PROJECT REPORT
ON

Resource Scheduling in Cloud Environment using CloudSim

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

Year (2019-20)

SUBMITTED BY

Yamini Shankar, 10555

CENTRAL UNIVERSITY OF HARYANA

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

UNDER THE SUPERVISION OF

External Supervisor

Dr. Dinesh Kumar

Assistant Professor

Department of Computer Science

Motilal Nehru National Institute of Technology Allahabad

Internal Supervisor

Dr. Suraj Arya

Asst. Professor

Dept of CS & IT

Central University of Haryana

Student's Declaration

I, **Yamini Shankar**, do here by declare that the project entitled "**Resource Scheduling in Cloud Environment using CloudSim**" submitted in the partial fulfillment of requirement for award of the degree of Master of Computer Application to Central University of Haryana, is my authentic work carried out during the 6th semester of my course under the supervision of Dr.Suraj Arya, Assistant Professor, Department of Computer Science &Information Technology, Central University of Haryana. The project has not formed the basis for award of any other degree, associate ship, fellowship or any other similar title.

Yamini Shankar with Signature and date

Roll no-10555

Session -2019-20

Reg. NO

Central University of Haryana

Dept. CS & IT

Certificate

This is to certify that this report entitled **Resource Scheduling in Cloud Environment using CloudSim** is a bonafide record of the Project presented by **Yamini Shankar**, Roll No. **10555**, Session **2019-20** under my Supervision in the partial fulfillment of the requirement for award the degree of Master of Computer Application.

Dr. Suraj Arya

Date: …/…/…

Project Advisor & Assistant Professor

Dept. CS & IT

Central University of Haryana

Central University of Haryana

Dept. CS & IT

Certificate

This is to certify that this report entitled **Resource Scheduling in Cloud Environment using CloudSim** is a bonafied record of the project presented by **Yamini Shankar**, Roll No. **10555**, Session **2019-20** a student of Master of Computer Application at Department of Computer Science And Information Technology, Central University of Haryana, in the partial fulfillment of the requirement for award the degree of Master of Computer Application.

Rajesh Kr. Gupta

Date: …/…/…

Post:
Dept. CS & IT

Central University of Haryana

ACKNOWLEDGEMENT

I would like to express my deep gratitude to **Dr. Dinesh Kumar**, Asst. Professor, MNNIT Allahabad and **Dr. Suraj Arya**, Asst. Professor, Central University of Haryana, my supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. My grateful thanks are also extended to **Miss Saumya** and **Miss Brijneeta** for their guidance, encouragement and support throughout.

I would also thank **Mr. Shijin Rajan** and **Mr. Abhishek** for helping be with the documentation and their valuable suggestions throughout.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

PREFACE

Cloud computing is the ability to access a pool of computing resources owned and maintained by a third party via the Internet. It is not a new technology but a way of delivering computing resources based on long existing technologies such as server virtualization. In computers, Scheduling is a process of arranging the submitted jobs/task into a very specific sequence of execution. It is an essential characteristic of any software operating environment, which is handled by a very special program known as a scheduler.

Quantifying the performance of provisioning (scheduling and allocation) policies in a real Cloud computing environment (Amazon EC2, Microsoft Azure Google App Engine) for different application models under transient conditions is extremely challenging. The use of real infrastructures, such as Amazon EC2 and Microsoft Azure, for benchmarking the application performance (throughput, cost benefits) under variable conditions (availability, workload patterns) is often constrained by the rigidity of the infrastructure. Hence, this makes the reproduction of results that can be relied upon, extremely difficult.

A more viable alternative is the use of simulation tools. These tools open up the possibility of evaluating the hypothesis (application benchmarking study) in a controlled environment where one can easily reproduce results.

I have done this project under the guidance of Dr. Dinesh Kumar, MNNIT, Allahabad. The project was undertaken to perform various case studies on the cloud environment by making changes in scheduling the cloudlet and the virtual machines.

Doing this project helped me gain clarity and understanding of the cloud as well as the simulation tools which are used to strategize the decisions for the cloud. This project was done in the partial fulfillment of the MCA degree, as a major project.

The simulation framework used i.e., CloudSim, on the IDE Eclipse is clearly explained along with the major classes and namespaces. The case studies are explained in the Chapter 6 of the file. The reader will get the relevant details and explanation if they wish to publish any such work.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Clouds aim the next-generation data centers as they enable platform for dynamic and flexible application provisioning. Some of the traditional and emerging Cloud-based application services include social networking, web hosting, content delivery, and real-time instrumented data processing. Each of these application types has different composition, configuration, and deployment requirements.

1.1 Cloud Computing and CloudSim

Cloud computing is the ability to access a pool of computing resources owned and maintained by a third party via the Internet. It is not a new technology but a way of delivering computing resources based on long existing technologies such as server virtualization. The "cloud" is composed of hardware, storage, networks, interfaces, and services that provide the means through which users can access the infrastructures, computing power, applications, and services on demand which are independent of locations. Cloud computing usually involves the transfer, storage, and processing of information on the 'providers' infrastructure, which is not included in the 'customers' control policy.

In computers, Scheduling is a process of arranging the submitted jobs/task into a very specific sequence of execution. It is an essential characteristic of any software operating environment, which is handled by a very special program known as a scheduler. Scheduler's main objective is to keep the underlined hardware resources(primarily processor) to be used effectively as well as efficient.

As cloud computing is the virtualized operating environment, and the virtual machines are the primary computing component which is responsible for the execution of the workloads (tasks). The virtual machine(s) are powered by a physical server host machine (i.e.) hardware. Depending on the requirement of the Virtual Machine (VM) there could be 'one to one' or 'many to one' mapping between the VM and host machine. That means in cloud computing the scheduling is done at both the mapping levels that are:

- Virtual Machine to Host Machines
- Tasks to Virtual Machines

Quantifying the performance of provisioning (scheduling and allocation) policies in a real Cloud computing environment (Amazon EC2, Microsoft Azure Google App Engine) for different application models under transient conditions is extremely challenging because:

a) Clouds exhibit varying demands, supply patterns, system sizes, and resources (hardware, software, network);
b) Users have heterogeneous, dynamic, and competing QoS requirements; and
c) Applications have varying performance, workload, and dynamic application scaling requirements.

The use of real infrastructures, such as Amazon EC2 and Microsoft Azure, for benchmarking the application performance (throughput, cost benefits) under variable conditions (availability, workload patterns) is often constrained by the rigidity of the infrastructure. Hence, this makes the reproduction of results that can be relied upon, extremely difficult. Further, it is complex and time consuming to re-configure benchmarking parameters across a massive-scale Cloud computing infrastructure over multiple test runs. Such limitations are caused by the conditions prevailing in the Cloud-based environments that are not in the control of developers of application services. Thus, it is not possible to perform benchmarking experiments in repeatable, dependable, and scalable environments using real-world Cloud environments.

A more viable alternative is the use of simulation tools. These tools open up the possibility of evaluating the hypothesis (application benchmarking study) in a controlled environment where one can easily reproduce results. Simulation-based approaches offer significant benefits to IT companies (or anyone who wants to offer his application services through clouds) by allowing them to:

a) test their services in repeatable and controllable environment;

b) tune the system bottlenecks before deploying on real clouds; and
c) experiment with different workload mix and resource performance scenarios on simulated infrastructures for developing and testing adaptive application provisioning techniques

CloudSim is a new, generalized, and extensible simulation framework that allows seamless modelling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. By using CloudSim, researchers and industry-based developers can test the performance of a newly developed application service in a controlled and easy to set-up environment. Based on the evaluation results reported by CloudSim, they can further fine-tune the service performance. The main advantages of using CloudSim for initial performance testing include:

a) *time effectiveness*: it requires very less effort and time to implement Cloud-based application provisioning test environment and
b) *flexibility and applicability*: developers can model and test the performance of their application services in heterogeneous Cloud environments (Amazon EC2, Microsoft Azure) with little programming and deployment effort.

CloudSim offers the following novel features:

a) support for modeling and simulation of large scale Cloud computing environments, including data centers, on a single physical computing node;
b) a self-contained platform for modeling Clouds, service brokers, provisioning, and allocation policies;
c) availability of a virtualization engine that aids in the creation and management of multiple, independent, and co-hosted virtualized services on a data center node and flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

These compelling features of CloudSim would speed up the development of new application provisioning algorithms for Cloud computing.

1.2 Problem Description

My project work in the internship is driven by the motivation for simulation of cloud computing environment and evaluation of resource provisioning algorithms for cloudlets both in the SPACE-SHARED and TIME-SHARED environments.

The major contribution in the project work is as follows:

a) To install, deploy and simulate the CloudSim Environment
b) Implementation of Shortest Job First (SJF), longest job first (LJF), and the first come first serve (FCFS) algorithms for the scheduling of cloudlets on several virtual machines.
c) Performing several case study by varying the number of tasks and virtual machines
d) Verify the scalability of the scheduling algorithms via simulation

1.3 About the Organization

Motilal Nehru National Institute of Technology Allahabad (**MNNIT** or **NIT Allahabad**), formerly Motilal Nehru Regional Engineering College (MNREC), is a public technical university located in Prayagraj (Allahabad), Uttar Pradesh, India. MNNIT is an Institute with total commitment to the quality and excellence in academic pursuits. It was established as one of the seventeen Regional Engineering Colleges of the India in the year 1961 and was associated college of the University of Allahabad, which is the third oldest university in India. The Institute has been selected as a Lead Institution under World Bank funded Govt. of India Project on Technical Education Quality Improvement Programme (TEQIP)(2002-2007). Two state level institutions are networked with MNNIT under the project.

- **MOTTO-**Success is born out of action**.**
- **ESTABLISHED**-1961
- **LOCATION-** Prayagraj, Uttar Pradesh.

2. LITERATURE OVERVIEW

The term cloud was first popularized by Amazon services in 2006 with the release of its ELASTIC COMPUTE CLOUD (EC2). Amazon EC2 is a part of Amazon's cloud computing platform, Amazon Web Services, which allows users to rent the virtual computers on which the clients run their own computer applications. Early references to cloud computing were in 1996 with the first known mention in a Compaq internal document. In early times as 1993, the term cloud was used to refer to platform for distributes computing.

In present day scenario, cloud computing is the ability to access a pool of computing resources owned and maintained by a third party via the Internet. It is not a new technology but a way of delivering computing resources based on long existing technologies such as server virtualization. The "cloud" is composed of hardware, storage, networks, interfaces, and services that provide the means through which users can access the infrastructures, computing power, applications, and services on demand which are independent of locations. Cloud computing usually involves the transfer, storage, and processing of information on the 'providers' infrastructure, which is not included in the 'customers' control policy.

Fig. 1 What is Cloud [4]

2.1 Cloud Computing and Its Features

The cloud allows users to take benefits of all the services and technologies without having to gain any prior deep knowledge or experience in each of them. The cloud aims

to cuts the cost and helps users to focus properly and solely on their business and aim without worrying about the services and the maintenance. The spine of the cloud computing is the virtualization technology. The virtualization separates a physical computing device into one or more devices virtually each of which can be easily used and manages to perform different and independent tasks. The virtualization on the OS level essentially creates a scalable system of multiple independent devices on which resources can be allocates and used more efficiently. Virtualization provides the agility required to speed up IT operations and reduces the cost by proper infrastructure utilization.

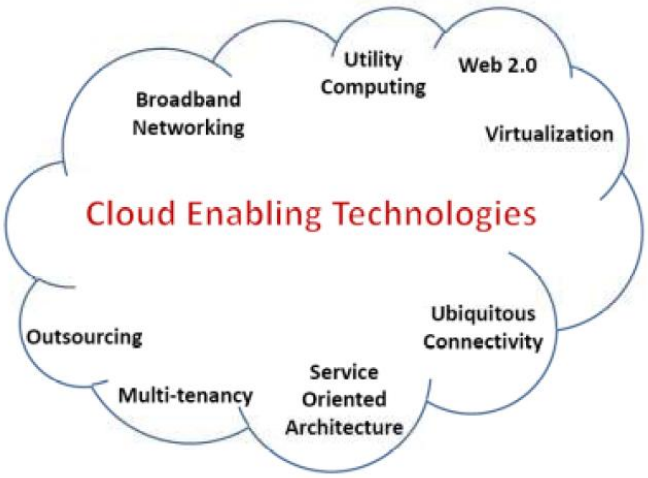

Fig. 2 Cloud enabling technologies[7]

The cloud computing exhibits the following features-

a) Agility for the organizations may be improved due to increased flexibility with re-provisioning, adding or expanding technological infrastructure resources. Agility refers to the ability to be mobile or to be able to move quickly and easily.
b) Cost reduction- cloud computing helps reduce the cost on a very large scale as all the facilities and other services are provided by the third party and the basic cost to setup the whole infrastructure and maintaining them reduces.
c) Device and location independence- As the infrastructure is off-site and provided by the third party, all the services and perks can be enjoyed without actually necessary to be on a

particular site. This helps the users and the clients to enjoy mobility along with their device and still be able to work and manage.

d) Maintenance becomes easier and very less to be worried about because all the services including the infrastructure and server is managed and maintained by the service providers and their IT professionals.

e) Multi-latency enables the sharing of resources and costs among many multiple users at the same time increasing the load balancing and efficiency.

f) Performance of the whole services increases as the cloud and its services is managed by the highly qualified professionals and IT experts from the service provider.

g) Productivity is increased due to multiple clients working on the same service and data at the same time simultaneously.

h) Availability improves with the use of multiple redundant sites and servers.

i) Security also improves due to centralization of data and better management.

j) On-demand self service- A consumer can provision computing abilities such as server time and network storage as needed automatically.

k) Resource pooling- The providers' computing resources are pooled to serve multiple consumers at the same time with allocation of different physical and virtual resources dynamically.

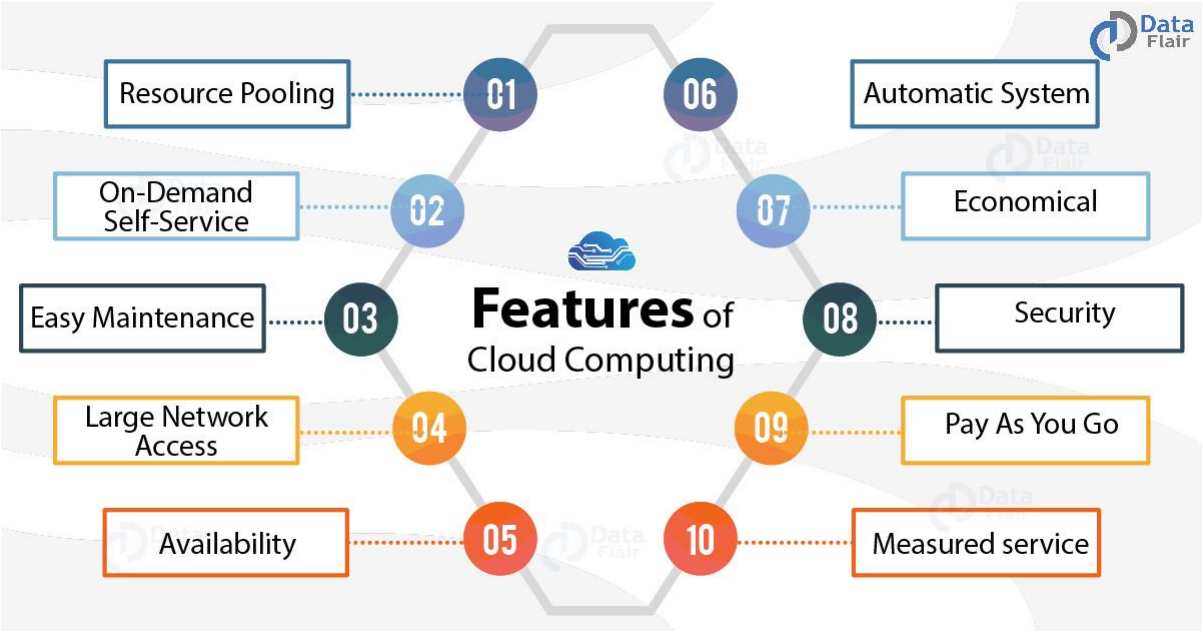

Fig. 3 Features of cloud computing[11]

2.2 Types of Clouds

(i) Private cloud

Private cloud is a cloud infrastructure employed solely for an organization whether managed internally or by a third party and hosted either internally or externally. Private cloud refers to cloud computing resources that are used exclusively within a business or an organization. A private cloud can be physically located on the company's on-site datacenter. Some companies also pay third party cloud service providers to host their private cloud. A private cloud is the one in which servers and infrastructures are maintained on a private network.

Example: Cloud services within one organization or a business.

(ii) Public cloud

A cloud is called a public cloud when the services of the cloud are rendered over a network that are open for the public to use. The public cloud services can be free to use. Generally, there is not much difference between the internal architecture of the private and public clouds for a client but the audience view is public and the services are open to everybody in the public clouds.

Major public cloud services providers like Amazon web services, Oracle, Microsoft own and operate their infrastructure at their data center and generally deliver then via the internet.

Example: Google mail service.

(iii)Hybrid cloud

Hybrid cloud is the composition of the public cloud and the private cloud that remains distinct entities but is bound together, offering the benefits of multiple deployment

models. Hybrid cloud combines public and private clouds, bound together by technology that allows data and applications to be shared between them. Hybrid cloud provides greater flexibility, more deployment options and helps optimize your existing infrastructure, security and compliance.

Example: Various use cases for hybrid cloud composition exists. For example, a company can have their sensitive data stored on a private cloud but the shared data provided on a public cloud.

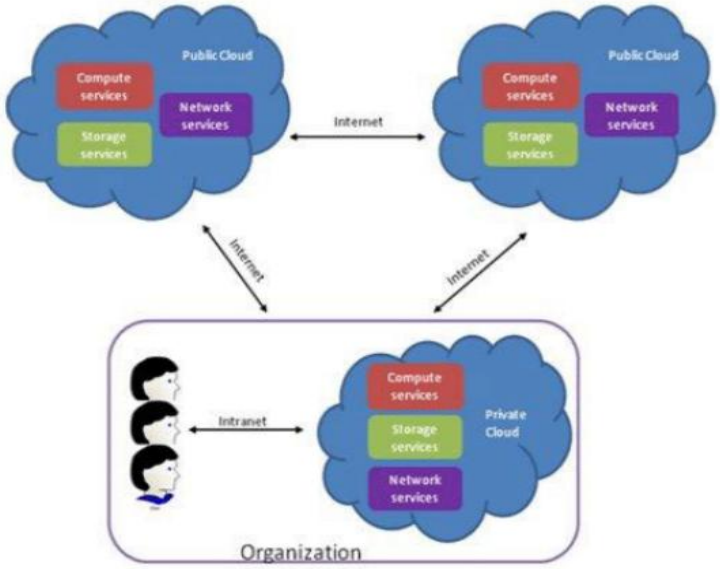

Fig. 4 Types of clouds[6]

Some other types of clouds are-

(i) Community Cloud

Community cloud shares infrastructure between several organizations from a specific community or businesses with common concerns such as security managed rather internally or third-party. The cost is spread over fewer users than public cloud (but mode than private cloud).

(ii) Distributed cloud

A cloud computing platform assembled from a distributed set of machines in different

geographical locations, connected and projected as a single network or service.

(iii)Multi-cloud

Multi-cloud is the use of multiple cloud computing services together as a single service with heterogeneous architecture to increase flexibility through choice. It differs from hybrid cloud as it has multiple cloud services rather than multiple deployment modes.

2.3 Service Models

Cloud-computing providers offer their services according to different models, of which the three standard models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These services offer increasing abstraction thus portrayed as layers in a stack.

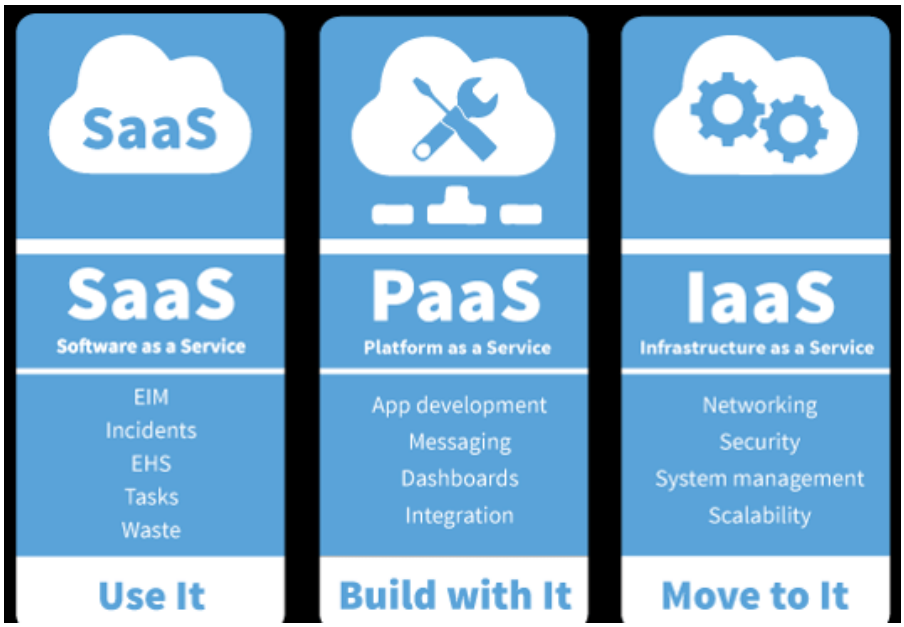

Fig. 5 Services in a cloud[4]

2.3.1 Infrastructure as a Service

At a basic level, IaaS public cloud providers offer storage and compute services on a pay-per-use basis. You rent the basic infrastructure like servers, machines, network, storage, OS from a cloud provider on a pay-per-use basis. Resources are supplied on demand from their large pools of equipments installed in data centres. For wide area connectivity, customers can use the internet. Example: Amazon EC2, Microsoft Azure etc.

2.3.2 Platform as a Service

PaaS vendors offer development environments to the application developers. The cloud providers deliver a computing platform, typically including OS, Programming language execution environment, database and web-server. Application developers develop and run their software on these platforms instead of buying and managing the underlying layers. With PaaS, the underlying resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually. It is designed to make it easier for developers to quickly create web or mobile apps, without worrying about the setup for the environment or underlying infrastructure of server, storage, networks, or databases needed for development. Example: Google app Engine.

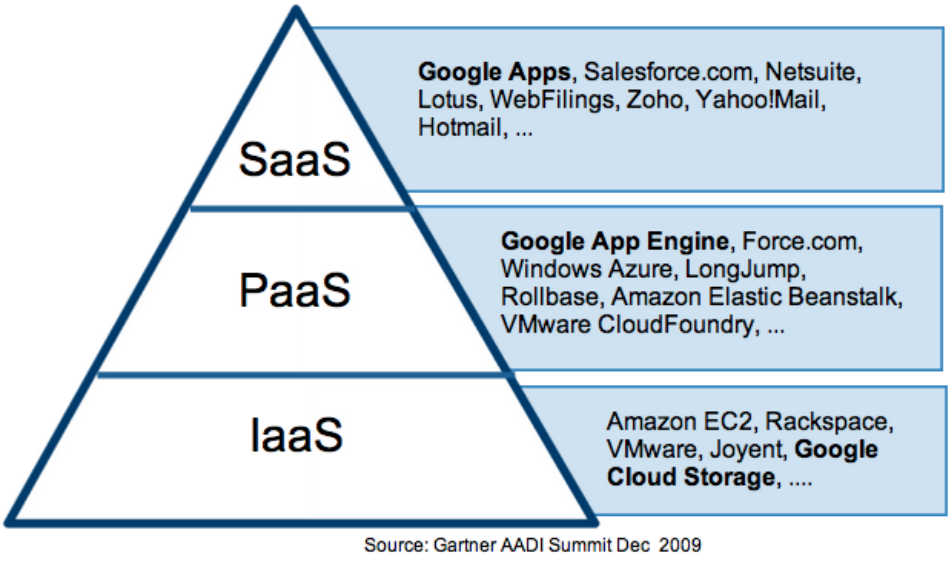

Fig. 6 Cloud services model and their examples[11]

2.3.3 Software as a Service

Often referred as on-demand software, SaaS is usually the software build and installed on the servers by the service providers and made available to the users on pay-per-use kind of method or some other. It is a method for delivering software applications over the internet, on demand and typically on subscription basis. With SaaS, cloud providers host and manage the software application and underlying infrastructure and handle any maintenance, like software upgrades, security patching. Users connect to application over the internet, usually with a web browser on their phones, table or PC.

Example: Google Apps.

2.4 Architecture Of Cloud

Cloud architecture involves multiple cloud components communicating with each other over a loose coupling mechanism. A simple architechture is as given below-

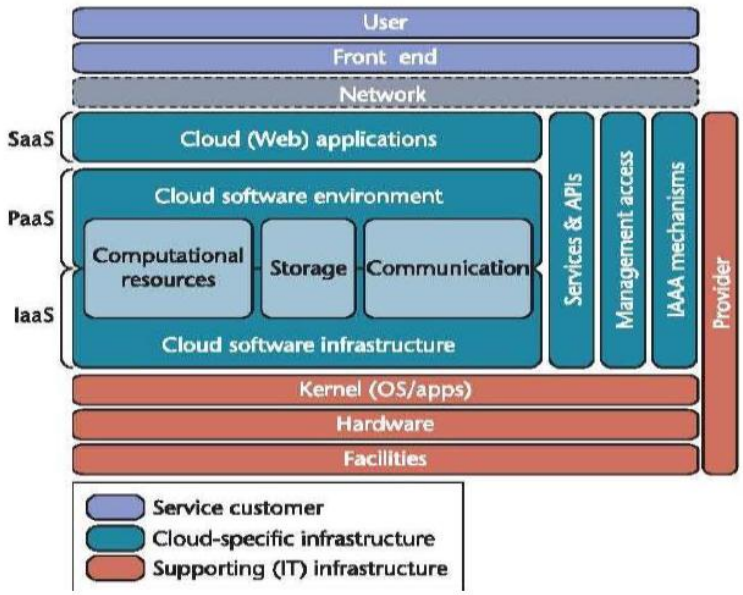

Fig. 7 Cloud computing architecture[5]

2.5 Security And Privacy

Cloud computing becomes a successful and popular business model due to its charming features. In addition to the benefits at hand, the features also result in serious cloud-specific security issues. The people whose concern is the cloud security continue to hesitate to transfer their business to cloud. Security issues have been the dominate barrier of the development and widespread use of cloud computing. There are three main challenges for building a secure and trustworthy cloud system :

(i) Outsourcing – Outsourcing brings down both capital expenditure (CapEx) and operational expenditure for cloud customers. However, outsourcing also means that customers physically lose control on their data and tasks. The loss of control problem has become one of the root causes of cloud insecurity. To address outsourcing security issues, first, the cloud provider shall be trustworthy by providing trust and secure computing and data storage; second, outsourced data and computation shall be verifiable to customers in terms of confidentiality, integrity, and other security services. In addition, outsourcing will potentially incur privacy violations, due to the fact that sensitive/classified data is out of the owners' control.

(ii) **Multi-tenancy** – Multi-tenancy means that the cloud platform is shared and utilized by multiple customers. Moreover, in a virtualized environment, data belonging to different customers may be placed on the same physical machine by certain resource allocation policy. Multi-tenancy is a definite choice of cloud venders due to its economic efficiency, it provides new vulnerabilities to the cloud platform. Without changing the multi-tenancy paradigm, it is imperative to design new security mechanisms to deal with the potential risks.

(iii) **Massive data and intense computation** – cloud computing is capable of handling mass data storage and intense computing tasks. Therefore, traditional

security mechanisms may not suffice due to unbearable computation or communication overhead. For example, to verify the integrity of data that is remotely stored, it is impractical to hash the entire data set. To this end, new strategies and protocols are expected.

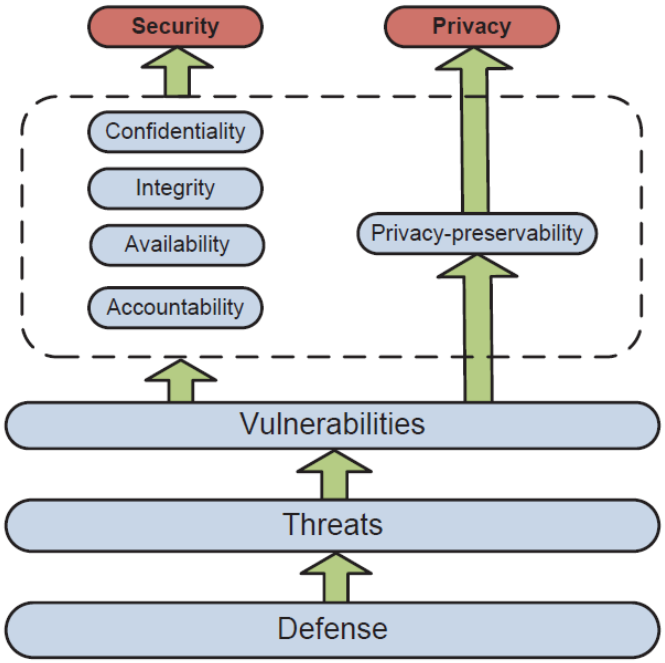

Fig. 8 Cloud Security and privacy[8]

2.5.1 Good Security Practices in Cloud Computing Environment

- Protection against internal and external threats- Security monitoring services help improve the effectiveness of the security infrastructure of the cloud. Monitoring teams correlate the imformation from various security devices to provide security analysts with the data they need to eliminate[7].

- Early Detection- An early detection ervice detects the new security vulnerabilities shortly after they appear. The threats generally are associated with the third party sources, and an alert or report is issued to customers[7].

2.5.2 Ensuring Privacy in cloud computing environment

Privacy in cloud computing environment is less of a technical issue and more of a Policy and Legal issue. Policies are required to be framed to conform to the legal framework protecting the privacy of the individuals and the organizations. Policies have to empower people to control their collection, use, and distribution of their personal information.

- Notice- The users must be notified about how their data will be collected, and used by the organization.

- Choice- Individuals must have the ability to opt out.

- Onward transfer- Transfer of data to the third parties must follow adequate data protection rules.

- Data Integrity- Data must be relevant and reliable for the purpose it was collected for.

- Access- Individuals must be able to access their information, update it and delete it if it is inaccurate.

3. EXISTING FRAMEWORKS AND THE USED FRAMEWORK

a) OPEN CLOUD TESTBED

Open Cloud Testbed (OCT) was proposed by Grossman to mainly benchmark different cloud computing systems, to investigate their interoperability. It is currently configured as a smaller-scale testbed. It is use to develop cloud computing software and infrastructure.

DRAWBACK- Not an open source framework and is limited(Registration Required).

b) OPEN CIRRCUS

Yahoo and HP have led the establishment of a global Cloud computing testbed, called Open Cirrus, supporting a federation of data centers located in 10 organizations [16]. Building such experimental environments is expensive and hard to conduct repeatable experiments as resource conditions vary from time to time due to its shared nature.

DRAWBACK- their accessibility is limited to members of this collaboration.

4. SYSTEM REQUIREMENTS

Any computer system with a dual-core processor, 2 GB RAM, and 1 GB storage is good enough to simulate the cloud-based systems using the cloudsim as this is required to support the JRE working. Also, the hardware requirements may differ on the basis of which IDE you are using for JAVA development. For complete setup procedure, refer Appendix A.

Now before setting up CloudSim, following resources must be installed/downloaded on the local system. Cloudsim simulation toolkit setup is easy.

- **Java Development Kit(JDK):** As the Cloudsim simulation toolkit is a class library written in the Java programming language, therefore, the latest version of Java(JDK) should be installed on your machine, which can be downloaded from Oracles Java portal. For assistance in the installation process, detailed documentation is provided by Oracle itself.

- **Eclipse IDE for Java developers:** As per your current installed operating system(Linux/Windows). Before you download to make sure to check if 32-bit or 64-bit version is applicable to your Computer machine.

- To handle errors while setup, refer Appendix B.

5. CLOUDSIM FRAMEWORK

Several simulators can be used to simulate the working of new services, among them CloudSim Simulation Toolkit is the more generalized and effective simulator for testing Cloud computing-related hypothesis. CloudSim is an extensible simulation framework developed by a team of researchers (under the guidance of Dr. Raj Kumar Buyya) at Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. This toolkit allows seamless modelling, simulation, and experimentation related to cloud-based infrastructures and application services.

This simulation toolkit allows the researchers as well as cloud developers to test the performance of the potential cloud application for performance testing in a controlled and easy to setup environment**.** And Also allows fine-tuning the overall service performance even before it is deployed in the production environment.

FEATURES OF CLOUDSIM SIMULATION TOOLKIT-

(i) Support for modelling and simulation of large-scale Cloud computing environments, including data centres, on a single physical computing node (could be a desktop, laptop, or server machine).

(ii) A self-contained platform for modelling Clouds, service brokers, provisioning, and allocation policies.

(iii)Facilitates the simulation of network connections across the simulated system elements.

(iv)Facility for simulation of federated Cloud environment that inter-networks resources from both private and public domains, a feature critical for research studies related to Cloudbursts and automatic application scaling.

(v) Availability of a virtualization engine that facilitates the creation and management of multiple, independent, and co-hosted virtualized services on a data centre node.

(vi) Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

5.1 CloudSim Architecture And Modelling The Cloud

The diagram below demonstrates the layered architecture of CloudSim Simulation Toolkit. The CloudSim Core simulation engine provides support for modeling and simulation of virtualized Cloud-based data center environments including queuing and processing of events, creation of cloud system entities (like data center, host, virtual machines, brokers, services, etc.) communication between components and management of the simulation clock.

The CloudSim layer provides dedicated management interfaces for Virtual Machines, memory, storage, and bandwidth. Also, it manages the other fundamental issues, such as provisioning of hosts to Virtual Machines, managing application execution, and monitoring dynamic system state(e.g. Network topology, sensors, storage characteristics, etc), etc.

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. The User Code layer is a custom layer where the user writes their own code to redefine the characteristics of the stimulating environment as per their new research findings.

The services 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.

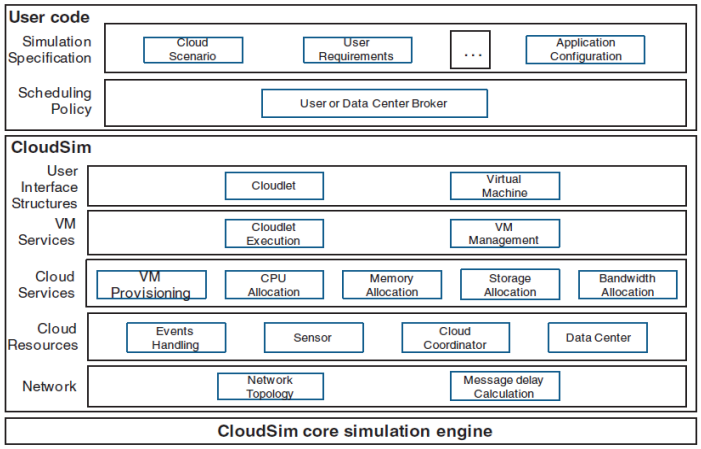

Fig. 9 Architecture of CloudSim framework [1]

A data center can manage several hosts that in turn manage 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.

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. 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). By default, VmAllocationPolicy implements a straightforward policy that allocates VMs to the Host on a First-Come-First-Serve (FCFS) basis. Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in 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.

5.2 Design And Implementation of The CloudSim

The overall class design for the CloudSim, which are also the building blocks of the simulator is as:

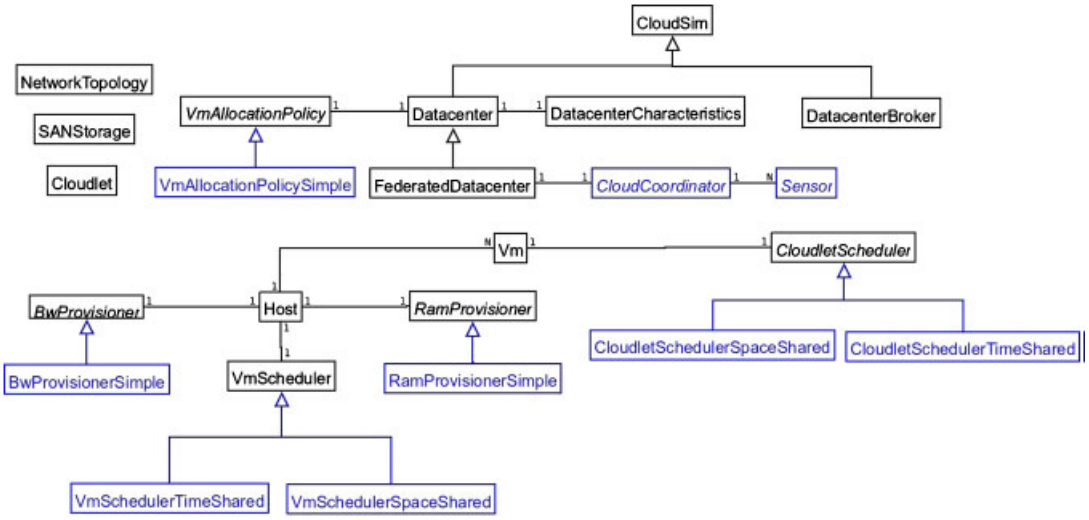

Fig. 10 CloudSim class diagram[1]

The description of all the major classes is as given below:

a) **Cloudlet**: This class model(define specific attributes such as length of instruction, input/output filesize, no of processor required, etc) the Cloud-based application services (program based tasks) such as content delivery, social networking, etc. CloudSim implements the complexity of an application in terms of its computational

requirements. As a developer, we know that every individual executable application/service workload has a pre-defined instruction length and requires certain network data flow (both pre and post fetches) overhead that it needs to undertake during its life cycle. this class allows modeling all the above-said requirements. (Refer Appendix C)

b) **CloudletScheduler**: This is responsible for the implementation of different policies that determine the share of processing power among Cloudlets in a VM. There are two types of provisioning policies offered: space-shared (using CloudetSchedulerSpaceShared class) and time-shared (using CloudletSchedulerTimeShared class**).**

c) **Datacenter** : This class model the core infrastructure-level services (i.e. hardware) that are offered by Cloud providers (Amazon, Azure, and App Engine). It encapsulates a set of hosts(resembling server machine model) instances that can either be homogeneous or heterogeneous concerning their hardware configurations (memory, cores, capacity, and storage). Also, every Datacenter component takes care of generalized application provisioning that enforces a set of policies for the allocation of bandwidth, memory, and storage devices to hosts and its related VMs.

d) **DatacenterBroker** : This class model a broker, which is responsible for mediating negotiations between SaaS and Cloud providers and such negotiations are driven by QoS requirements.

e) **DatacenterCharacteristics**: This class contains configuration information of data center resources like the available host list, the fine-grained cost for each resource type, etc.

f) **Host**: This class models a physical resource such as a computer or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (if it is a multi-core machine), an allocation of policy for provisioning the compute, memory and bandwidth to the VMs.

g) **RamProvisioner**: This is an abstract class that represents the provisioning policy for allocating primary memory (RAM) to Virtual Machines. The execution and deployment of VM on a host are 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. The RAM resource request will be rejected if it is beyond the available capacity.

h) **BwProvisioner**: 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 policies (priority, QoS) to reflect the needs of their applications.

i) **Vm**: This class model a Virtual Machine (VM), which is managed and hosted by a Cloud host component. Every VM component has access to a component that stores the following characteristics related to a VM (i.e.) accessible memory, processor, storage size, and the VM's internal provisioning policy that is extended from an abstract class called the CloudletScheduler.

j) **VmAllocationPolicy**: This is an abstract class that represents a provisioning policy to be utilized by VM Monitor for mapping VMs to hosts. The primary role is to select the best fit host in a data center that meets the memory, storage, and availability requirement for VM deployment mapping.

k) **VmScheduler**: This is an abstract class implemented by a Host component that models the allocation policies (space-shared, time-shared) defining the rules for processor cores allocations to VMs. To accommodate application-specific processor sharing policies, the class functionality can be extended to define a new set of provisioning rules.

l) **CloudSim**: This is the prime class, with the role of managing entity event queues and controlling the sequential execution of simulation events. Every event that is generated by the CloudSim entity at run-time is stored in the queue called future events. These events are sorted by their time parameters and are enqueued into the future queue. Next, the events that are scheduled at each step of the simulation are removed from the future events queue and transferred to the deferred event queue. Following this, an event processing method is invoked for each entity, which chooses events from the deferred event queue and performs appropriate actions. Such an organization allows flexible management of simulation and provides the following powerful capabilities:

- Deactivation (hold/pause) of entities.
- Context switching of entities between different states (e.g. waiting to active). Pause and resume the process of simulation.
- Creation of new entities at run-time.
- Aborting and restarting simulation at run-time.

m) **FutureQueue**: This class implements the future event queue accessed by CloudSim and ***acts as a ready queue to the simulation engine***.

n) **DeferredQueue**: This class implements the deferred event queue used by CloudSim and hold such events which are failed or paused. ***It acts as a wait queue of the simulation engine, where preempted resource requests are kept.***

o) **CloudInformationService**: A CIS is an entity that provides resource registration, indexing, and discovering capabilities. CIS supports two basic primitives:

- **publish()**, on the start of simulation it allows entities to register themselves with CIS
- **search()**, allows Brokers in discovering resources status and endpoint addresses of other entities. This entity also acts as a notification service to the other entities about the end of the simulation.

p) **SimEntity**: This is an abstract class, which represents a simulation entity (such as DataCenter, DatacenterBroker, etc)) that is able to send messages to other entities and processes received messages as well as fire and handle events. All entities must extend this class and override its three core methods:

- **startEntity**(), which define actions for entity initialization.
- **processEvent**(), which defines actions processing of each called event(s) with respect to the entity.
- **shutdownEntity**(),which define actions for entity destruction.

q) **CloudSimTags**. This class contains various static event/command tags that indicate the type of action that needs to be undertaken by CloudSim entities when they receive or send events.

r) **SimEvent**: This entity represents a simulation event that is passed between two or more entities. SimEvent stores the following information about an event:

- type,
- init time,
- time at which the event should occur,
- finish time,
- time at which the event should be delivered to its destination entity,
- IDs of the source and destination entities,
- the tag of the event, and
- Data that have to be passed to the destination entity.

s) **CloudSimShutdown**: This is an entity class that waits for the termination of all end-user submitted cloudlets(tasks) and broker entities events, and once detected, then signals the end of simulation to CIS.

5.3 CloudSim Project Structure

The Cloudsim Simulation Toolkit is an API that is written using the Java programming language and its classes are structured in a very specific way. Let's have a look at the Cloudsim project structure and understand how the Cloudsim architecture is divided into different packages and their important classes that facilitate the cloud

simulation using Cloudsim.

a) The eclipse project explorer should contain 6 folders and 7 files.

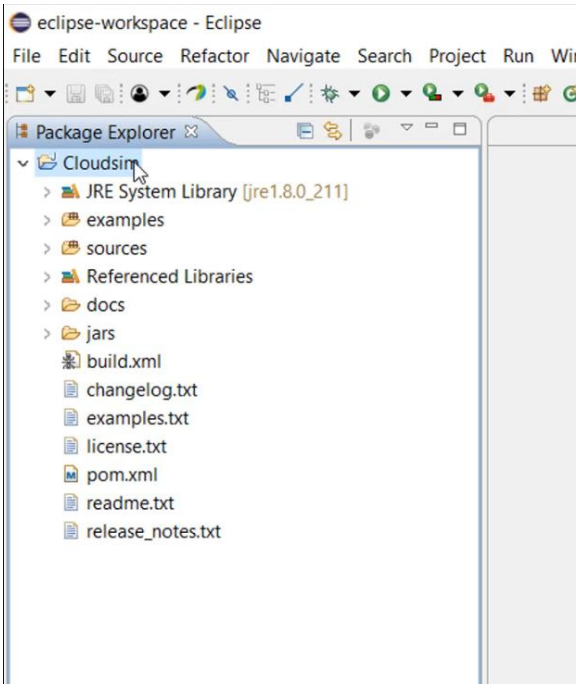

Fig. 11 Package explorer in Eclipse

b) The readme.txt file contains the necessary information about this project, like how to install and run the cloudsim, what is the basic directory structure.

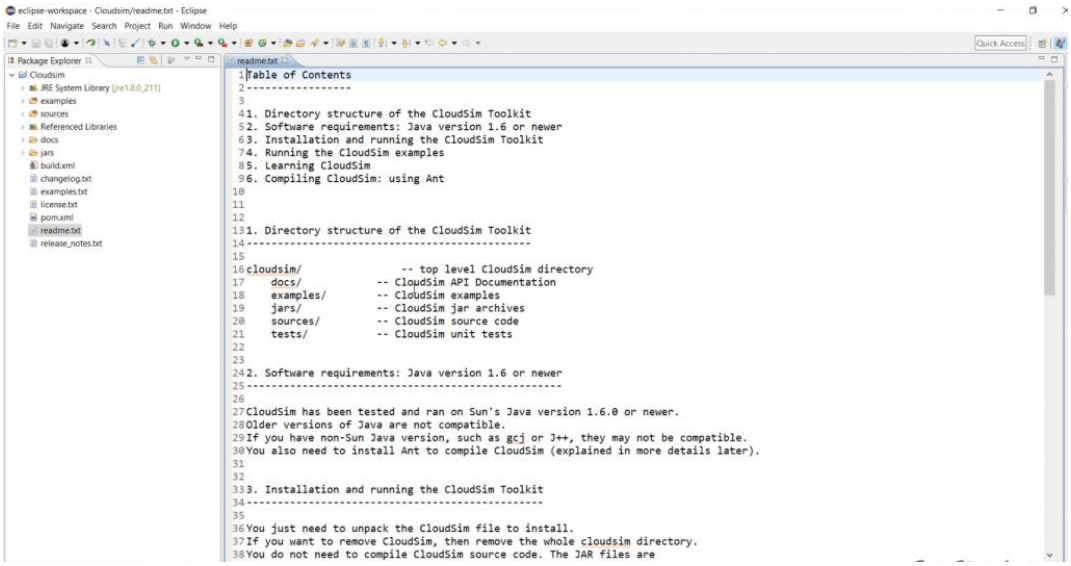

Fig. 12 Readme.txt in CloudSim

c) The directory structure in detail and understand the various namespaces available in the “Source” folder.

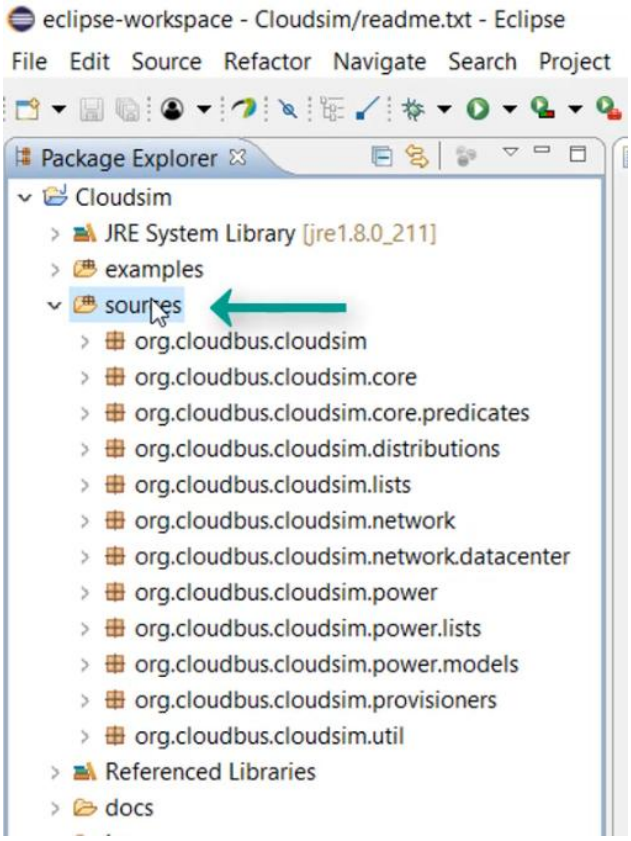

Fig. 13 Namespaces in the sources folders

d) There exist namespaces; each namespace has a set of classes with specific correlated functions as discussed below:

- **Org.cloudbus.cloudsim:** It contains the model classes of various basic hardware components, their groups, and the allocation/utilization methods, etc. Here, the model means that the classes contain the implementation of the various attributes and behaviors of the real-life cloud computing hardware component as a collection of Java methods. Once these classes initiate during the simulation, they are going to simulate the behavior of the real-life cloud-based system component.

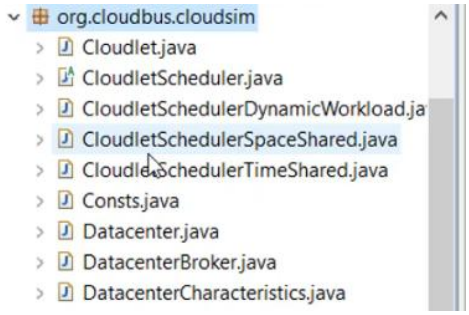

Fig. 14 Classes under org.cloudbus.cloudsim package

- **Org.cloudbus.cloudsim.core:** This namespace contains the implementation of the simulation engine where the cloudsim.java class is the main class and is responsible for the start and stop of the simulation process. Similarly, the simentity.java class is for maintaining the state of the simulated cloud components. Simevent.java, futurequeue.java, and defferedqueue.java are responsible for maintaining the inter-entity(cloud component) related event call during the simulation process

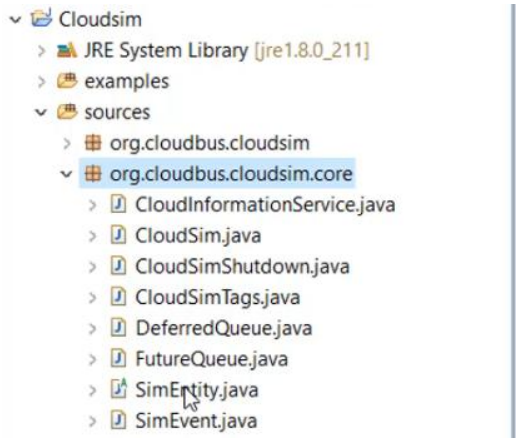

Fig. 15 Classes under org.cloudbus.cloudsim.core namespace

- **Org.cloudbus.cloudsim.lists:** It contains the implementation of lists to be used globally during the simulation process. These are the specialized set of classes, where sorting and comparison operation implemented. There exist the list class for the host, processing element, cloudlet, virtual machine, and resources.

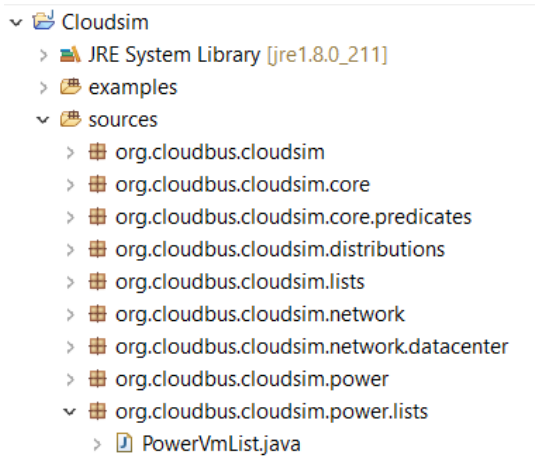

Fig. 16 Classes under org.cloudbus.cloudsim.lists namespce

- **Org.cloudbus.cloudsim.network.datacenter:** This namespace contains the extended implementation of the basic cloud hardware components which can support the simulation of federated cloud functions distributed across the regions.

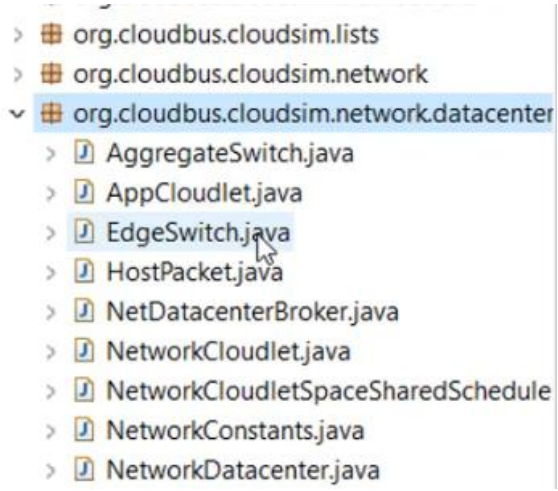

Fig. 17 classes under datacenter namespace

- **Org.cloudbus.cloudsim.power:** This package contains the extended implementation of cloud components, which can simulate the green or power-aware computing scenarios. This namespace implements the VM migration related classes where extensive algorithm codes for virtual machine selection and allocation related to migrations are available.

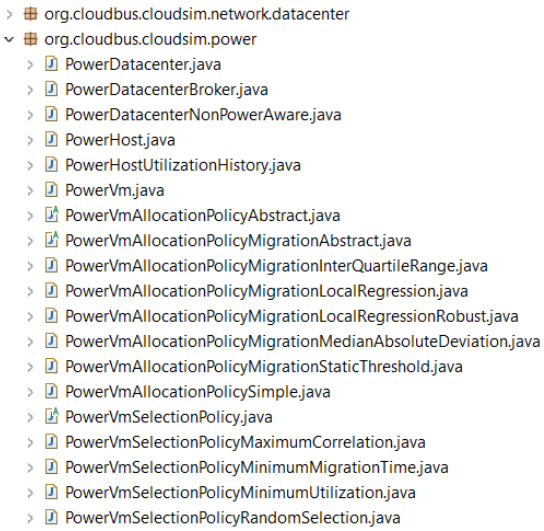

Fig. 18 Extended cloud components

- **Org.cloudbus.cloudsim.provisioners:** This namespace contains the behavior implementation regarding how the specified cloud component provisioned to requesting virtual resources.

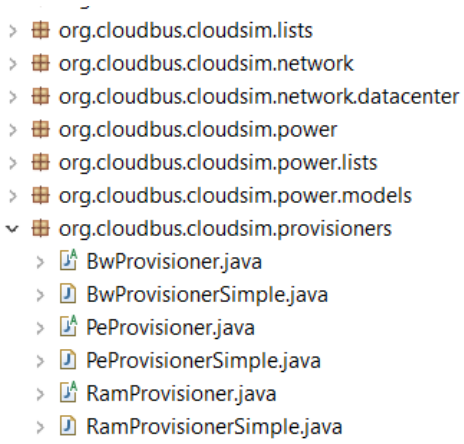

Fig. 19 Classes for requesting the vital resources

e) **Reference Libraries:** It contains the various dependency jars again automatically added by the eclipse during the first build process.

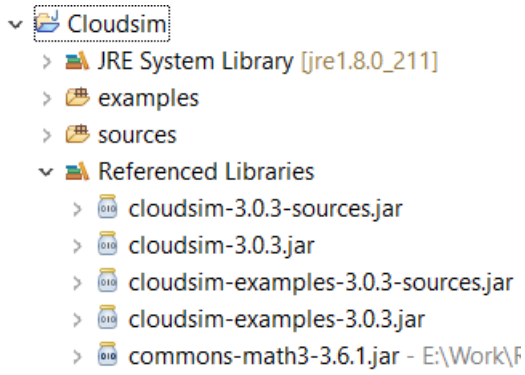

Fig. 20 CloudSim reference libraries

f) **Jars:** This folder is supplied with the source zip file and contains the pre-compiled build of the cloudsim 3.0.3. It can be used for creating custom project scenarios whenever required. We will work on this in further blog posts.

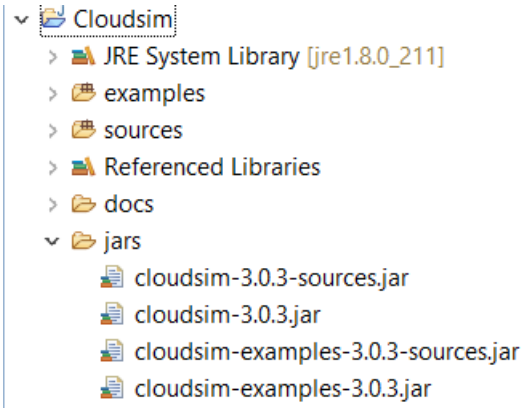

Fig. 21 CloudSim jar files

5.4 Cloudlet In Cloudsim Simulation

Cloudlet in Cloudsim defined the workload, which is to be executed during the simulation run of the cloudsim simulation engine. Cloudlet in Cloudsim is a model class

that exists inside the package 'org.cloudbus.cloudsim'. Cloudlet is one of the most important models which defined the specifications for a simulation engine corresponding to the real-life candidate application to be considered for moving to a Cloud-based system. (Refer Appendix C).

Cloudlet in Cloudsim defines following are the attributes that are being by the cloudsim simulation engine:

a) **userId:** the broker id to through which this cloudlet will be executed and the result of execution will be returned & displayed on the console through the same broker.

b) **cloudletLength:** defines the length of the cloudlet in terms of the expected number of instructions required to be executed in the lifetime of the workload under execution.

c) **cloudletFileSize:** As the cloudlet is supposed to be executed over the cloud that means the input data transfer over the network should be happening during the execution. This attribute defines the total size of the input data in bytes.

d) **cloudletOutputSize:** Similar to cloudletFileSize the output data size is also defined in bytes.

e) **numberOfPes:** Defines the number of processors required to execute this cloudlet, if parallel processing is required to be performed.

f) **cloudletId:** used as a unique identification number for cloudlets, required for referencing throughout the simulation flow.

g) **status:** defines the current state of the cloudlet. A cloudlet could be in any one of the state as follows: CREATED, READY, QUEUED, INEXEC, SUCCESS, FAILED, CANCELED, PAUSED, RESUMED, FAILED_RESOURCE_UNAVAILABLE. These states are defined explicitly in cloudlet model class and are assigned a numerical value.

h) **num:** defined the time at various stages of cloudlet execution

i) **execStartTime:** Defines the latest execution time and is updated every time the status of cloudlet shuffle among CANCEL, PAUSED and RESUMED

j) **finishTime:** Stores the time, when the execution of the cloudlet is finished and status is not a success.

k) **reservationId:** defines the ID of the resource reserved for which this cloudlet.

l) **record:** defines in boolean(True/False) if the history of the cloudlet execution across different resources is maintained or not.

m) **newline:** Defined newline characteristics and used during the string concatenations while logging the history of cloudlet execution.

n) **history:** maintains the history of the cloudlet execution across simulation flow.

o) **resList:** Maintains the log of the resources where the cloudlet is being executed.

p) **index:** defines the ID of the resource on which the cloudlet is being currently assigned and helps in getting the status

q) **classType:** Defines the resource scheduling information(never used in cloudsim 3.0.3)

r) **netToS:** Defines the type of service for the network transactions. This is a field for IPv6 for ensuring the quality of service.

s) **vmId:** stores the Id of the virtual machine to which this cloudlet is to be allocated for allocation. This is done in the case where you are planning to move a real-life application to a cloud-based system with a specific Virtual machine configuration. Such mapping of cloudlet helps in determining the behavior of the application in various scenarios as simulated over the cloudsim simulation engine.

t) **costPerBw:** defines the cost associated with the usage of the network for cloudlet network usage. this is overridden if the cost on bandwidth is defined by the datacenter instance which is always done.

u) **accumulatedBwCost:** Stores the overall cost of bandwidth consumed by the cloudlet while processing during the simulation. This can be used to check what is the cost involved in network bandwidth for a particular candidate application which is under consideration to be moved to the cloud-based system.

v) **utilizationModelCpu/utilizationModelRam/utilizationModelBw:** these attributes defines the type of model to be used for executing the defined cloudlet. It may individually defined for each attribute from following: UtilizationModelFull, UtilizationModelNull, UtilizationModelStochastic and UtilizationModelPlanetLabInMemory(used in power-aware workloads only).

w) **requiredFiles:** defines the path for the list of files to be required by the cloudlet during execution. These are the input files and are directly associated with the network bandwidth cost.

Following are the states that each cloudlet can sustain during the simulation flow:

a) **CREATED:** Defined when the cloudlet is created and added to the CloudletList(used by the datacenter broker during the simulation flow)

b) **READY:** Defined when the cloudlet is assigned with a resource for execution

c) **QUEUED:** Defined when the cloudlet is moved to the cloud resource on which it will wait for its execution

d) **INEXEC:** Defined when the cloudlet is put into execution on the assigned cloud resource;

e) **SUCCESS:** Defined when the cloudlet execution is successfully completed and is ready to be sent back to the broker with its final status

f) **FAILED:** Defined in case the cloudlet is failed to be executed to the planned cloud node e.g. when the VM for which the cloudlet is to be assigned is failed to start due to any reason the cloudlet execution will be failed.

g) **CANCELED:** Defined, when the execution of the cloudlet is canceled due to any situation and can be further denoted in the failed state is required

h) **PAUSED:** Defined, when the cloudlet execution is put on hold. This could be due to some high priority cloudlet arrival or due to VM migration etc.

i) **RESUMED:** Defined to resume the cloudlet execution after it is being paused.

j) **FAILED_RESOURCE_UNAVAILABLE:** Defined when the cloudlet execution failed due to the failure of the resource on which it was supposed to be executed.

The cloudlets would shuffle between these states any number of times depending upon the simulation scenario under consideration.

One of the most important behaviors of the Cloudlet model class is that it may be moved from one resource to another resource due to any implicit/explicit reasons, therefore, the track is to be maintained for its movement during the simulation flow. To support this a nested class named 'Resource' is defined which contains the following attributes:

a) **submissionTime:** defines when the cloudlet was submitted to the provided cloud resource instance.

b) **wallClockTime:** defines the total time that a cloudlet spent of the currently provided cloud resource.

c) **actualCPUTime:** defines the actual time spent under the execution and may or may not differ from wallClockTime.

5.5 Virtual Machine And Task Scheduling In Cloudsim

Let's understand the various classes and hierarchies provided in the CloudSim for

scheduling of virtual machines and cloudlets.

5.5.1 Basics of scheduling

In computers, Scheduling is a process of arranging the submitted jobs/task into a very specific sequence of execution. It is an essential characteristic of any software operating environment, which is handled by a very special program known as a scheduler.

Scheduler's main objective is to keep the underlined hardware resources(primarily processor) to be used effectively as well as efficient. In general, the scheduler may prefer to have any of the following scheduling approaches:

- **Space-shared:** In this, the requested resources are allocated dedicatedly to the requesting workload for execution and will be released only on completion. **Space-shared is also known as a batch process scheduling.**

- **Time-shared:** In this, the requested resources would be shared among more than one workload(task). The sharing is done based on time-sliced allocation where each workload is allocated with a required resource for a defined time(e.g., 200 milliseconds). Once the defined time slice is over, the current workload execution paused, and the resource is released. The released resource gets allocated to the next workload for the same defined time slice, and this cycle goes on till the time all the workloads execution is over. Time-shared is also known as round-robin scheduling.

5.5.2 Scheduling in cloud

As cloud computing is the virtualized operating environment, and the virtual machines are the primary computing component which is responsible for the execution of the workloads (tasks). The virtual machine(s) are powered by a physical server host machine (i.e.) hardware. Depending on the requirement of the Virtual Machine (VM) there could be **'**one to one' or 'many to one' mapping between the VM

and host machine. That means in cloud computing the scheduling is done at both the mapping levels that are:

- Virtual Machine to Host Machines
- Tasks to Virtual Machines

5.6 Scheduling in CloudSim

The CloudSim simulation toolkit framework has effectively addresses the Scheduling scenario and implemented it as a set of the programmable class hierarchies with parent class as:

- VmScheduler
- CloudletScheduler

Also, Virtual Machine (VM) and Task (Cloudlet) scheduling are one of the most important and the popular use case to be simulated by researchers using the CloudSim simulation toolkit.

5.6.1 Virtual Machine Scheduling

The **VmScheduler** is an abstract class that defines and implements the policy used to share processing power among virtual machines running on a specified host. The hierarchy of the cloudsim virtual machine scheduler classes is as:

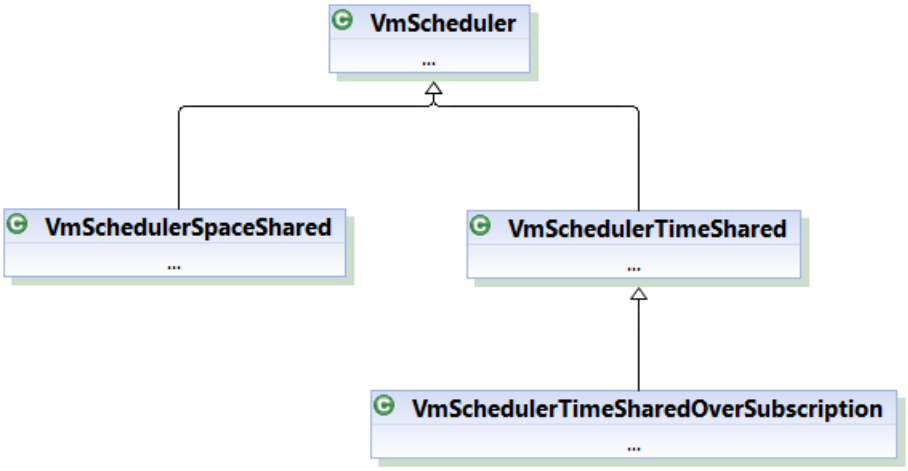

Fig. 22 VmScheduler class hierarchy [1]

These classes can be located in "*org.cloudbus.cloudsim*" package of cloudsim. The definition of this abstract class is extended to the following types of policies implemented as classes:

- **VmSchedulerTimeShared**: This class implements the VM scheduling policy that allocates one or more processing elements to a single Virtual machine and allows the sharing of processing elements by multiple virtual machines with a specified time slice. This class also considers the overhead of VM allocation switching(similar to context switching) in policy definition. Here, the VM allocation will fail if the number of processing elements requested is not available. for example, if the VM request for quad-core processor whereas the allocated host has an only dual-core the allocation will fail.

- **VmSchedulerSpaceShared**: This class implements the VM scheduling policy that allocates one or more processing elements to a single virtual machine, but this policy implementation does not support sharing of processing elements (i.e.) all the requested resources will be used by the allocated VM till the time the VM is

not destroyed. Also, Under this allocation policy, if any virtual machine requests a processing element and is not available at that time, the allocation fails.

- The application of the VmScheduler classes is while instantiating the host model. Following is the code snippet:

```
int hostId = 0;
int ram = 2048; // host memory (MB)
long storage = 1000000; // host storage
int bw = 10000;


hostList.add(
                new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList,
                new VmSchedulerTimeShared(peList)
        )
); // This is our machine
```

5.6.2 Cloudlet Scheduling

The "***CloudletScheduler***" is an abstract class that defines the basic skeleton to implement the policy to be used for cloudlet scheduling to be performed by a virtual machine. The hierarchy of the cloudsim Cloudlet scheduler classes is as:

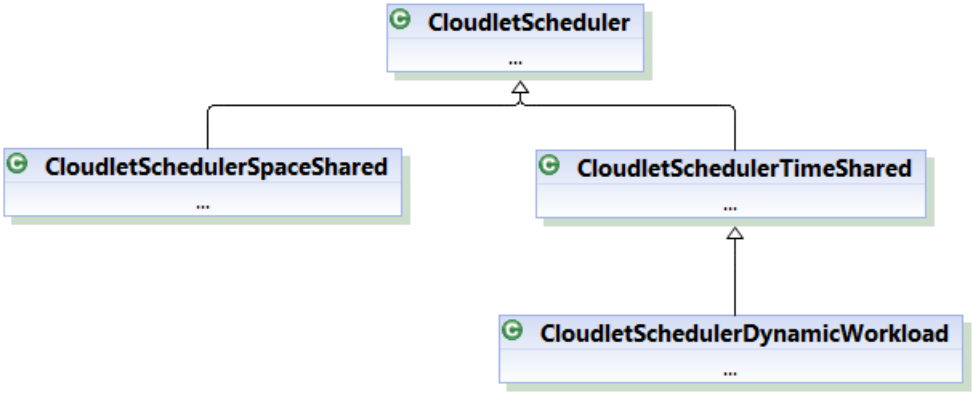

Fig. 23 CloudletScheduler class hierarchy [1]

These classes exist in "*org.cloudbus.cloudsim*" package of cloudsim. The definition of this abstract class is extended as the following types of policies implemented as these individual classes in cloudsim:

- **CloudletSchedulerSpaceShared**: This class implements a policy of scheduling for Virtual machine to execute cloudlet(s) in space shared environment (i.e.) only one cloudlet will be executed on a virtual machine at a time. It means cloudlets share the same queue and requests are processed one at a time per computing core. *Space-sharing* is similar to batch processing.

- **CloudletSchedulerTimeShared**: This class implements a policy of cloudlet scheduling for Virtual machines to execute cloudlets in a time-shared environment (i.e.) more than one cloudlet will be submitted to the virtual machine and each will get its specified share of time. It means several requests (cloudlets) are processed at once but they must share the computing power of that virtual machine(by simulating context switching), so they will affect each other's processing time. It basically influences the completion time of a cloudlet in CloudSim. Time-sharing is probably referring to the concept of sharing executing power (such as

CPU, logical processor, GPU) and is commonly known as the round-robin scheduling.

- The application of the CloudletScheduler classes is while instantiating the Vm model. Following is the code snippet:

```
int vmid = 0;

int mips = 1000;

long size = 10000; // image size (MB)

int ram = 512; // vm memory (MB)

long bw = 1000;

int pesNumber = 1; // number of cpus

String vmm = "Xen"; // VMM name



Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared()); // create VM
```

5.7 Flow In Cloudsim

Whenever we run any case in the CloudSim, the following things are done in the following order-

(i) The CIS (Cloud Information System) is created which contains all the information regarding the resources available on the Data Centre. Each data centre will have some hosts and each host further can have some or one virtual machine. The CIS takes care f the registry of the data centres.

(ii) The data centre will have some hosts, and each host will have some configurations and characteristics as processing elements(PES), RAM, BandWidth etc.

(iii)The virtual machines inside the hosts further will have some parameters(according to virtualization).

(iv) Whenever a data centre is created, we need to register it under th CIS.

(v) Further, we need to have a broker(DataCenterBroker class) which is responsible for submitting the tasks(cloudlets) to the data centre.

(vi) The broker firstly communicates with the CIS to get the information related to the resources in the data centre(characteristics).

(vii) All the tasks known as the cloudlets are submitted to the broker and the broker assigns them to the machines on the hosts.

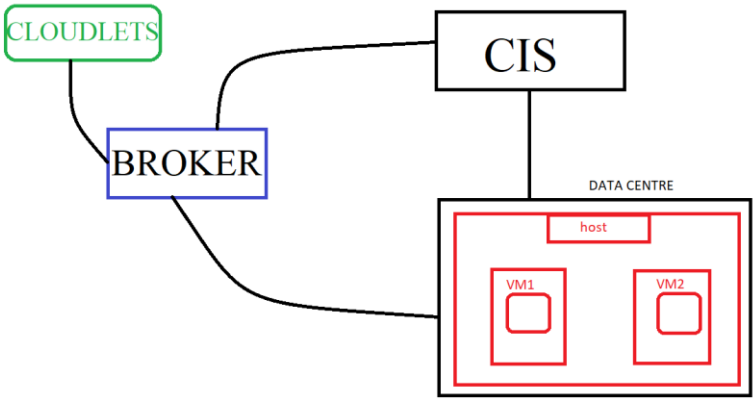

Fig. 24 Components and their interactioon in CloudSim

6. CASE STUDIES, RESULTS AND OUTPUTS

In this section various use cases will be examined and shown for different features of cloudlets and Virtual Machines. For example, we can have scheduling of the cloudlets as FCFS(first come first serve) and VM allocation policy also as FCFS which is whichever machine is available at the time when the cloudlet comes will be allotted to it. The characteristics and features of the VM and the CLOUDLETS as as below-

Table 1 Characteristic of the cloudlets

Length	**300MI**
File Size	300MI
Output Size	300MI
CPUs	1
Utilization Model	Full
Scheduler	Space shared

cloudletId-the unique ID of this Cloudlet

cloudletLength-the length or size (in MI) of this cloudlet to be executed in a Datacenter

cloudletFileSize-the file size (in byte) of this cloudlet BEFORE submitting to a Datacenter

cloudletOutputSize-the file size (in byte) of this cloudlet AFTER finish executing by a Datacenter

Table 2 Characteristics of the VMs

Size(storage)	**1000MB**
RAM	512MB
MIPS	1000
BW	1000
CPU	1
Scheduler	Time Shared

CASE 1-

VM allocation policy - First Come First Serve

Cloudlets Scheduling Policy - First Come First Serve

VM Scheduler - Time Shared

Cloudlet Scheduler -Space Shared

Cloudlets Utilization Model -FULL

No. Of VMs=10

No. Of Cloudlets=100

Table 3 CASE 1 sample outputs

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
9	SUCCESS	3	9	0.51	0.2	0.71
7	SUCCESS	3	7	0.84	0.2	1.04
2	SUCCESS	2	2	0.93	0.2	1.13
1	SUCCESS	2	1	1.24	0.2	1.44
17	SUCCESS	3	7	0.41	1.04	1.45
3	SUCCESS	2	3	1.35	0.2	1.55
5	SUCCESS	2	5	1.35	0.2	1.55
8	SUCCESS	3	8	1.36	0.2	1.56
4	SUCCESS	2	4	1.71	0.2	1.91
12	SUCCESS	2	2	0.98	1.13	2.11
11	SUCCESS	2	1	0.79	1.44	2.22
19	SUCCESS	3	9	1.56	0.71	2.27

0	SUCCESS	2	0	2.16	0.2	2.36
6	SUCCESS	3	6	2.22	0.2	2.42
18	SUCCESS	3	8	1.03	1.56	2.58
13	SUCCESS	2	3	1.36	1.55	2.91
28	SUCCESS	3	8	0.38	2.58	2.97
15	SUCCESS	2	5	1.52	1.55	3.07
29	SUCCESS	3	9	0.96	2.27	3.23
25	SUCCESS	2	5	0.39	3.07	3.46

In this case, we implemented the first come first serve policy for the cloudlets as well as the VMs. The changes for the cloudlets where made in the submitCloudlets() in the DatacenterBroker.java to submit the tasks (cloudlets here) as they're received to the VMs. We have here 100 tasks(base length 300 + random lengths added) with 10 VMs (512 MB RAM each). Refer Appendix D for the Complete Outputs.

CASE 2-

VM Allocation Policy - First Come First Serve

Cloudlets Scheduling Policy -Longest Job First

VM Scheduler -Time Shared

Cloudlet Scheduler -Space Shared

Cloudlets Utilization Model -FULL

No. Of VMs=10

No. Of Cloudlets=100.

Table 4 CASE 2 sample outputs

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
63	SUCCESS	3	9	2.1	0.2	2.3
28	SUCCESS	2	5	2.17	0.2	2.37
53	SUCCESS	3	6	2.21	0.2	2.41
44	SUCCESS	3	7	2.21	0.2	2.41
19	SUCCESS	3	8	2.21	0.2	2.41
89	SUCCESS	2	1	2.28	0.2	2.48
4	SUCCESS	2	2	2.28	0.2	2.48
27	SUCCESS	2	4	2.28	0.2	2.48
38	SUCCESS	2	3	2.28	0.2	2.48
62	SUCCESS	2	0	2.39	0.2	2.59
13	SUCCESS	3	9	1.83	2.3	4.13

42	SUCCESS	3	8	1.91	2.41	4.32
40	SUCCESS	2	5	1.97	2.37	4.34
95	SUCCESS	3	6	2.02	2.41	4.43
61	SUCCESS	3	7	2.02	2.41	4.43
94	SUCCESS	2	4	2.03	2.48	4.51
57	SUCCESS	2	1	2.14	2.48	4.62
25	SUCCESS	2	2	2.14	2.48	4.62
8	SUCCESS	2	3	2.14	2.48	4.62
10	SUCCESS	2	0	2.14	2.59	4.73

In this case, we implemented the longest job first policy for the cloudlets and first come first serve for the VMs. The changes for the cloudlets where made in the submitCloudlets() in the DatacenterBroker.java to submit the tasks while sorting them in the non-increasing lengths according to their lengths (cloudlets here) to the VMs. We have here 100 tasks(base length 300 + random lengths added) with 10 VMs (512 MB RAM each). Refer Appendix D for the Complete Outputs.

CASE 3-

VM Allocation Policy -First Come First Serve

Cloudlets Scheduling Policy -Shortest Job First

VM Scheduler -Time Shared

Cloudlet Scheduler -Space Shared

Cloudlets Utilization Model -FULL

No. Of VMs=10

No. Of Cloudlets=100.

Table 5 CASE 3 sample outputs

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
89	SUCCESS	2	0	0.31	0.2	0.51
59	SUCCESS	3	6	0.37	0.2	0.57
83	SUCCESS	2	1	0.42	0.2	0.62
51	SUCCESS	2	2	0.42	0.2	0.62
11	SUCCESS	2	4	0.42	0.2	0.62
6	SUCCESS	2	3	0.42	0.2	0.62
14	SUCCESS	2	5	0.42	0.2	0.62
49	SUCCESS	3	7	0.48	0.2	0.68
55	SUCCESS	3	8	0.48	0.2	0.68
28	SUCCESS	3	9	0.48	0.2	0.68
93	SUCCESS	2	0	0.46	0.51	0.97

64	SUCCESS	2	1	0.49	0.62	1.11
20	SUCCESS	3	6	0.57	0.57	1.14
72	SUCCESS	2	2	0.6	0.62	1.22
27	SUCCESS	2	4	0.6	0.62	1.22
13	SUCCESS	2	3	0.6	0.62	1.22
98	SUCCESS	2	5	0.6	0.62	1.22
87	SUCCESS	3	7	0.57	0.68	1.25
26	SUCCESS	3	8	0.68	0.68	1.36
88	SUCCESS	3	9	0.68	0.68	1.36

In this case, we implemented the shortest job first policy for the cloudlets and first come first serve for the VMs. The changes for the cloudlets where made in the submitCloudlets() in the DatacenterBroker.java to submit the tasks (cloudlets here) while sorting them to the non-decreasing order of their lengths they're received, to the VMs. We have here 100 tasks(base length 300 + random lengths added) with 10 VMs (512 MB RAM each). Refer Appendix D for the Complete Outputs.

CASE 4-

VM Allocation Policy -Round Robin

Cloudlets Scheduling Policy -Shortest Job First

VM Scheduler -Time Shared

Cloudlet Scheduler -Time Shared

Cloudlets Utilization Model -FULL

No. Of VMs=5

No. Of Cloudlets=50.

Table 6 CASE 4 sample outputs

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
41	SUCCESS	2	0	3	0.1	3.1
46	SUCCESS	2	1	3.11	0.1	3.21
1	SUCCESS	2	2	3.58	0.1	3.68
9	SUCCESS	2	3	3.69	0.1	3.79
25	SUCCESS	2	4	4.07	0.1	4.17
7	SUCCESS	2	0	4.39	0.1	4.49
21	SUCCESS	2	1	4.85	0.1	4.95
36	SUCCESS	2	2	5.62	0.1	5.72
4	SUCCESS	2	3	6.02	0.1	6.12
35	SUCCESS	2	0	6.9	0.1	7
24	SUCCESS	2	4	7.11	0.1	7.21

23	SUCCESS	2	1	7.22	0.1	7.32
26	SUCCESS	2	2	7.33	0.1	7.43
19	SUCCESS	2	0	8.26	0.1	8.36
13	SUCCESS	2	3	8.26	0.1	8.36
6	SUCCESS	2	1	8.49	0.1	8.59
15	SUCCESS	2	4	8.49	0.1	8.59
44	SUCCESS	2	2	9.23	0.1	9.33
47	SUCCESS	2	0	9.34	0.1	9.44
12	SUCCESS	2	3	9.62	0.1	9.72

In this case, we implemented the shortest job first policy for the cloudlets and first come first serve for the VMs. The changes for the TimeShared() for the cloudlets were made in the createVM() function. The changes for the cloudlets where made in the submitCloudlets() in the DatacenterBroker.java to submit the tasks (cloudlets here) as they're received to the VMs. We have here 100 tasks(base length 300 + random lengths added) with 10 VMs (512 MB RAM each). Refer Appendix D for the Complete Outputs.

t) RESULTS

The following graphs show the comparison of average response times and average turn around times for 100 tasks (cloudlets) when 1 and 3 virtual machines were taken for different scheduling algorithms for the cloudlets, namely Shortest Job First(SJF), Longest Job First(LJF), and First Come First Serve(FCFS).

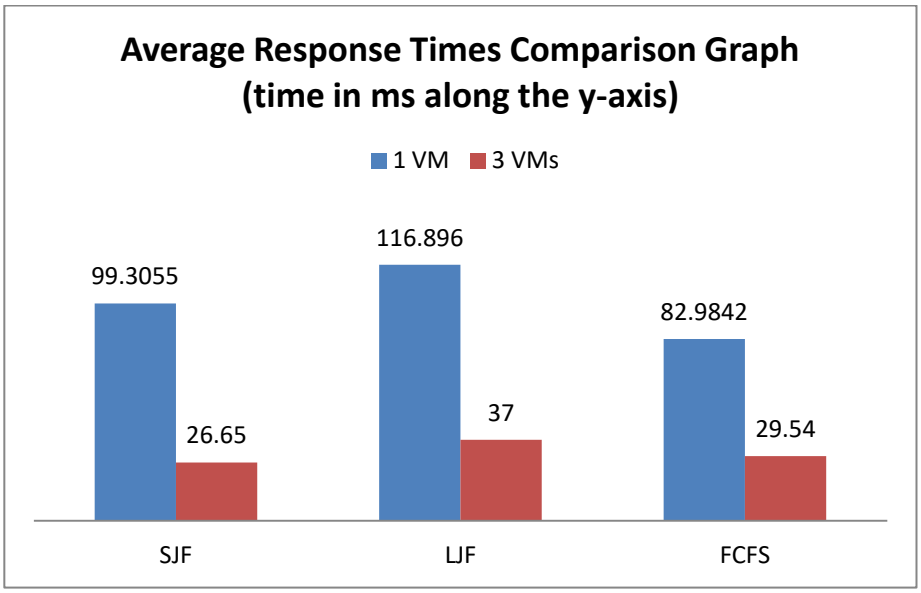

Fig. 25 Average Response Times Comparison Graph

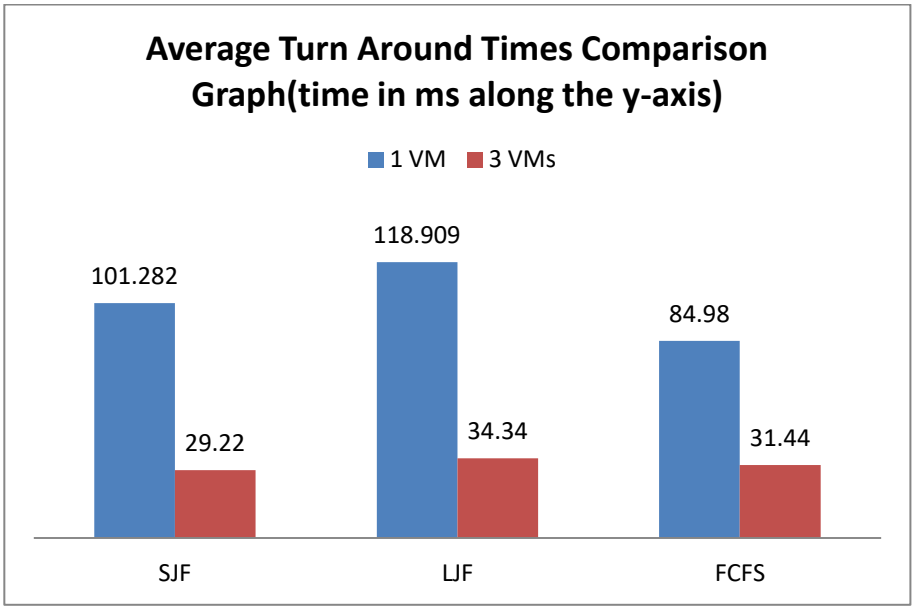

Fig. 26 Average Turn Around Times Comparison Graph

u) OUTPUTS

The following output shows the flow and working of one single task on one single virtual machine. The task is of length 400.

The VM specifications are as follows:-

```
int vmid = 0;
int mips = 1000;
long size = 10000; // image size (MB)
int ram = 512; // vm memory (MB)
long bw = 1000;
int pesNumber = 1; // number of cpus
String vmm = "Xen";
```

The cloudlet specifications are as follows:-

```
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();
```

```
<terminated> CloudSimExample1 [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (18-May-2020, 2:08:36 pm)
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
1.Cloudlet ID: 0 ,Cloudlet Length: 400000
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        400        0.1        400.1
CloudSimExample1 finished!
```

Fig. 27 Scheduling a cloudlet on host

7. FUTURE WORK

Future enhancements will include the case studies of more variations in the virtual machine scheduling and allocation policies. Different policies of the virtual machines, host allocation and task scheduling can be implemented for learning the optimal load balancing in the cloud hence increasing the overall efficiency and productivity.

8. REFERENCES

[1] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C´esar A. F. De Rose and Rajkumar Buyya : CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.

[2] Khadijah Bahwaireth, Lo'ai Tawalbeh, Elhadj Benkhelifa, Yaser Jararweh & Mohammad A. Tawalbeh : Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications.

[3] Utkal Sinha, Mayank Shekhar : Comparison of Various Cloud Simulation tools available in Cloud Computing.

[4] Priyanshu Srivastava, Rizwan Khan : A Review Paper on Cloud Computing.

[5] Santosh Kumar and R. H. Goudar : Cloud Computing – Research Issues, Challenges, Architecture, Platforms and Applications: A Survey.

[6] Mervat Adib Bamiah and Sarfraz Nawaz Brohi : Exploring the Cloud Deployment and Service Delivery Models.

[7] J. M. Suri (DDG I) TEC, B. K. Nath (DIR I) TEC : Security and Privacy in Cloud Computing.

[8] Zhifeng Xiao and Yang Xiao, Senior Member, IEEE : Security and Privacy in Cloud Computing

[9] https://www.superwits.com/library/cloudsim-simulation-framework

[10] cloudsimtutorials.online

[11] Data-flair blogs

[12] aws.amazon.com

APPENDICES

APPENDIX A: Setup of CloudSim

- **Java Development Kit(JDK):** As the Cloudsim simulation toolkit is a class library written in the Java programming language, therefore, the latest version of Java(JDK) should be installed on your machine, which can be downloaded from Oracles Java Folder. For assistance in the installation process, detailed documentation is provided by Oracle itself and you may follow the installation instructions.

- **Eclipse IDE for Java developers:** As per your current installed operating system (Linux/Windows). Before you download to make sure to check if 32-bit or 64-bit version is applicable to your Computer machine. Link for **Eclipse Kepler** version is available at the following link.

- **Download CloudSim source code**: To date, various versions of CloudSim are released the latest version is 5.0, which is based on a container-based engine. Whereas to **keep the setup simple for beginners we will be setting up the most used version i.e. 3.0.3**, which can be directly downloaded by clicking on any of the following: Click for Windows or click for Linux.

- **One external requirement of Cloudsim** i.e. common jar package of math-related functions is to be downloaded from the Apache website.

- Unzip **Eclipse**, **Cloudsim** and **Common Math libraries** to some common folder.

First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe

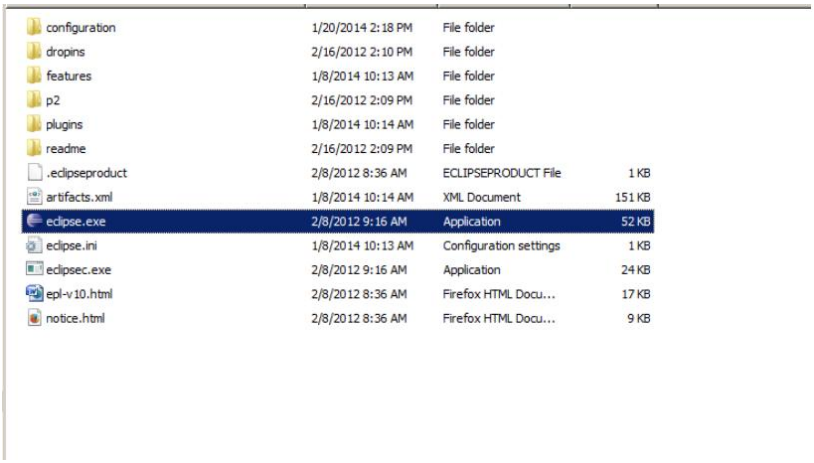

Fig- unzipped folder

- Now within Eclipse window navigate the menu: ***File -> New -> Project,*** to open the new project wizard
- A '***New Project***' wizard should open. There are a number of options displayed and you have to find & select the '***Java Project***' option, once done click ***'Next***''.
- Now a detailed new project window will open, here you will provide the project name and the path of CloudSim project source code, which will be done as follows:

- Project Name: CloudSim.

- Unselect the ***'Use default location'*** option and then click on ***'Browse'*** to open the path where you have unzipped the Cloudsim project and finally click Next to set project settings.

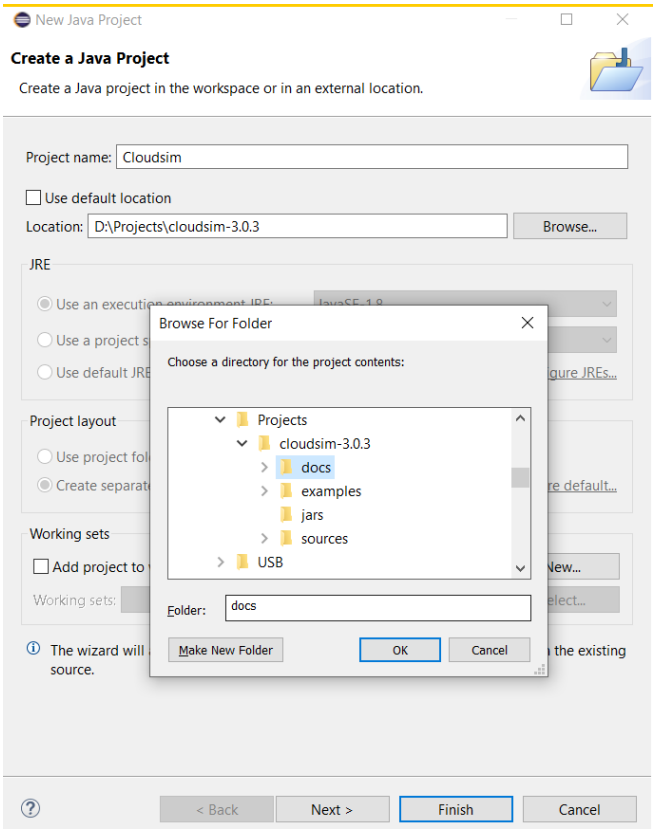

Fig- eclipse new project wizard

- Now open ***'Libraries'*** tab and if you do not find commons-math3-3.x.jar *(here 'x' means the minor version release of the library which could be 2 or greater)* in the list then simply click on **'Add External Jar'** (commons-math3-3.x.jar will be included in the project from this step)

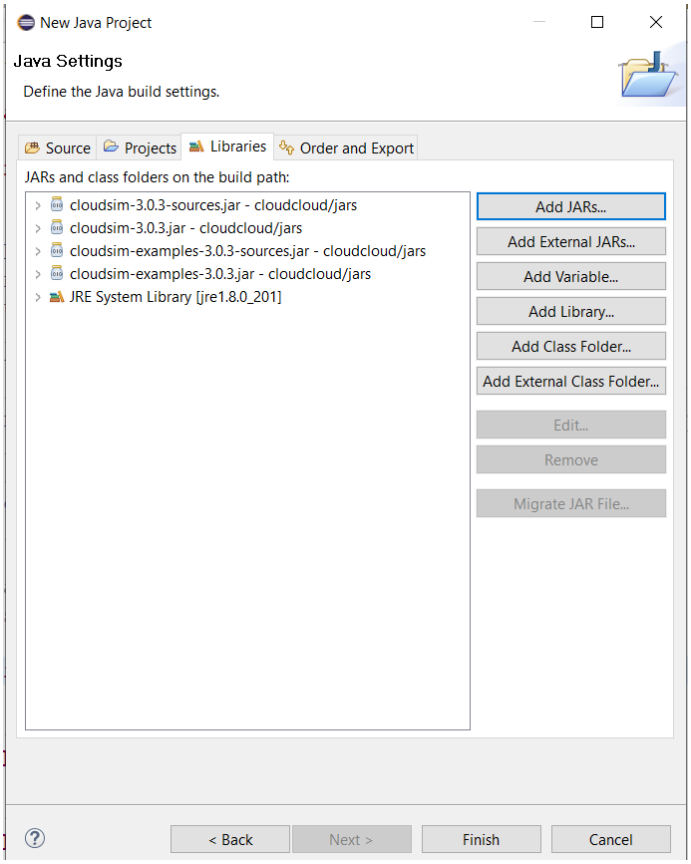

Fig- java settings wizard

- Once you have clicked on '***Add External JAR's***' Open the path where you have unzipped the commons-math binaries and select '***Commons-math3-3.x.jar***' and click on open.

- Ensure external jar that you opened in the previous step is displayed in the list and then click on '***Finish***' (your system may take 2-3 minutes to configure the project)

- Once the project is configured you can open the '***Project Explorer***' and start exploring the Cloudsim project. Also for the first time eclipse automatically start building the workspace for newly configured Cloudsim project, which may take some time depending on the configuration of the computer system.

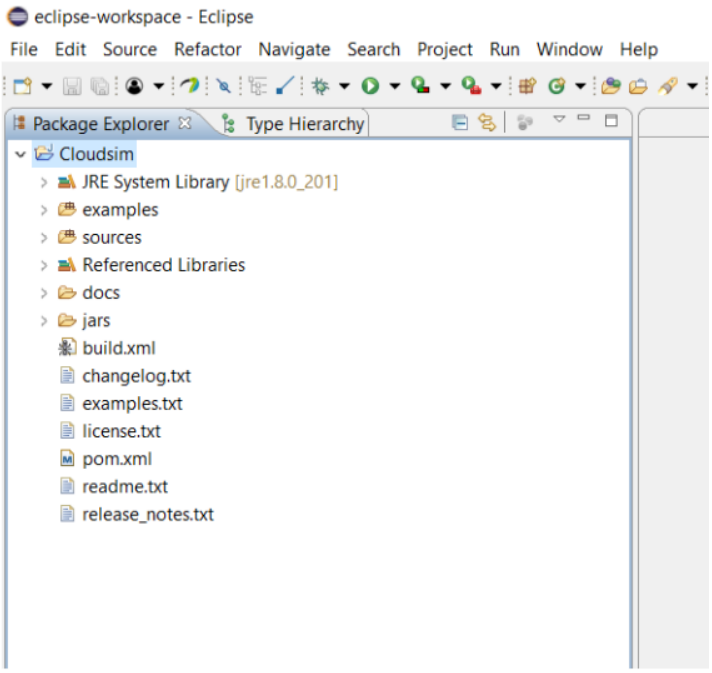

Fig- framework project explorer

- Now just to check you within the '**Project Explorer**', you should navigate to the '**examples**' folder, then expand the package '***org.cloudbus.cloudsim.examples***' and double click to open the '***CloudsimExample1.java***'.

APPENDIX B: Errors in the project.

Most of the times, the type of error that you may encounter during setting up of the cloudsim in eclipse should be stating either:
"Syntax error, parameterized types are only available if source level is 1.5 or greater."

OR

"Syntax error 'for each' statements are only available if source level is 1.5 or greater."

These errors occur due to two reasons:
Case 1: The version of Java installed on your computer system is older than 1.5
Case 2: Your default java environment for the eclipse project not set to the latest java version which is available on your computer system.

For Case 1, you have to download and install the latest java version from Oracle/Java website.

For Case 2, you have to do some modifications in the eclipse project, and for this, you should follow these steps:

1. In the eclipse IDE menu, click on 'Project' and from the dropped menu list click on 'Properties' to open a project properties window.

2. On the project properties window, from the list available on left select 'Java Compiler', this will display the compiler options.

3. Now, click on the 'Enable Project Specific Settings' checkbox, which further enables the options provided just below the checkbox.

4. Now, under JDK compliance heading change the 'Compiler Compliance Level' to the highest number available on the list.

5. Also, make sure "Use default compliance settings" should be in the checked state.

6. Now, click on the "Apply and Close" button. Now a popup alert will be displayed asking for the confirmation to apply the settings. Click 'Yes'.

Now you will notice that the rebuild process for the project will get started and to confirm the error is resolved or not. You may check the error log window or run any example class available in org.cloudbus.cloudsim.examples namespace.

APPENDIX C: CLOUDLETS IN CLOUDSIM

```
/*

First method signature with 8 parameters with simplest cloudlet execution with out
maintaining its history or additional files

*/

public Cloudlet(

                        final int cloudletId,

                        final long cloudletLength,

                        final int pesNumber,

                        final long cloudletFileSize,

                        final long cloudletOutputSize,

                        final UtilizationModel utilizationModelCpu,

                        final UtilizationModel utilizationModelRam,

                        final UtilizationModel utilizationModelBw)



/*

Second method signature with 10 parameters with support for cloudlet execution
history maintenance as well as additionl list of file requirements for execution.

*/

public Cloudlet(

                        final int cloudletId,

                        final long cloudletLength,

                        final int pesNumber,

                        final long cloudletFileSize,
```

```
                            final long cloudletOutputSize,

                            final UtilizationModel utilizationModelCpu,

                            final UtilizationModel utilizationModelRam,

                            final UtilizationModel utilizationModelBw,

                            final boolean record,

                            final List<String> fileList)



/*

Third method signature with 9 parameters with support for cloudlet execution
additionl list of file requirements for execution.

*/

public Cloudlet(

                            final int cloudletId,

                            final long cloudletLength,

                            final int pesNumber,

                            final long cloudletFileSize,

                            final long cloudletOutputSize,

                            final UtilizationModel utilizationModelCpu,

                            final UtilizationModel utilizationModelRam,

                            final UtilizationModel utilizationModelBw,

                            final List<String> fileList)
```

APPENDIX D: CASE STUDIES

CASE 1:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
9	SUCCESS	3	9	0.51	0.2	0.71
7	SUCCESS	3	7	0.84	0.2	1.04
2	SUCCESS	2	2	0.93	0.2	1.13
1	SUCCESS	2	1	1.24	0.2	1.44
17	SUCCESS	3	7	0.41	1.04	1.45
3	SUCCESS	2	3	1.35	0.2	1.55
5	SUCCESS	2	5	1.35	0.2	1.55
8	SUCCESS	3	8	1.36	0.2	1.56
4	SUCCESS	2	4	1.71	0.2	1.91
12	SUCCESS	2	2	0.98	1.13	2.11
11	SUCCESS	2	1	0.79	1.44	2.22
19	SUCCESS	3	9	1.56	0.71	2.27
0	SUCCESS	2	0	2.16	0.2	2.36
6	SUCCESS	3	6	2.22	0.2	2.42
18	SUCCESS	3	8	1.03	1.56	2.58
13	SUCCESS	2	3	1.36	1.55	2.91
28	SUCCESS	3	8	0.38	2.58	2.97
15	SUCCESS	2	5	1.52	1.55	3.07

29	SUCCESS	3	9	0.96	2.27	3.23
25	SUCCESS	2	5	0.39	3.07	3.46
22	SUCCESS	2	2	1.46	2.11	3.57
27	SUCCESS	3	7	2.17	1.45	3.61
23	SUCCESS	2	3	0.77	2.91	3.68
16	SUCCESS	3	6	1.54	2.42	3.96
10	SUCCESS	2	0	1.65	2.36	4
32	SUCCESS	2	2	0.54	3.57	4.11
21	SUCCESS	2	1	2	2.22	4.22
14	SUCCESS	2	4	2.31	1.91	4.22
20	SUCCESS	2	0	0.33	4	4.33
39	SUCCESS	3	9	1.24	3.23	4.47
31	SUCCESS	2	1	0.48	4.22	4.7
38	SUCCESS	3	8	1.84	2.97	4.8
26	SUCCESS	3	6	0.95	3.96	4.91
24	SUCCESS	2	4	0.83	4.22	5.06
49	SUCCESS	3	9	0.69	4.47	5.16
30	SUCCESS	2	0	1.05	4.33	5.39
33	SUCCESS	2	3	1.84	3.68	5.52
37	SUCCESS	3	7	1.96	3.61	5.57
35	SUCCESS	2	5	2.17	3.46	5.63
48	SUCCESS	3	8	0.88	4.8	5.68

41	SUCCESS	2	1	1.19	4.7	5.89
42	SUCCESS	2	2	1.89	4.11	6
58	SUCCESS	3	8	0.38	5.68	6.07
34	SUCCESS	2	4	1.52	5.06	6.58
43	SUCCESS	2	3	1.62	5.52	7.14
36	SUCCESS	3	6	2.29	4.91	7.2
40	SUCCESS	2	0	1.86	5.39	7.25
45	SUCCESS	2	5	1.62	5.63	7.25
59	SUCCESS	3	9	2.22	5.16	7.38
51	SUCCESS	2	1	1.5	5.89	7.39
52	SUCCESS	2	2	1.61	6	7.61
47	SUCCESS	3	7	2.05	5.57	7.63
50	SUCCESS	2	0	0.49	7.25	7.74
68	SUCCESS	3	8	1.71	6.07	7.77
55	SUCCESS	2	5	0.59	7.25	7.85
62	SUCCESS	2	2	0.38	7.61	8
57	SUCCESS	3	7	0.4	7.63	8.03
46	SUCCESS	3	6	0.94	7.2	8.14
78	SUCCESS	3	8	0.36	7.77	8.14
88	SUCCESS	3	8	0.4	8.14	8.54
44	SUCCESS	2	4	1.98	6.58	8.57
69	SUCCESS	3	9	1.27	7.38	8.65

56	SUCCESS	3	6	0.71	8.14	8.85
65	SUCCESS	2	5	1.1	7.85	8.95
61	SUCCESS	2	1	1.67	7.39	9.06
53	SUCCESS	2	3	2.02	7.14	9.17
72	SUCCESS	2	2	1.45	8	9.44
54	SUCCESS	2	4	0.99	8.57	9.55
63	SUCCESS	2	3	0.39	9.17	9.55
66	SUCCESS	3	6	0.84	8.85	9.69
60	SUCCESS	2	0	2.08	7.74	9.81
98	SUCCESS	3	8	1.38	8.54	9.92
73	SUCCESS	2	3	0.47	9.55	10.03
67	SUCCESS	3	7	2.08	8.03	10.11
75	SUCCESS	2	5	1.19	8.95	10.14
85	SUCCESS	2	5	0.31	10.14	10.45
71	SUCCESS	2	1	1.5	9.06	10.56
83	SUCCESS	2	3	0.53	10.03	10.56
70	SUCCESS	2	0	0.85	9.81	10.67
64	SUCCESS	2	4	1.22	9.55	10.78
79	SUCCESS	3	9	2.17	8.65	10.82
95	SUCCESS	2	5	0.47	10.45	10.92
77	SUCCESS	3	7	1.09	10.11	11.2
76	SUCCESS	3	6	1.74	9.69	11.43

82	SUCCESS	2	2	2.09	9.44	11.53
87	SUCCESS	3	7	0.66	11.2	11.86
80	SUCCESS	2	0	1.21	10.67	11.88
81	SUCCESS	2	1	1.43	10.56	11.99
93	SUCCESS	2	3	1.43	10.56	11.99
89	SUCCESS	3	9	1.72	10.82	12.54
92	SUCCESS	2	2	1.27	11.53	12.8
74	SUCCESS	2	4	2.13	10.78	12.91
91	SUCCESS	2	1	1.09	11.99	13.07
86	SUCCESS	3	6	1.93	11.43	13.35
84	SUCCESS	2	4	0.57	12.91	13.48
97	SUCCESS	3	7	1.75	11.86	13.61
90	SUCCESS	2	0	2.01	11.88	13.89
99	SUCCESS	3	9	1.9	12.54	14.44
96	SUCCESS	3	6	1.33	13.35	14.68
94	SUCCESS	2	4	2.22	13.48	15.7

CASE 2:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
63	SUCCESS	3	9	2.1	0.2	2.3
28	SUCCESS	2	5	2.17	0.2	2.37

53	SUCCESS	3	6	2.21	0.2	2.41
44	SUCCESS	3	7	2.21	0.2	2.41
19	SUCCESS	3	8	2.21	0.2	2.41
89	SUCCESS	2	1	2.28	0.2	2.48
4	SUCCESS	2	2	2.28	0.2	2.48
27	SUCCESS	2	4	2.28	0.2	2.48
38	SUCCESS	2	3	2.28	0.2	2.48
62	SUCCESS	2	0	2.39	0.2	2.59
13	SUCCESS	3	9	1.83	2.3	4.13
42	SUCCESS	3	8	1.91	2.41	4.32
40	SUCCESS	2	5	1.97	2.37	4.34
95	SUCCESS	3	6	2.02	2.41	4.43
61	SUCCESS	3	7	2.02	2.41	4.43
94	SUCCESS	2	4	2.03	2.48	4.51
57	SUCCESS	2	1	2.14	2.48	4.62
25	SUCCESS	2	2	2.14	2.48	4.62
8	SUCCESS	2	3	2.14	2.48	4.62
10	SUCCESS	2	0	2.14	2.59	4.73
92	SUCCESS	3	9	1.68	4.13	5.81
66	SUCCESS	3	8	1.69	4.32	6.02
68	SUCCESS	2	5	1.75	4.34	6.09
39	SUCCESS	3	7	1.73	4.43	6.16

41	SUCCESS	2	4	1.76	4.51	6.27
70	SUCCESS	3	6	1.84	4.43	6.27
52	SUCCESS	2	3	1.78	4.62	6.4
51	SUCCESS	2	1	1.89	4.62	6.51
22	SUCCESS	2	2	1.89	4.62	6.51
36	SUCCESS	2	0	1.89	4.73	6.62
32	SUCCESS	3	9	1.52	5.81	7.33
69	SUCCESS	3	8	1.54	6.02	7.55
31	SUCCESS	2	5	1.56	6.09	7.65
58	SUCCESS	3	7	1.54	6.16	7.7
47	SUCCESS	3	6	1.54	6.27	7.81
71	SUCCESS	2	4	1.56	6.27	7.83
26	SUCCESS	2	3	1.56	6.4	7.96
65	SUCCESS	2	2	1.63	6.51	8.14
77	SUCCESS	2	1	1.74	6.51	8.25
33	SUCCESS	2	0	1.74	6.62	8.36
93	SUCCESS	3	9	1.28	7.33	8.61
11	SUCCESS	3	8	1.3	7.55	8.85
7	SUCCESS	2	5	1.36	7.65	9.02
67	SUCCESS	3	7	1.34	7.7	9.04
60	SUCCESS	3	6	1.35	7.81	9.16
1	SUCCESS	2	4	1.41	7.83	9.24

24	SUCCESS	2	3	1.41	7.96	9.37
5	SUCCESS	2	2	1.42	8.14	9.57
72	SUCCESS	3	9	1.13	8.61	9.74
15	SUCCESS	2	1	1.49	8.25	9.74
97	SUCCESS	2	0	1.5	8.36	9.87
80	SUCCESS	3	8	1.16	8.85	10.01
9	SUCCESS	2	5	1.2	9.02	10.21
84	SUCCESS	3	7	1.17	9.04	10.21
29	SUCCESS	3	6	1.17	9.16	10.34
14	SUCCESS	2	4	1.2	9.24	10.44
74	SUCCESS	2	3	1.2	9.37	10.58
86	SUCCESS	3	9	0.92	9.74	10.65
96	SUCCESS	2	2	1.27	9.57	10.84
21	SUCCESS	3	8	0.93	10.01	10.94
81	SUCCESS	2	1	1.27	9.74	11.02
54	SUCCESS	2	0	1.28	9.87	11.14
82	SUCCESS	3	7	0.99	10.21	11.21
49	SUCCESS	2	5	1.06	10.21	11.27
99	SUCCESS	3	6	1	10.34	11.34
30	SUCCESS	3	9	0.8	10.65	11.45
75	SUCCESS	2	4	1.07	10.44	11.51
34	SUCCESS	2	3	1.07	10.58	11.65

78	SUCCESS	3	8	0.73	10.94	11.68
48	SUCCESS	2	2	1.09	10.84	11.93
3	SUCCESS	3	9	0.48	11.45	11.93
0	SUCCESS	3	7	0.84	11.21	12.04
56	SUCCESS	2	5	0.78	11.27	12.05
37	SUCCESS	3	6	0.81	11.34	12.15
59	SUCCESS	2	1	1.14	11.02	12.16
16	SUCCESS	3	8	0.59	11.68	12.26
87	SUCCESS	3	9	0.33	11.93	12.26
79	SUCCESS	2	0	1.12	11.14	12.27
18	SUCCESS	2	4	0.87	11.51	12.38
35	SUCCESS	2	3	0.84	11.65	12.49
64	SUCCESS	3	8	0.31	12.26	12.58
76	SUCCESS	2	5	0.59	12.05	12.64
83	SUCCESS	3	7	0.64	12.04	12.69
88	SUCCESS	2	2	0.84	11.93	12.77
55	SUCCESS	3	6	0.64	12.15	12.8
73	SUCCESS	2	4	0.59	12.38	12.97
12	SUCCESS	3	7	0.32	12.69	13
46	SUCCESS	2	1	0.92	12.16	13.08
98	SUCCESS	2	5	0.44	12.64	13.08
6	SUCCESS	3	6	0.33	12.8	13.13

20	SUCCESS	2	0	0.92	12.27	13.19
45	SUCCESS	2	3	0.7	12.49	13.19
91	SUCCESS	2	4	0.39	12.97	13.36
23	SUCCESS	2	2	0.71	12.77	13.47
17	SUCCESS	2	3	0.39	13.19	13.58
90	SUCCESS	2	1	0.65	13.08	13.73
2	SUCCESS	2	0	0.68	13.19	13.87
43	SUCCESS	2	2	0.5	13.47	13.98
85	SUCCESS	2	1	0.43	13.73	14.16
50	SUCCESS	2	0	0.45	13.87	14.32

CASE 3:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
89	SUCCESS	2	0	0.31	0.2	0.51
59	SUCCESS	3	6	0.37	0.2	0.57
83	SUCCESS	2	1	0.42	0.2	0.62
51	SUCCESS	2	2	0.42	0.2	0.62
11	SUCCESS	2	4	0.42	0.2	0.62
6	SUCCESS	2	3	0.42	0.2	0.62
14	SUCCESS	2	5	0.42	0.2	0.62
49	SUCCESS	3	7	0.48	0.2	0.68
55	SUCCESS	3	8	0.48	0.2	0.68
28	SUCCESS	3	9	0.48	0.2	0.68

93	SUCCESS	2	0	0.46	0.51	0.97
64	SUCCESS	2	1	0.49	0.62	1.11
20	SUCCESS	3	6	0.57	0.57	1.14
72	SUCCESS	2	2	0.6	0.62	1.22
27	SUCCESS	2	4	0.6	0.62	1.22
13	SUCCESS	2	3	0.6	0.62	1.22
98	SUCCESS	2	5	0.6	0.62	1.22
87	SUCCESS	3	7	0.57	0.68	1.25
26	SUCCESS	3	8	0.68	0.68	1.36
88	SUCCESS	3	9	0.68	0.68	1.36
92	SUCCESS	2	0	0.68	0.97	1.65
12	SUCCESS	2	1	0.7	1.11	1.81
39	SUCCESS	3	6	0.77	1.14	1.91
67	SUCCESS	2	2	0.71	1.22	1.93
81	SUCCESS	2	4	0.82	1.22	2.04
42	SUCCESS	2	3	0.82	1.22	2.04
7	SUCCESS	2	5	0.82	1.22	2.04
70	SUCCESS	3	7	0.83	1.25	2.08
22	SUCCESS	3	8	0.83	1.36	2.19
79	SUCCESS	3	9	0.83	1.36	2.19
85	SUCCESS	2	0	0.87	1.65	2.52
40	SUCCESS	2	1	0.89	1.81	2.7

47	SUCCESS	2	2	0.89	1.93	2.82
56	SUCCESS	3	6	1.1	1.91	3.02
9	SUCCESS	2	3	1.04	2.04	3.08
30	SUCCESS	3	7	1.11	2.08	3.19
97	SUCCESS	2	4	1.15	2.04	3.19
10	SUCCESS	2	5	1.15	2.04	3.19
66	SUCCESS	3	8	1.14	2.19	3.33
37	SUCCESS	3	9	1.25	2.19	3.44
24	SUCCESS	2	0	1.17	2.52	3.7
61	SUCCESS	2	1	1.19	2.7	3.88
95	SUCCESS	2	2	1.21	2.82	4.03
45	SUCCESS	3	6	1.31	3.02	4.33
31	SUCCESS	2	3	1.29	3.08	4.38
48	SUCCESS	2	4	1.3	3.19	4.5
73	SUCCESS	3	7	1.32	3.19	4.51
41	SUCCESS	2	5	1.41	3.19	4.61
23	SUCCESS	3	8	1.35	3.33	4.67
62	SUCCESS	3	9	1.39	3.44	4.83
65	SUCCESS	2	0	1.42	3.7	5.12
57	SUCCESS	2	1	1.43	3.88	5.31
17	SUCCESS	2	2	1.43	4.03	5.46
21	SUCCESS	2	3	1.45	4.38	5.83

76	SUCCESS	3	6	1.54	4.33	5.87
18	SUCCESS	2	4	1.46	4.5	5.95
44	SUCCESS	3	7	1.57	4.51	6.08
63	SUCCESS	2	5	1.52	4.61	6.13
90	SUCCESS	3	8	1.6	4.67	6.27
5	SUCCESS	3	9	1.6	4.83	6.43
0	SUCCESS	2	0	1.64	5.12	6.76
46	SUCCESS	2	1	1.67	5.31	6.98
54	SUCCESS	2	2	1.67	5.46	7.13
69	SUCCESS	2	3	1.67	5.83	7.5
25	SUCCESS	2	4	1.68	5.95	7.64
4	SUCCESS	3	6	1.77	5.87	7.64
78	SUCCESS	2	5	1.71	6.13	7.84
35	SUCCESS	3	7	1.77	6.08	7.85
60	SUCCESS	3	8	1.82	6.27	8.1
94	SUCCESS	3	9	1.84	6.43	8.27
77	SUCCESS	2	0	1.86	6.76	8.63
8	SUCCESS	2	1	1.88	6.98	8.85
43	SUCCESS	2	2	1.88	7.13	9
84	SUCCESS	2	3	1.88	7.5	9.38
99	SUCCESS	2	4	1.89	7.64	9.53
58	SUCCESS	3	6	1.92	7.64	9.56

86	SUCCESS	2	5	1.89	7.84	9.73
3	SUCCESS	3	7	1.94	7.85	9.79
33	SUCCESS	3	8	1.96	8.1	10.06
34	SUCCESS	3	9	2.01	8.27	10.27
32	SUCCESS	2	0	2.01	8.63	10.64
91	SUCCESS	2	1	2.01	8.85	10.87
52	SUCCESS	2	2	2.02	9	11.03
75	SUCCESS	2	3	2.06	9.38	11.44
50	SUCCESS	2	4	2.06	9.53	11.59
15	SUCCESS	3	6	2.09	9.56	11.65
1	SUCCESS	2	5	2.08	9.73	11.82
68	SUCCESS	3	7	2.11	9.79	11.9
29	SUCCESS	3	8	2.13	10.06	12.19
96	SUCCESS	3	9	2.13	10.27	12.41
36	SUCCESS	2	0	2.14	10.64	12.78
2	SUCCESS	2	1	2.15	10.87	13.01
53	SUCCESS	2	2	2.17	11.03	13.2
19	SUCCESS	2	3	2.19	11.44	13.63
71	SUCCESS	2	4	2.2	11.59	13.79
82	SUCCESS	3	6	2.22	11.65	13.87
80	SUCCESS	2	5	2.2	11.82	14.02
38	SUCCESS	3	7	2.23	11.9	14.13

16	SUCCESS	3	8	2.24	12.19	14.43
74	SUCCESS	3	9	2.25	12.41	14.66

CASE 4:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
41	SUCCESS	2	0	3	0.1	3.1
46	SUCCESS	2	1	3.11	0.1	3.21
1	SUCCESS	2	2	3.58	0.1	3.68
9	SUCCESS	2	3	3.69	0.1	3.79
25	SUCCESS	2	4	4.07	0.1	4.17
7	SUCCESS	2	0	4.39	0.1	4.49
21	SUCCESS	2	1	4.85	0.1	4.95
36	SUCCESS	2	2	5.62	0.1	5.72
4	SUCCESS	2	3	6.02	0.1	6.12
35	SUCCESS	2	0	6.9	0.1	7
24	SUCCESS	2	4	7.11	0.1	7.21
23	SUCCESS	2	1	7.22	0.1	7.32
26	SUCCESS	2	2	7.33	0.1	7.43
19	SUCCESS	2	0	8.26	0.1	8.36
13	SUCCESS	2	3	8.26	0.1	8.36
6	SUCCESS	2	1	8.49	0.1	8.59
15	SUCCESS	2	4	8.49	0.1	8.59

44	SUCCESS	2	2	9.23	0.1	9.33
47	SUCCESS	2	0	9.34	0.1	9.44
12	SUCCESS	2	3	9.62	0.1	9.72
32	SUCCESS	2	1	9.73	0.1	9.83
30	SUCCESS	2	4	9.87	0.1	9.97
14	SUCCESS	2	2	10.11	0.1	10.21
0	SUCCESS	2	0	10.25	0.1	10.35
48	SUCCESS	2	3	10.36	0.1	10.46
22	SUCCESS	2	1	10.47	0.1	10.57
17	SUCCESS	2	0	10.66	0.1	10.76
3	SUCCESS	2	2	10.77	0.1	10.87
31	SUCCESS	2	4	10.77	0.1	10.87
11	SUCCESS	2	3	11.04	0.1	11.14
5	SUCCESS	2	1	11.15	0.1	11.25
27	SUCCESS	2	0	11.26	0.1	11.36
8	SUCCESS	2	4	11.43	0.1	11.53
2	SUCCESS	2	1	11.54	0.1	11.64
37	SUCCESS	2	2	11.65	0.1	11.75
43	SUCCESS	2	0	11.82	0.1	11.92
40	SUCCESS	2	2	11.93	0.1	12.03
45	SUCCESS	2	3	11.93	0.1	12.03
18	SUCCESS	2	0	12.04	0.1	12.14

16	SUCCESS	2	1	12.15	0.1	12.25
39	SUCCESS	2	4	12.26	0.1	12.36
10	SUCCESS	2	1	12.37	0.1	12.47
42	SUCCESS	2	3	12.48	0.1	12.58
34	SUCCESS	2	2	12.62	0.1	12.72
29	SUCCESS	2	2	12.84	0.1	12.94
28	SUCCESS	2	3	13.05	0.1	13.15
38	SUCCESS	2	4	13.16	0.1	13.26
33	SUCCESS	2	3	13.27	0.1	13.37
20	SUCCESS	2	4	13.39	0.1	13.49
49	SUCCESS	2	4	13.59	0.1	13.69