Configuration Manual

MSc Research Project
MSc in Cloud Computing
(MSCCLOUD1_JAN21)

Tushar Sonawane
X20195753

School of Computing
National College of Ireland

Supervisor: Punit Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Tushar Pramod Sonawane.

Student ID: X20195753.

Programme: MSc Cloud Computing.

Module: MSc Research Project.

Supervisor: Punit Gupta.

Submission Due Date: 1 / 2 / 2023

Project Title: Energy Efficiency & Consumption in Data Centre by Dynamic Resource Allocation Technique for Green Cloud Computing.

WordCount:1070 **Page Count:** 11.

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……………………………………………………………………………………………………

Date: 1 / 2 / 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Tushar Pramod Sonawane
X20195753

1 Introduction

This configuration manual discusses the step-by-step procedure used in this project, from environment setup through implementation and assessment, to determine if the performance of a transformer-based model is enhanced by applying an aspect-based sentiment analysis technique to it. This configuration manual contains details regarding the programming language used, the system's configuration, and the required libraries. The study's findings, many experiments, and the assessment metrics that were applied to each are addressed.

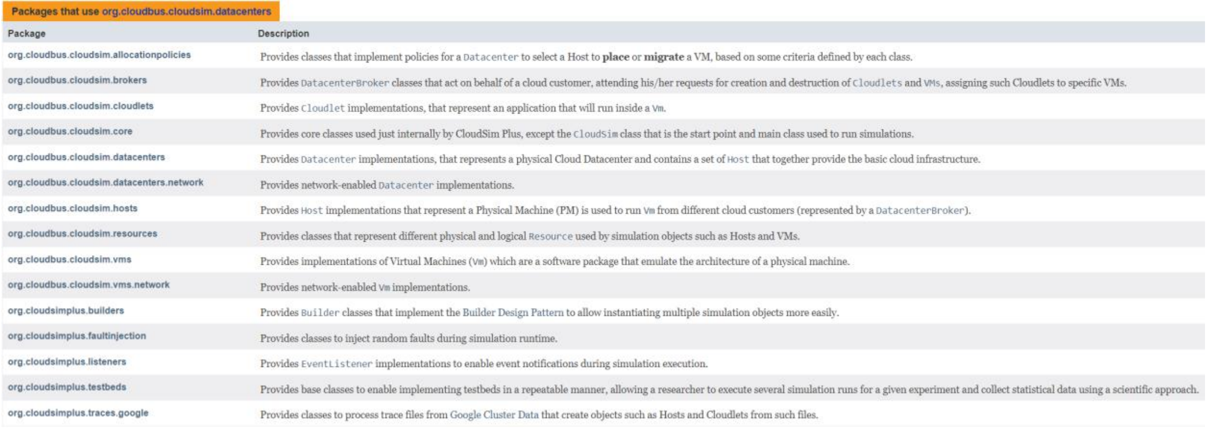
2 Environment Setup

2.1 System Specification.

The CloudSim Simulator was utilized to carry out this research. The open-source version, CloudSim 3.0.3, is accessible on GitHub. By assigning the desired number of hosts and virtual machines to the CloudSim simulator's Random Constant files, we can create our own virtual data centre. Since the entire code was written in Java, we utilized the Eclipse IDE to run it. The code execution time on the Eclipse IDE is reduced because of faster GPU hardware.

2.2 Simulation Environment & Technical Specification.

Packages



Packages that use org.cloudbus.cloudsim.datacenters

Package	Description
org.cloudbus.cloudsim.allocationpolicies	Provides classes that implement policies for a Datacenter to select a Host to **place** or **migrate** a VM, based on some criteria defined by each class.
org.cloudbus.cloudsim.brokers	Provides DatacenterBroker classes that act on behalf of a cloud customer, attending his/her requests for creation and destruction of Cloudlets and VMs, assigning such Cloudlets to specific VMs.
org.cloudbus.cloudsim.cloudlets	Provides Cloudlet implementations, that represent an application that will run inside a Vm.
org.cloudbus.cloudsim.core	Provides core classes used just internally by CloudSim Plus, except the CloudSim class that is the start point and main class used to run simulations.
org.cloudbus.cloudsim.datacenters	Provides Datacenter implementations, that represents a physical Cloud Datacenter and contains a set of Host that together provide the basic cloud infrastructure.
org.cloudbus.cloudsim.datacenters.network	Provides network-enabled Datacenter implementations.
org.cloudbus.cloudsim.hosts	Provides Host implementations that represent a Physical Machine (PM) is used to run Vm from different cloud customers (represented by a DatacenterBroker).
org.cloudbus.cloudsim.resources	Provides classes that represent different physical and logical Resource used by simulation objects such as Hosts and VMs.
org.cloudbus.cloudsim.vms	Provides implementations of Virtual Machines (Vm) which are a software package that emulate the architecture of a physical machine.
org.cloudbus.cloudsim.vms.network	Provides network-enabled Vm implementations.
org.cloudsimplus.builders	Provides Builder classes that implement the Builder Design Pattern to allow instantiating multiple simulation objects more easily.
org.cloudsimplus.faultinjection	Provides classes to inject random faults during simulation runtime.
org.cloudsimplus.listeners	Provides EventListener implementations to enable event notifications during simulation execution.
org.cloudsimplus.testbeds	Provides base classes to enable implementing testbeds in a repeatable manner, allowing a researcher to execute several simulation runs for a given experiment and collect statistical data using a scientific approach.
org.cloudsimplus.traces.google	Provides classes to process trace files from Google Cluster Data that create objects such as Hosts and Cloudlets from such files.

Figure 1: Packages used in cloudsim.datacenters

- Allocation Policies
- Datacenter Broker
- Cloudlets
- Hosts
- VMS (Virtual Machine)

- Resources
- Elitism

2.3 Data Source.

To determine the data center's energy consumption, the data in this section refers to the number of VMs and Hosts. It is simple to create a virtual data center environment using cloudsim, where we may distribute resources, virtual hosts, and virtual machines. In CloudSim, the file RandomConstant contains all the parameters that allow us to assign the number of hosts and virtual machines.

2.4 Prerequisites.

- Working or fundamental understanding of Java programming is necessary for users.
- Integrated Development Environments (IDEs) for Java, such as Eclipse, should be able to be used.
- It is necessary to understand the Java code from the simulation program CloudSim.

2.5 Software Tools Used

- Eclipse IDE
- CloudSim
- Microsoft Excel – Used to visualize and study the results.
- JDK 17 – Contains Java Libraries to run Java Programs.

2.6 Hardware Tools Used

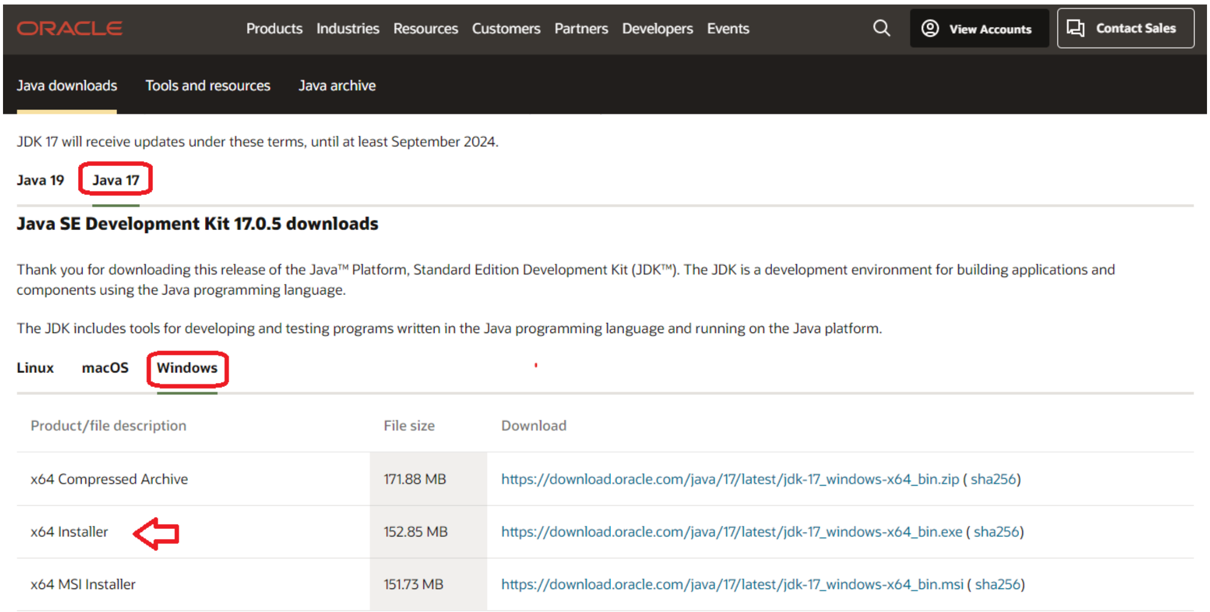
- Operating System: Windows / Linux / Mac.
- RAM: 8 Gb or more.
- Processor: Minimum Core i5 or more.

3 Software Installation

Step-by-step implementation process.

3.1 Java Development Kit Installation

Download JDK 17 from the below link.
https://www.oracle.com/ie/java/technologies/downloads/#jdk17-windows



JDK 17 will receive updates under these terms, until at least September 2024.

Java SE Development Kit 17.0.5 downloads

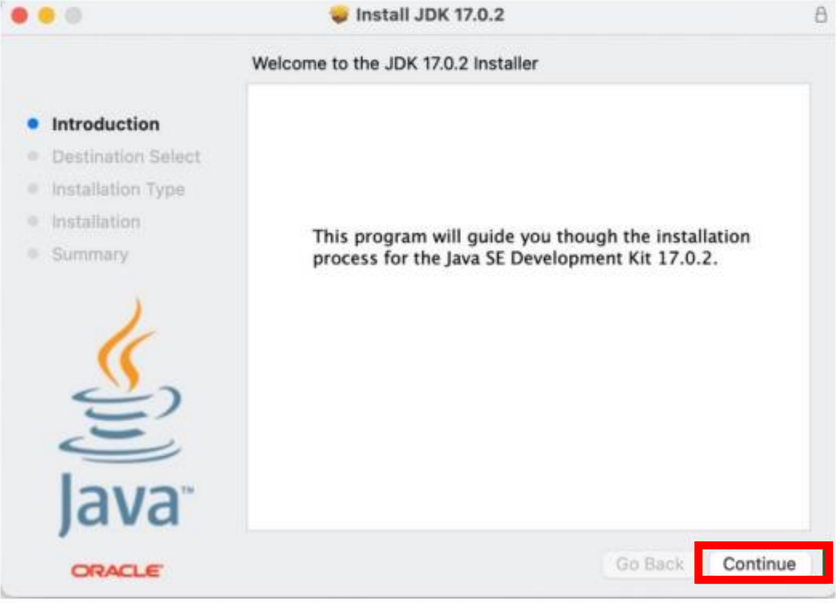
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

Product/file description	File size	Download
x64 Compressed Archive	171.88 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	152.85 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	151.73 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

Figure 2: JDK official website

- **Install Java JDK 17 on your system**



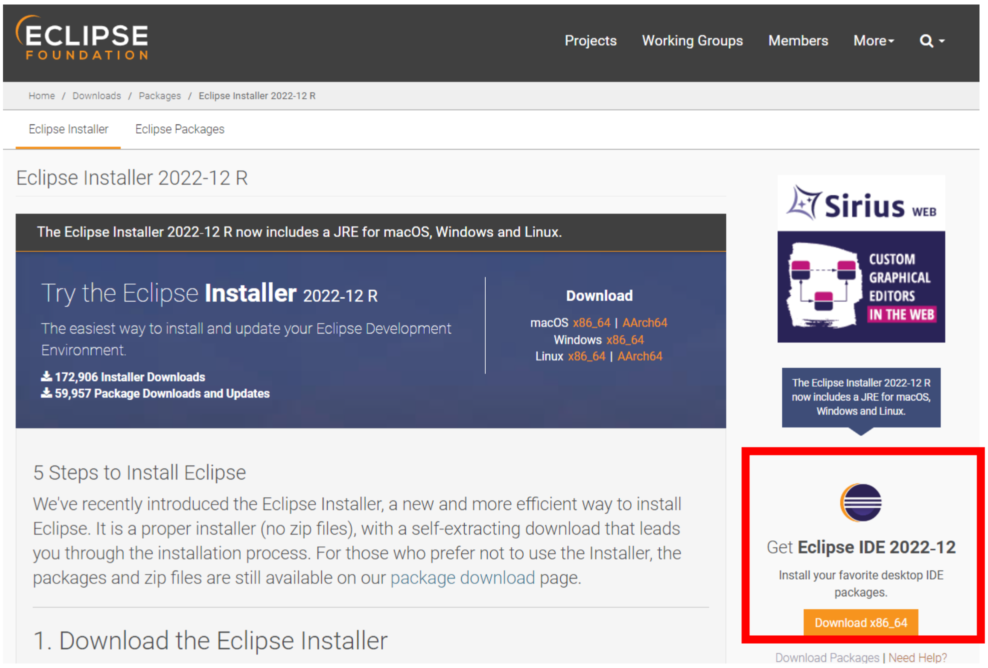
Welcome to the JDK 17.0.2 Installer

This program will guide you though the installation process for the Java SE Development Kit 17.0.2.

Figure 3: JDK Installation

3.2 Eclipse (IDE) Installtion

- https://www.eclipse.org/downloads/packages/installer



Eclipse Installer 2022-12 R

The Eclipse Installer 2022-12 R now includes a JRE for macOS, Windows and Linux.

Try the Eclipse **Installer** 2022-12 R

The easiest way to install and update your Eclipse Development Environment.

172,906 Installer Downloads
59,957 Package Downloads and Updates

The Eclipse Installer 2022-12 R now includes a JRE for macOS, Windows and Linux.

Get **Eclipse IDE 2022-12**

Install your favorite desktop IDE packages.

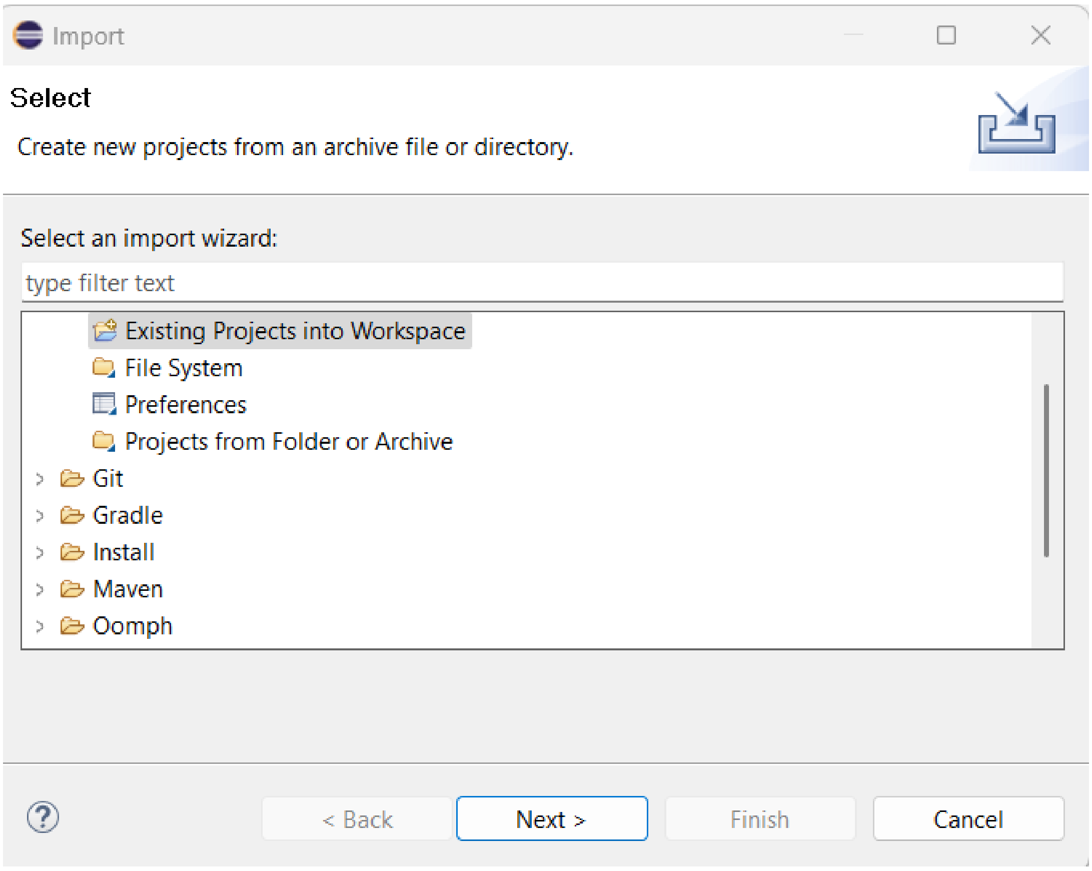
5 Steps to Install Eclipse

We've recently introduced the Eclipse Installer, a new and more efficient way to install Eclipse. It is a proper installer (no zip files), with a self-extracting download that leads you through the installation process. For those who prefer not to use the Installer, the packages and zip files are still available on our package download page.

1. Download the Eclipse Installer

Figure 4: Eclipse Downloader

- **Open Eclipse and create/import a new project under the home directory**



Select

Create new projects from an archive file or directory.

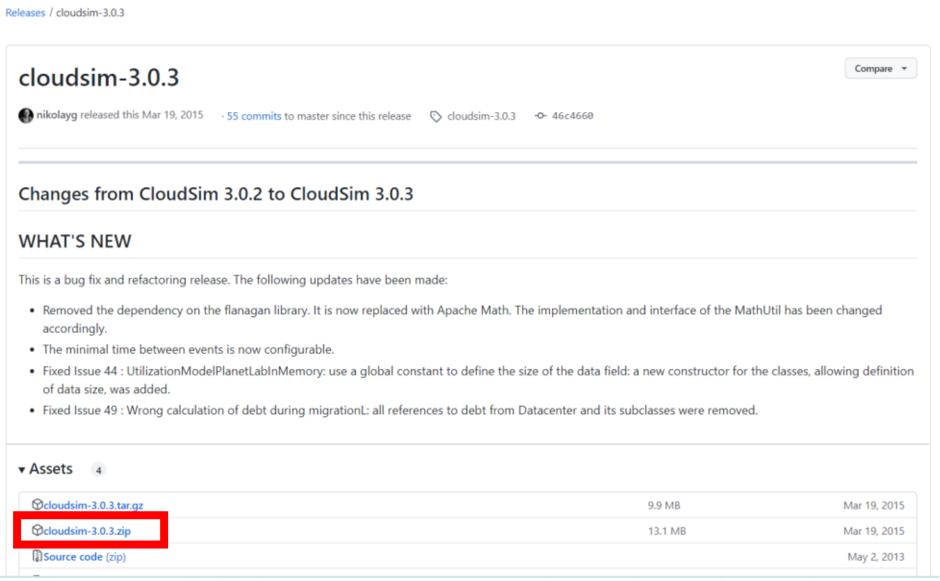
Select an import wizard:

type filter text

Figure 5: Import CloudSim

3.3 Download & Install CloudSim

We are using CloudSim 3.0.3 from GitHub https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3



Releases / cloudsim-3.0.3

cloudsim-3.0.3

nikolayg released this Mar 19, 2015 · 55 commits to master since this release · cloudsim-3.0.3 · 46c4660

Changes from CloudSim 3.0.2 to CloudSim 3.0.3

WHAT'S NEW

This is a bug fix and refactoring release. The following updates have been made:

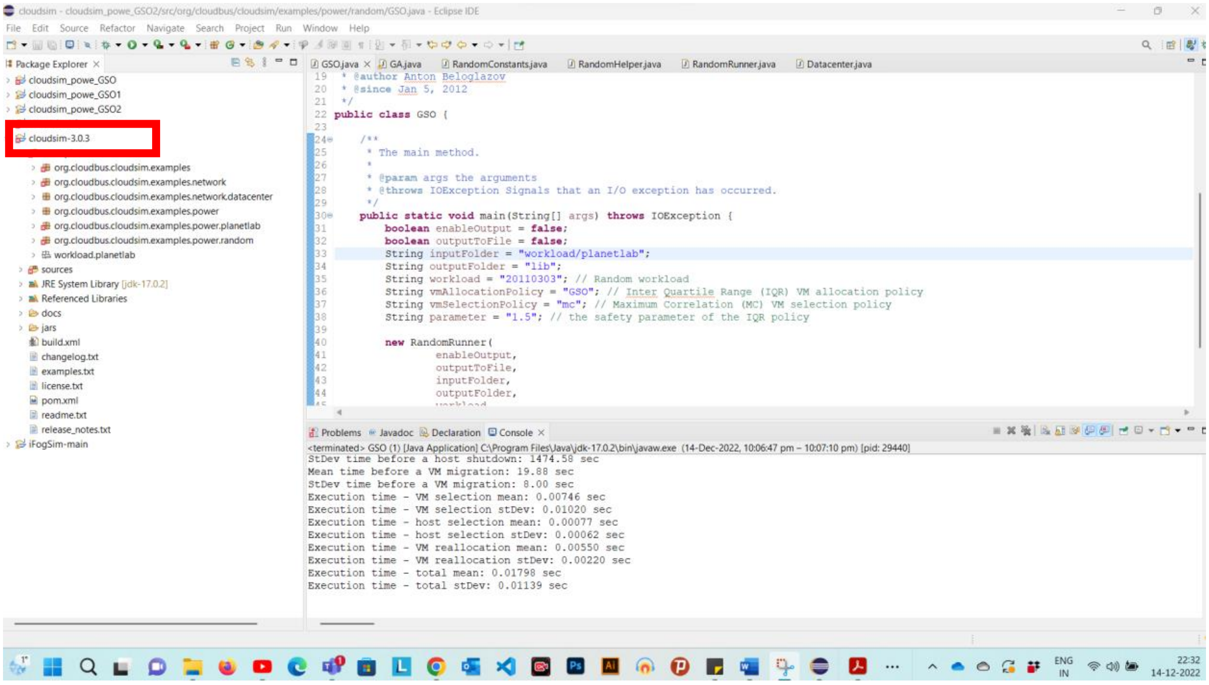
- Removed the dependency on the flanagan library. It is now replaced with Apache Math. The implementation and interface of the MathUtil has been changed accordingly.
- The minimal time between events is now configurable.
- Fixed Issue 44 : UtilizationModelPlanetLabInMemory: use a global constant to define the size of the data field: a new constructor for the classes, allowing definition of data size, was added.
- Fixed Issue 49 : Wrong calculation of debt during migrationL: all references to debt from Datacenter and its subclasses were removed.

Assets 4

cloudsim-3.0.3.tar.gz	9.9 MB	Mar 19, 2015
cloudsim-3.0.3.zip	13.1 MB	Mar 19, 2015
Source code (zip)		May 2, 2013

Figure 6: GitHub CloudSim Simulator

- Once the cloudsim is downloaded we can import it to Eclipse IDE



cloudsim - cloudsim_powe_GSO2/src/org/cloudbus/cloudsim/examples/power/random/GSO.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloudsim_powe_GSO
cloudsim_powe_GSO1
cloudsim_powe_GSO2
cloudsim-3.0.3
org.cloudbus.cloudsim.examples
org.cloudbus.cloudsim.examples.network
org.cloudbus.cloudsim.examples.network.datacenter
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.planetlab
org.cloudbus.cloudsim.examples.power.random
workload.planetlab
sources
JRE System Library [jdk-17.0.2]
Referenced Libraries
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
iFogSim-main
GSO.java
GA.java
RandomConstants.java
RandomHelper.java
RandomRunner.java
Datacenter.java
Problems
Javadoc
Declaration
Console
ENG
IN
22:32
14-12-2022

```
 * @author Anton Beloglazov
 * @since Jan 5, 2012
 */
public class GSO {

	/**
	 * The main method.
	 *
	 * @param args the arguments
	 * @throws IOException Signals that an I/O exception has occurred.
	 */
	public static void main(String[] args) throws IOException {
		boolean enableOutput = false;
		boolean outputToFile = false;
		String inputFolder = "workload/planetlab";
		String outputFolder = "lib";
		String workload = "20110303"; // Random workload
		String vmAllocationPolicy = "GSO"; // Inter Quartile Range (IQR) VM allocation policy
		String vmSelectionPolicy = "mc"; // Maximum Correlation (MC) VM selection policy
		String parameter = "1.5"; // the safety parameter of the IQR policy

		new RandomRunner(
				enableOutput,
				outputToFile,
				inputFolder,
				outputFolder,
```

```
<terminated> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe  (14-Dec-2022, 10:06:47 pm – 10:07:10 pm) [pid: 29440]
StDev time before a host shutdown: 1474.58 sec
Mean time before a VM migration: 19.88 sec
StDev time before a VM migration: 8.00 sec
Execution time - VM selection mean: 0.00746 sec
Execution time - VM selection stDev: 0.01020 sec
Execution time - host selection mean: 0.00077 sec
Execution time - host selection stDev: 0.00062 sec
Execution time - VM reallocation mean: 0.00550 sec
Execution time - VM reallocation stDev: 0.00220 sec
Execution time - total mean: 0.01798 sec
Execution time - total stDev: 0.01139 sec
```

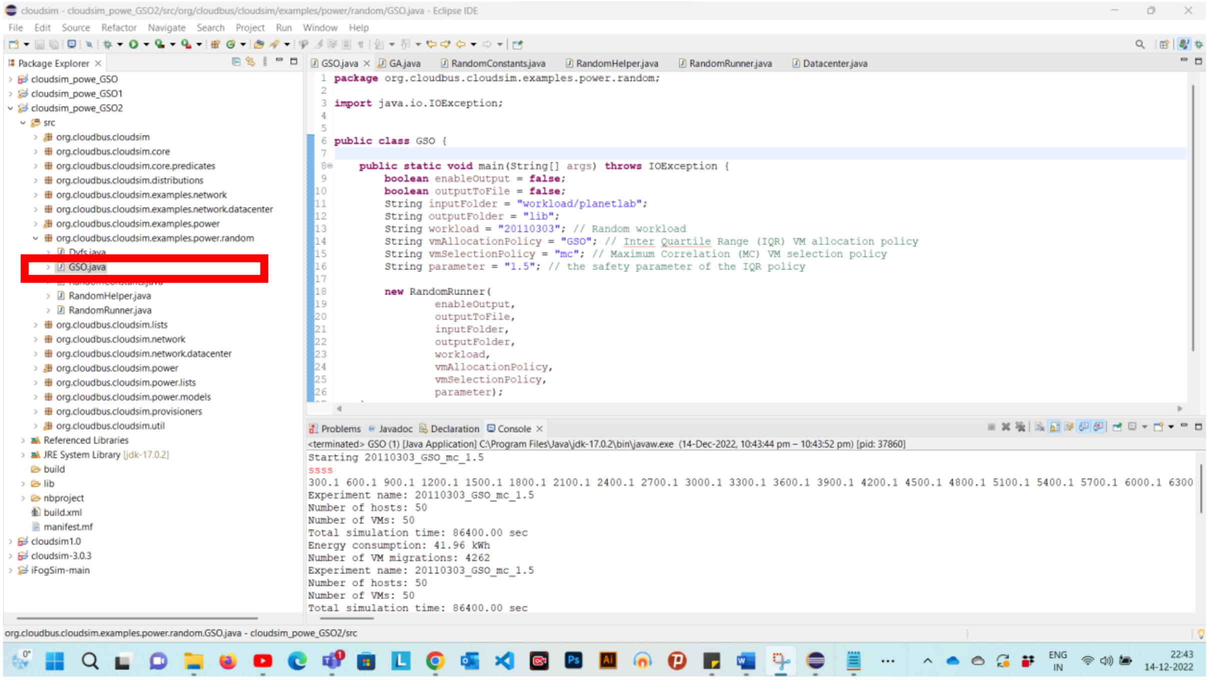
Figure 6: Import CloudSim 3.0.3

- Once Cloudsim is imported, import “**commons-math3-3.3**” library.

4 Project Development

Different project stages along with the results.

4.1 CloudSim Environment

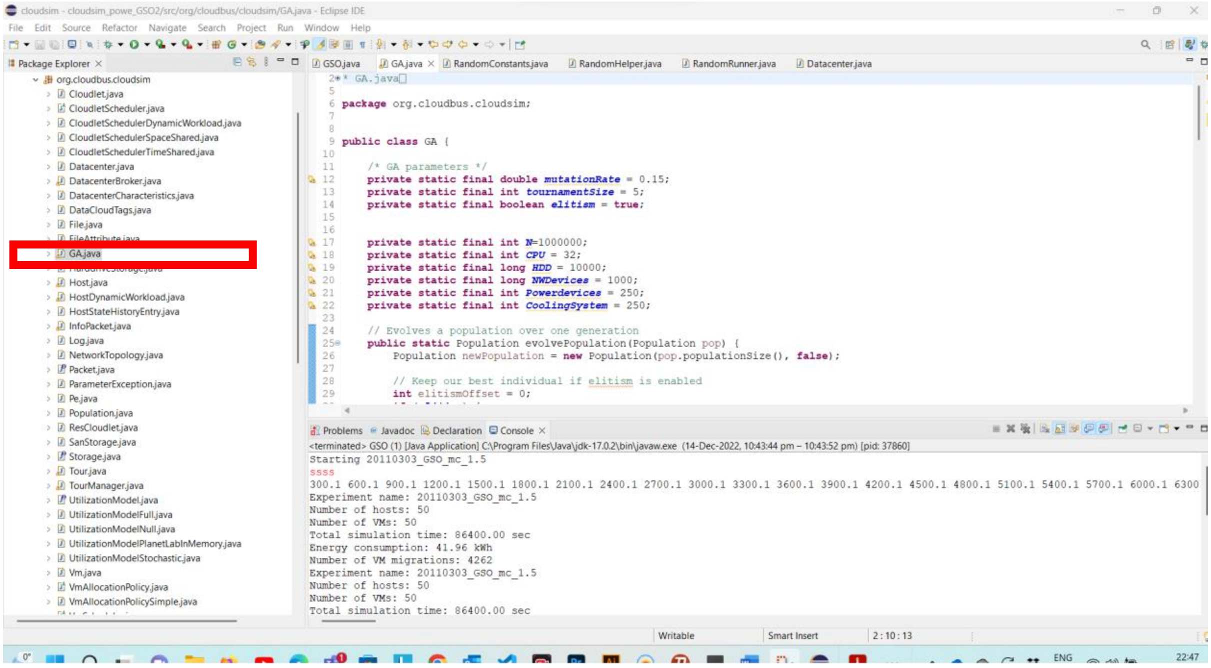

cloudsim - cloudsim_powe_GSO2/src/org/cloudbus/cloudsim/examples/power/random/GSO.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
GSO.java
RandomHelper.java
RandomRunner.java
Problems Javadoc Declaration Console

```
package org.cloudbus.cloudsim.examples.power.random;

import java.io.IOException;


public class GSO {

    public static void main(String[] args) throws IOException {
        boolean enableOutput = false;
        boolean outputToFile = false;
        String inputFolder = "workload/planetlab";
        String outputFolder = "lib";
        String workload = "20110303"; // Random workload
        String vmAllocationPolicy = "GSO"; // Inter Quartile Range (IQR) VM allocation policy
        String vmSelectionPolicy = "mc"; // Maximum Correlation (MC) VM selection policy
        String parameter = "1.5"; // the safety parameter of the IQR policy

        new RandomRunner(
                enableOutput,
                outputToFile,
                inputFolder,
                outputFolder,
                workload,
                vmAllocationPolicy,
                vmSelectionPolicy,
                parameter);
```

```
<terminated> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (14-Dec-2022, 10:43:44 pm – 10:43:52 pm) [pid: 37860]
Starting 20110303_GSO_mc_1.5
ssss
300.1 600.1 900.1 1200.1 1500.1 1800.1 2100.1 2400.1 2700.1 3000.1 3300.1 3600.1 3900.1 4200.1 4500.1 4800.1 5100.1 5400.1 5700.1 6000.1 6300
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 41.96 kWh
Number of VM migrations: 4262
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
```

Figure 7: Run GSO.java to show the results

4.2 Virtual Datacenter resource allocation algorithm logic in GA.java file.

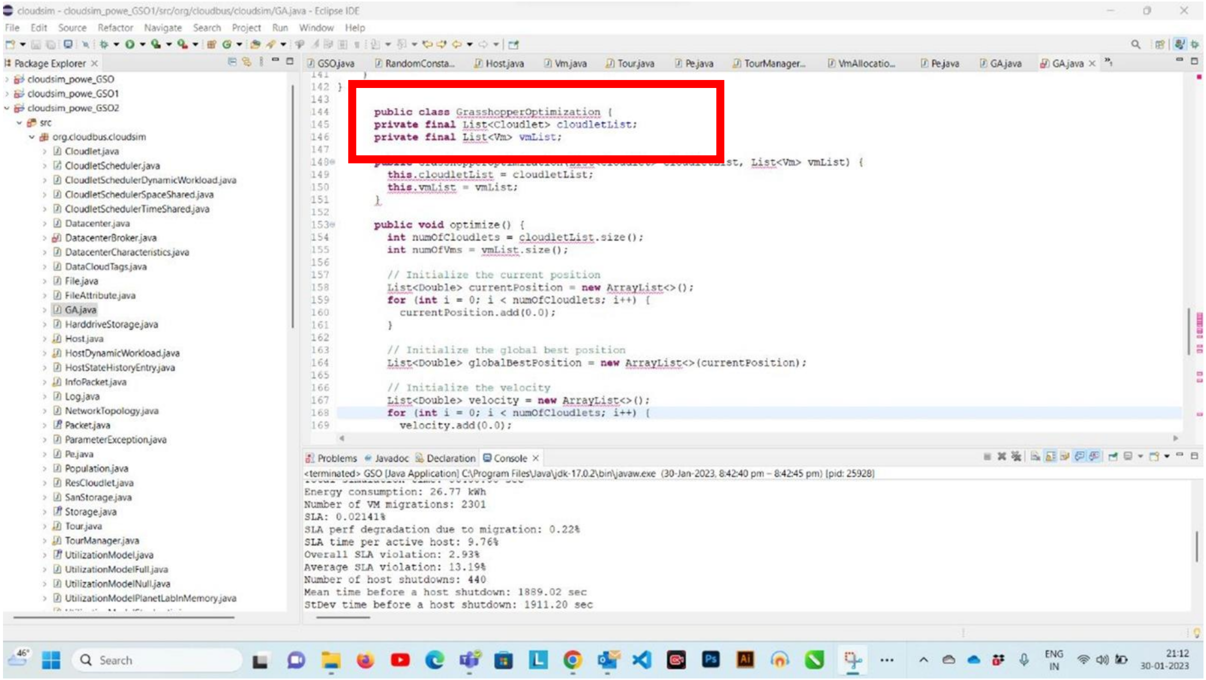

cloudsim - cloudsim_powe_GSO2/src/org/cloudbus/cloudsim/GA.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
GA.java
Problems Javadoc Declaration Console
Writable
Smart Insert
2 : 10 : 13

```
 * GA.java

package org.cloudbus.cloudsim;


public class GA {

    /* GA parameters */
    private static final double mutationRate = 0.15;
    private static final int tournamentSize = 5;
    private static final boolean elitism = true;


    private static final int N=1000000;
    private static final int CPU = 32;
    private static final long HDD = 10000;
    private static final long NWDevices = 1000;
    private static final int Powerdevices = 250;
    private static final int CoolingSystem = 250;

    // Evolves a population over one generation
    public static Population evolvePopulation(Population pop) {
        Population newPopulation = new Population(pop.populationSize(), false);

        // Keep our best individual if elitism is enabled
        int elitismOffset = 0;
```

```
<terminated> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (14-Dec-2022, 10:43:44 pm – 10:43:52 pm) [pid: 37860]
Starting 20110303_GSO_mc_1.5
ssss
300.1 600.1 900.1 1200.1 1500.1 1800.1 2100.1 2400.1 2700.1 3000.1 3300.1 3600.1 3900.1 4200.1 4500.1 4800.1 5100.1 5400.1 5700.1 6000.1 6300
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 41.96 kWh
Number of VM migrations: 4262
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
```

Figure 8: GA.java contains logical algorithm.

- GA is genetic algorithm used to implement our logical algorithm and defined our paraments of datacenter.

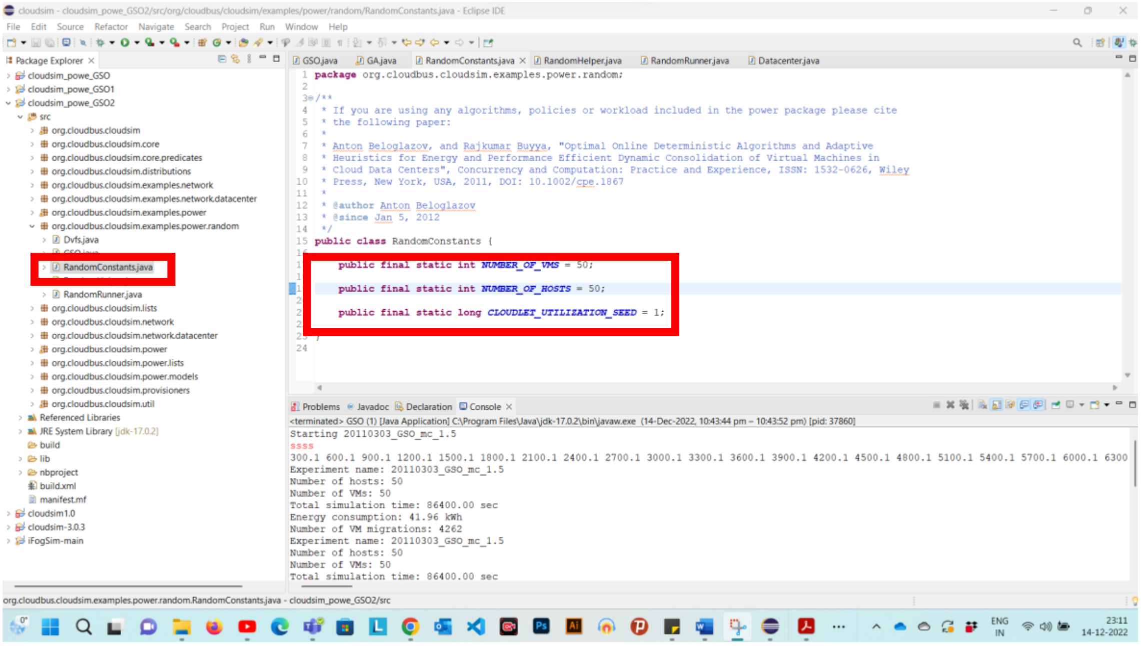


```
}

public class GrasshopperOptimization {
private final List<Cloudlet> cloudletList;
private final List<Vm> vmList;

public GrasshopperOptimization(List<Cloudlet> cloudletList, List<Vm> vmList) {
  this.cloudletList = cloudletList;
  this.vmList = vmList;
}

public void optimize() {
  int numOfCloudlets = cloudletList.size();
  int numOfVms = vmList.size();

  // Initialize the current position
  List<Double> currentPosition = new ArrayList<>();
  for (int i = 0; i < numOfCloudlets; i++) {
    currentPosition.add(0.0);
  }

  // Initialize the global best position
  List<Double> globalBestPosition = new ArrayList<>(currentPosition);

  // Initialize the velocity
  List<Double> velocity = new ArrayList<>();
  for (int i = 0; i < numOfCloudlets; i++) {
    velocity.add(0.0);
```

```
<terminated> GSO [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (30-Jan-2023, 8:42:40 pm – 8:42:45 pm) [pid: 25928]
Energy consumption: 26.77 kWh
Number of VM migrations: 2301
SLA: 0.02141%
SLA perf degradation due to migration: 0.22%
SLA time per active host: 9.76%
Overall SLA violation: 2.93%
Average SLA violation: 13.19%
Number of host shutdowns: 440
Mean time before a host shutdown: 1889.02 sec
StDev time before a host shutdown: 1911.20 sec
```

- GOA Perform the Grasshopper Optimization where we can optimize the tours by applying movement on each dimension.
- The proposed solution utilizes pre-defined packages provided by CloudSim to simulate the resource allocation in a data center environment. The program creates multiple random solutions, called tours, and then improves them using genetic algorithms and operations such as cross overs and mutations. These solutions are then modified according to the grasshopper algorithm, including variations in mutation rate and implementing the concept of "elitism" (preserving the best solution) to achieve the optimal solution.

4.3 Resource allocation.


cloudsim - cloudsim_powe_GSO2/src/org/cloudbus/cloudsim/examples/power/random/RandomConstants.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloudsim_powe_GSO
cloudsim_powe_GSO1
cloudsim_powe_GSO2
src
org.cloudbus.cloudsim
org.cloudbus.cloudsim.core
org.cloudbus.cloudsim.core.predicates
org.cloudbus.cloudsim.distributions
org.cloudbus.cloudsim.examples.network
org.cloudbus.cloudsim.examples.network.datacenter
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.random
Dvfs.java
RandomConstants.java
RandomRunner.java
org.cloudbus.cloudsim.lists
org.cloudbus.cloudsim.network
org.cloudbus.cloudsim.network.datacenter
org.cloudbus.cloudsim.power
org.cloudbus.cloudsim.power.lists
org.cloudbus.cloudsim.power.models
org.cloudbus.cloudsim.provisioners
org.cloudbus.cloudsim.util
Referenced Libraries
JRE System Library [jdk-17.0.2]
build
lib
nbproject
build.xml
manifest.mf
cloudsim1.0
cloudsim-3.0.3
iFogSim-main
GSO.java
GA.java
RandomConstants.java
RandomHelper.java
RandomRunner.java
Datacenter.java
Problems
Javadoc
Declaration
Console
org.cloudbus.cloudsim.examples.power.random.RandomConstants.java - cloudsim_powe_GSO2/src
ENG IN
23:11
14-12-2022

```
package org.cloudbus.cloudsim.examples.power.random;

/**
 * If you are using any algorithms, policies or workload included in the power package please cite
 * the following paper:
 *
 * Anton Beloglazov, and Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Adaptive
 * Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in
 * Cloud Data Centers", Concurrency and Computation: Practice and Experience, ISSN: 1532-0626, Wiley
 * Press, New York, USA, 2011, DOI: 10.1002/cpe.1867
 *
 * @author Anton Beloglazov
 * @since Jan 5, 2012
 */
public class RandomConstants {

    public final static int NUMBER_OF_VMS = 50;

    public final static int NUMBER_OF_HOSTS = 50;

    public final static long CLOUDLET_UTILIZATION_SEED = 1;

}
```

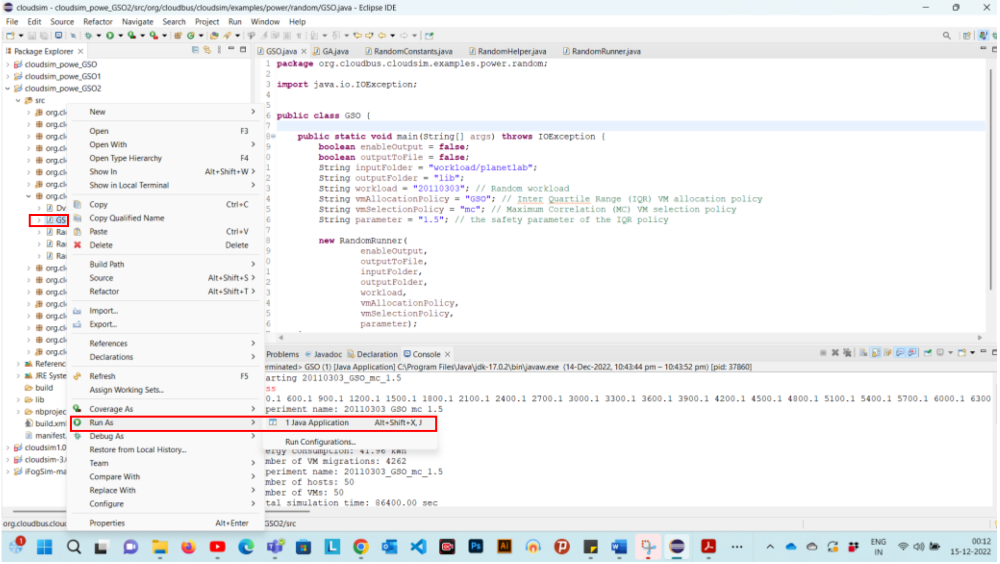
```
<terminated> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (14-Dec-2022, 10:43:44 pm – 10:43:52 pm) [pid: 37860]
Starting 20110303_GSO_mc_1.5
ssss
300.1 600.1 900.1 1200.1 1500.1 1800.1 2100.1 2400.1 2700.1 3000.1 3300.1 3600.1 3900.1 4200.1 4500.1 4800.1 5100.1 5400.1 5700.1 6000.1 6300
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 41.96 kWh
Number of VM migrations: 4262
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
```

Figure 9: Resource allocation

- **Random Constants** file under **power.random** can be modified as per the simulation requirements where we can assign the number of VM and Host we need to test our energy consumption.

4.4 Simulation Output.

- We will run the GSO.java to get the optimal results from our proposed algorithm.



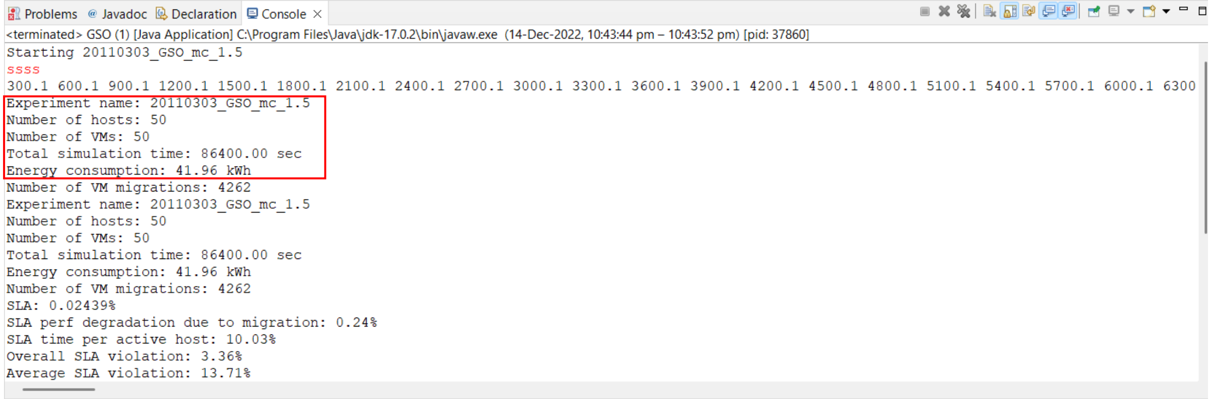
```
package org.cloudbus.cloudsim.examples.power.random;

import java.io.IOException;


public class GSO {

    public static void main(String[] args) throws IOException {
        boolean enableOutput = false;
        boolean outputToFile = false;
        String inputFolder = "workload/planetlab";
        String outputFolder = "lib";
        String workload = "20110303"; // Random workload
        String vmAllocationPolicy = "GSO"; // Inter Quartile Range (IQR) VM allocation policy
        String vmSelectionPolicy = "mc"; // Maximum Correlation (MC) VM selection policy
        String parameter = "1.5"; // the safety parameter of the IQR policy

        new RandomRunner(
                enableOutput,
                outputToFile,
                inputFolder,
                outputFolder,
                workload,
                vmAllocationPolicy,
                vmSelectionPolicy,
                parameter);
```

```
rminated> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (14-Dec-2022, 10:43:44 pm – 10:43:52 pm) [pid: 37860]
arting 20110303_GSO_mc_1.5
ss
0.1 600.1 900.1 1200.1 1500.1 1800.1 2100.1 2400.1 2700.1 3000.1 3300.1 3600.1 3900.1 4200.1 4500.1 4800.1 5100.1 5400.1 5700.1 6000.1 6300
periment name: 20110303 GSO mc 1.5
ergy consumption: 41.96 kWh
mber of VM migrations: 4262
periment name: 20110303_GSO_mc_1.5
mber of hosts: 50
mber of VMs: 50
tal simulation time: 86400.00 sec
```

Figure 10: Simulation Started

- Once we define the size of the VM & Hosts and start the simulation process the results will be obtain as shown in below section.

- Result with **proposed algorithm (EEDRA)** with Host and VM Size of 50.

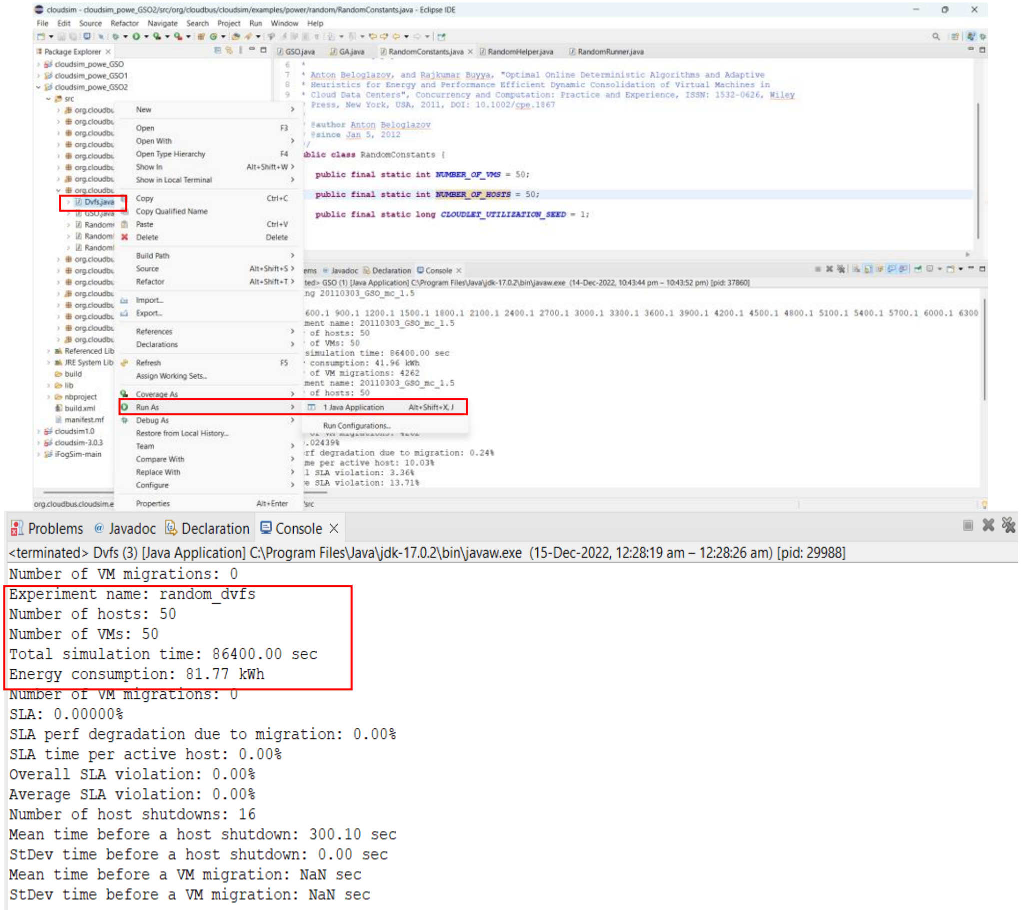


```
<terminated> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (14-Dec-2022, 10:43:44 pm – 10:43:52 pm) [pid: 37860]
Starting 20110303_GSO_mc_1.5
ssss
300.1 600.1 900.1 1200.1 1500.1 1800.1 2100.1 2400.1 2700.1 3000.1 3300.1 3600.1 3900.1 4200.1 4500.1 4800.1 5100.1 5400.1 5700.1 6000.1 6300
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 41.96 kWh
Number of VM migrations: 4262
Experiment name: 20110303_GSO_mc_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 41.96 kWh
Number of VM migrations: 4262
SLA: 0.02439%
SLA perf degradation due to migration: 0.24%
SLA time per active host: 10.03%
Overall SLA violation: 3.36%
Average SLA violation: 13.71%
```

Figure 11: Simulation Result

- As we can see the **total energy consumption** is around **41.96 kWh**. Which is expected result for our algorithm.

- Result with **DVFS algorithm** with Host and VM Size of 50.


cloudsim - cloudsim_powe_GSO2/src/org/cloudbus/cloudsim/examples/power/random/RandomConstants.java - Eclipse IDE
Package Explorer
Dvfs.java
Run As
1 Java Application Alt+Shift+X, J
Run Configurations...
Problems
Javadoc
Declaration
Console

```
 * Anton Beloglazov, and Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Adaptive
 * Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in
 * Cloud Data Centers", Concurrency and Computation: Practice and Experience, ISSN: 1532-0626, Wiley
 * Press, New York, USA, 2011, DOI: 10.1002/cpe.1867

 * @author Anton Beloglazov
 * @since Jan 5, 2012
 */
ublic class RandomConstants {

  public final static int NUMBER_OF_VMS = 50;

  public final static int NUMBER_OF_HOSTS = 50;

  public final static long CLOUDLET_UTILIZATION_SEED = 1;
```

```
ted> GSO (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (14-Dec-2022, 10:43:44 pm – 10:43:52 pm) [pid: 37860]
ng 20110303_GSO_mc_1.5

600.1 900.1 1200.1 1500.1 1800.1 2100.1 2400.1 2700.1 3000.1 3300.1 3600.1 3900.1 4200.1 4500.1 4800.1 5100.1 5400.1 5700.1 6000.1 6300
ment name: 20110303_GSO_mc_1.5
 of hosts: 50
 of VMs: 50
simulation time: 86400.00 sec
 consumption: 41.96 kWh
 of VM migrations: 4262
ment name: 20110303_GSO_mc_1.5
 of hosts: 50
.02439%
rf degradation due to migration: 0.24%
me per active host: 10.03%
l SLA violation: 3.36%
e SLA violation: 13.71%
```

```
<terminated> Dvfs (3) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe  (15-Dec-2022, 12:28:19 am – 12:28:26 am) [pid: 29988]
Number of VM migrations: 0
Experiment name: random_dvfs
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 81.77 kWh
Number of VM migrations: 0
SLA: 0.00000%
SLA perf degradation due to migration: 0.00%
SLA time per active host: 0.00%
Overall SLA violation: 0.00%
Average SLA violation: 0.00%
Number of host shutdowns: 16
Mean time before a host shutdown: 300.10 sec
StDev time before a host shutdown: 0.00 sec
Mean time before a VM migration: NaN sec
StDev time before a VM migration: NaN sec
```

Figure 12: Simulation Result of DVFS

- As we can see the **total energy consumption** is around **81.77 kWh**. Which is much higher than our proposed algorithm.

5 Conclusion

Complete setup instructions from beginning to end, as well as the necessary dependencies for energy consumption, are illustrated in the configuration manual. To ensure that the energy usage is consistent after each cycle, we have additionally tested our algorithms with various host and virtual machine sizes. As the demand for cloud datacentres increases, it is crucial to maximize resource usage across the long lifespan of the particular data centre along with its energy consumption. Making use of dynamic resource allocation we can effectively consume energy and contribute towards green cloud computing.

References

Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, *24*(13), 1397-1420.

Download the latest Java Lts Free. Oracle Ireland. (n.d.). Retrieved December 15, 2022, from https://www.oracle.com/ie/java/technologies/downloads/#jdk17-windows

Eclipse installer 2022-12 R. Eclipse Installer 2022-12 R | Eclipse Packages. (n.d.). Retrieved December 15, 2022, from https://www.eclipse.org/downloads/packages/installer

Cloudslab. (n.d.). Release cloudsim-3.0.3 · cloudslab/cloudsim. GitHub. Retrieved December 15, 2022, from https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3