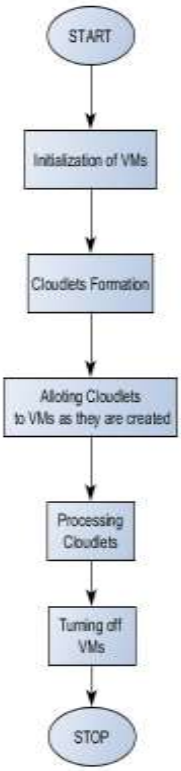

Figure 4: FCFS

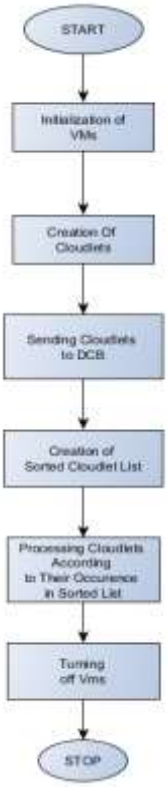

Figure 5: SJF

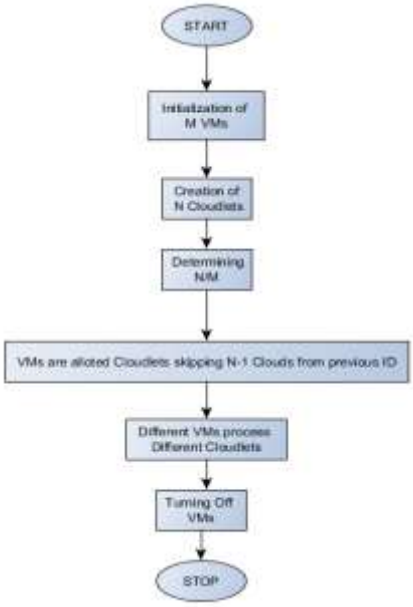

Figure 6: Opulence Split Algorithm

4.3.3. ALGORITHM

The opulence split algorithm exploits dynamism and fault tolerance in order to generate an algorithm which leads to formation of cloudlets and allocation of resources in real time. After getting the execution time and the utilisation percentage, the scheduler can either complete the task or allot it to a different virtual machine, therefore taking in the next task which is waiting in the queue. The main aim of the algorithm is to minimize waiting time and maximize throughput.

Opulence Split Algorithm
Step 1: Start
Step 2: Initialization of virtual machines
Step 3: Cloudlet formation
Step 4: Determine N/M
Step 5: Every nth cloudlet will be allocated to the virtual machine skipping N-1 Cloud from previous ID
Step 6: Get Execution time and Utilisation Rate
Step 7: Process cloudlets in a circular fashion with n-1 different Virtual Machine processing different cloudlets.
Step 8: Close down virtual machines

Table 1: The Opulence Split Algorithm

5. RESULTS AND DISCUSSION

The execution of the algorithms was conducted in cloud sim. The different algorithms like FCFS, SJF, Opulence Split and Particle Swarm Optimization were implemented and their executions were observed. The different algorithms were implemented in cloudsim with a task size of 10. The task sizes are directly proportional to the total completion time of the algorithms. The particle swarm optimization algorithm was implemented with a particle size=30 in order to track the performance of the algorithm. Figure 5,6,7 and 8 show the completion times of different algorithms.

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
00	SUCCESS	03	06	3445.49	00.2	3445.69
01	SUCCESS	02	03	3483.54	00.2	3483.74
02	SUCCESS	02	05	2405.12	00.2	2405.32
03	SUCCESS	02	05	2909.7	2405.32	5315.02
04	SUCCESS	02	05	2410.35	5315.02	7725.37
05	SUCCESS	02	03	4230.05	3483.74	7713.8
06	SUCCESS	02	03	3894.74	7713.8	11608.54
07	SUCCESS	02	03	1145.88	11608.54	12754.42
08	SUCCESS	02	04	2193.71	00.2	2193.91
09	SUCCESS	02	05	2529.81	7725.37	10255.18
10	SUCCESS	02	03	1128.66	12754.42	13883.08

Figure 7: FCFS

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
02	SUCCESS	05	05	2405.12	00.1	2405.22
00	SUCCESS	02	02	2598.4	00.1	2598.5
03	SUCCESS	04	04	2643.06	00.1	2643.16
01	SUCCESS	06	06	2894	00.1	2894.1
06	SUCCESS	03	03	3894.74	00.1	3894.84
04	SUCCESS	06	06	1088	2894.1	3982.1
09	SUCCESS	02	02	1594.43	2598.5	4192.92
10	SUCCESS	05	05	1967.3	2405.22	4372.52
05	SUCCESS	04	04	3289.26	2643.16	5932.41
12	SUCCESS	03	03	2324.88	3894.84	6219.73

Figure 8: SJF

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
08	SUCCESS	02	02	1379.77	00.1	1379.87
02	SUCCESS	03	03	1863.62	00.1	1863.72
13	SUCCESS	04	04	1939.54	00.1	1939.64
00	SUCCESS	06	06	3445.49	00.1	3445.59
03	SUCCESS	06	06	365.38	3445.59	3810.97
01	SUCCESS	05	05	4066.28	00.1	4066.38
24	SUCCESS	04	04	2411.76	1939.64	4351.4
11	SUCCESS	02	02	3340.98	1379.87	4720.86
04	SUCCESS	06	06	1088	3810.97	4898.96
05	SUCCESS	03	03	4230.05	1863.72	6093.78

Figure 9: Opulence Split

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
03	SUCCESS	03	03	365.38	00.1	365.48
20	SUCCESS	05	05	906.79	00.1	906.89
07	SUCCESS	04	04	1145.88	00.1	1145.98
22	SUCCESS	06	06	1230.53	00.1	1230.63
04	SUCCESS	03	03	1088	365.48	1453.48
14	SUCCESS	04	04	907.35	1145.98	2053.32
27	SUCCESS	06	06	1311.59	1230.63	2542.22
00	SUCCESS	02	02	2598.4	00.1	2598.5
15	SUCCESS	04	04	580.75	2053.32	2634.08
21	SUCCESS	05	05	2045.46	906.89	2952.36

Figure 10: Particle Swarm Optimization

Figure 10 compares the make span of the different algorithms. Particle Swam Optimization is highly efficient in in dealing with a number of tasks. Makespan refers to the total time elapsed between the arrival and completion of tasks. On the other hand, a parallel study on the performance of particle swarm optimization involved variation of the particle size to observe the effect on the makespan and overall performance of the algorithm. Figure 11 explains the variance of makespan with the particle size.

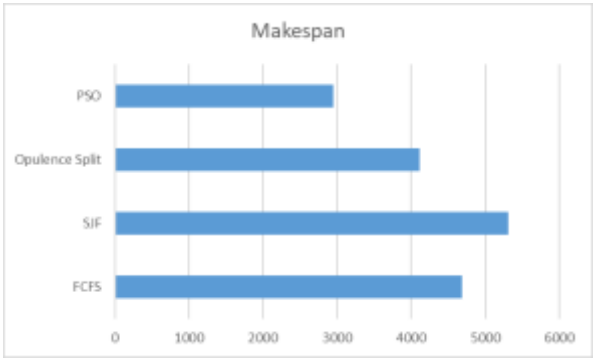

Figure 11: Makespan Comparison

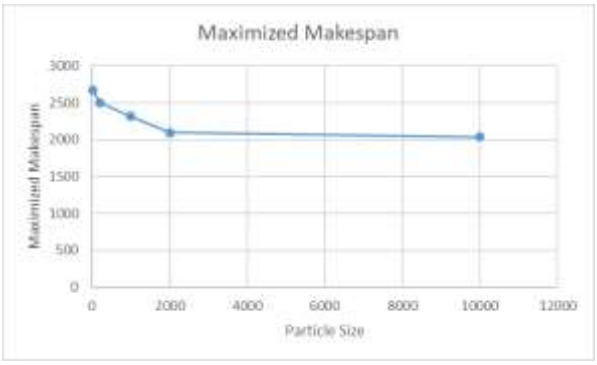

Figure 12: Makespan vs Particle Size

Figure 12 makes a comparison with respect to the task completion time in the different algorithms. Table 2 is a concise explanation of the researched algorithms based on different metrics of comparison.

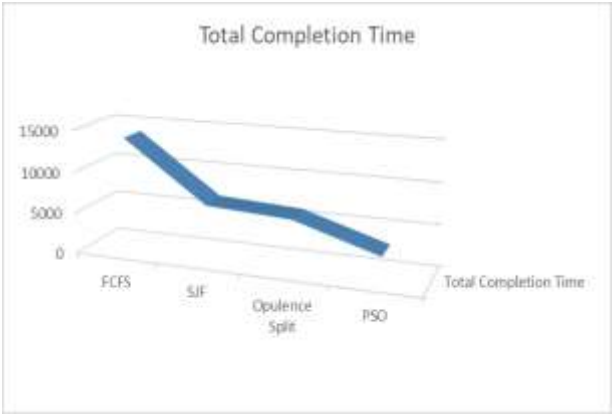

Figure 13: Time Comparison of Different Algorithms

Scheduling Technique	Criteria of Scheduling	Advantage	Disadvantage
FCFS	Arrival on basis of cloudlet id	Easy to implement	No considerations of optimisations
MAX-MIN	Completion Time	Better makespan	Poor Load Balancing
SJF	Completion time, arrival	Lesser Starvation Minimised TAT	Starvation of larger processes.
OPULENCE SPLIT	Division of Cloudlets is made possible.	Load Balancing and concurrency control implementation	Sometimes, number of cloudlets are uneven, hence energy is wasted.
PSO	Particle Size	Highly Efficient and scalable	Difficult to Integrate with Real Time Workflows

Table 2: Empirical Comparison of Different Algorithms

6. CONCLUSION

In this project we successfully analysed the different cloudlet scheduling algorithms with respect to their properties. We also proposed an improved algorithm named Opulence Split implemented to balance the load between different machines. Simulation of all the algorithms was done on CloudSim. It was observed that Particle Swarm Optimization is a highly efficient and scalable algorithm. With respect to the increasing number of tasks, it can be used to obtain better throughput and reduced makespan. However it fails to be effective as it requires memory latency and is harder to integrate with real time scenarios. FCFS, SJF, Max-Min, Min-Min etc are traditional scheduling algorithms pretty useful in their respective scenarios. However they have their own limitations and drawbacks. The Opulence Split Algorithm invokes concurrency control and increased load balancing and fault tolerance to maximise resource utilization by using dynamic cloudlet creation. However, the algorithm did take time to execute, it provided better performance compared to FCFS and SJF. It is easier to implement Opulence Split Algorithm when dealing with real time scenarios and there is a resource constraint. The techniques currently present are effective although they are not working on finiteness and may fail to implement principles of concurrency. Future work can be used to make the algorithms more energy efficient and increase dynamism.

7. REFERENCES

[1] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[3] Dillon, Tharam, Chen Wu, and Elizabeth Chang. "Cloud computing: issues and challenges." Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. Ieee, 2010.

[4] Fazel, M. D., S. Jamali, and M. Bekravi. "Survey on job scheduling algorithms in cloud computing." International Journal of Emerging Trends and Technology in Computer Science (2014): 151-4.

[5] Mittal, Shubham, and Avita Katal. "An Optimized Task Scheduling Algorithm in Cloud Computing."Advanced Computing (IACC), 2016 IEEE 6th International Conference on IEEE, 2016.

[6] Nishant, Kumar, et al. "Load balancing of nodes in cloud using ant colony optimization." Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on. IEEE, 2012.

[7] Xu, Baomin, et al. "Job scheduling algorithm based on Berger model in cloud environment." Advances in Engineering Software 42.7 (2011): 419-425.

[8] Mathew, Teena, K. Chandra Sekaran, and John Jose. "Study and analysis of various task scheduling algorithms in the cloud computing environment." Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on . IEEE, 2014.

[9] Wadhonkar, A., & Theng, D. (2016, February). A survey on different scheduling algorithms in cloud computing. In Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2016 2nd International Conference on (pp. 665-669). IEEE

[10] Guo, L., Zhao, S., Shen, S., & Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. JNW,7(3),547-553.

[11] Cloud Environment Simulator, C. (2017). CloudSim Architecture. Available at: http://www.thedailyprogrammer.com/2016/09/cloudsimsimulation-software-for-cloud.html [Accessed 1 JAN. 2019].

[12] Calheiros, Rodrigo N., et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience 41.1 (2011): 23-50.

[13] Cloud Environment Simulator, C. (2017). CloudSim Class Diagram. Available at: http://www.thecloudsimframework.com/2016/09/cloudsimsimulation-software-for-cloud.html [Accessed 29 FEB. 2019].