Energy Monitoring & Reporting: Create a detailed energy consumption monitoring tool for VMs

Submitted in partial fulfilment of the **CAPSTONE PROJECT** in VIRTUALIZATION TECHNOLOGY, which is a part of

Integrated M. Tech. in Cybersecurity

By

Aakaash K S – 20MEI10055

Dinesh M - 20MEI10010

Prakadesh - 20MEI10076

Sriram K - 20MEI10062

Sai Kiran - 20MEI10064

Submitted to

Dr. Hemraj S. Lamkuche

School of Computing Science and Engineering,
VIT Bhopal University, Madhya Pradesh
India

October 2023

Motivation

Our motivation behind developing an energy consumption monitoring tool for Virtual Machines (VMs) stems from the pressing demand to enhance resource efficiency, reduce operational costs, and embrace environmental sustainability in the realm of data centres and cloud environments. We recognize the increasing importance of adhering to eco-friendly practices and industry regulations, as well as the desire to strike a balance between performance optimization and budgetary constraints. With these considerations in mind, we have committed to creating a versatile and precise tool that empowers organizations to make well-informed decisions regarding their energy consumption.

Capstone Project Approval

This is to certify that the Integrated M. Tech. Capstone Project report titled **“Energy Monitoring & Reporting: Create a detailed energy consumption monitoring tool for VMs”** by

Aakaash K S – 20MEI10055

Dinesh M - 20MEI10010

Prakadesh - 20MEI10076

Sriram K - 20MEI10062

Sai Kiran - 20MEI10064

is approved for the degree of **Integrated M. Tech. in Cybersecurity**.

Dr. Hemraj S. Lamkuche

(Course Coordinator)

Date: 26/10/2023

Place: Bhopal

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honestly and integrity and have not misrepresented or fabricated or falsified any ideas, data, facts or sources in my submission. I understand that any violation of the above will be cause of disciplinary action by the institute and evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 26/10/2023

Place: Bhopal

Error! Reference source not found.**(s) –**
Registration No

Aakaash K S – 20MEI10055

Dinesh M - 20MEI10010

Prakadesh - 20MEI10076

Sriram K - 20MEI10062

Sai Kiran - 20MEI10064

Abstract

Cloud computing is a rising technology. In which virtual machine migration and dynamic resource allocation can play a major role in providing services to the user. The user will send their access requests to the cloud constantly when migration was working. So here we are going to schedule those access requests to ensure the performance of cloud storage system. Our main goal is to implement the scheduling algorithm for migration of Hosts. So here we could establish the connection between system availability and resource availability to improve the performance of cloud storage. The scheduling algorithm can be implemented in CloudSim toolkit. It supports both system and behaviour modelling of cloud system using components such as data centres (DC), Hosts, Virtual machines (VMs).

Key words: Cloud computing, Cloud Sim, Storage, system availability and resource availability.

Table of Contents

Contents

Chapter 1
PROJECT DESCRIPTION AND OUTLINE

1.1 Introduction

In today's technology-driven environment, effective resource utilisation is critical. Because of the exponential rise of data centres and cloud computing, there is now a strong emphasis on energy usage in virtualized settings. As newer technology dissipates more power in server systems, the necessity for good energy management technologies becomes more obvious. This is especially visible in the rising costs of power supply and cooling in data centres, emphasising the importance of having effective energy monitoring and reporting technologies. This project aims to address the issue of managing energy usage in virtualized server systems. While standard energy management solutions were designed for standalone systems, they frequently fail to meet the unique requirements of individual virtual machines and their applications in complex virtualized settings.

1.2 Motivation for the work

Optimizing resource utilization, particularly energy utilization, has become a vital need in our quickly expanding technological world. With the exponential rise of data centers and cloud computing, effective energy monitoring and reporting technologies in virtualized systems are becoming increasingly important. Modern server technology demonstrates increasing power dissipation, emphasizing the significance of energy efficiency. Furthermore, the expenses of power supply and cooling have grown to be key drivers driving the need for sophisticated energy monitoring and reporting systems.

The need for comprehensive energy monitoring and reporting tools suited for Virtual Machines (VMs) within virtualized server systems drives this research. As data centers grow to meet the ever-increasing needs of contemporary computing, optimal resource utilization is critical. Virtualization technology, which allows several virtual computers to run on a single physical server, is a cost-effective method for increasing resource utilization and lowering operational expenses. However, this strategy complicates energy monitoring and reporting. Virtual machines run independently of one another, and conventional energy monitoring solutions frequently lack the flexibility to adjust to each virtual machine's unique software and application-specific energy consumption demands. As a result, there is an urgent need for a smart energy monitoring and reporting system

1.3 Project Statement

In the realm of energy monitoring and reporting, several critical challenges persist. These challenges are not specific to any project but encompass the broader context of energy management in virtualized server systems. The following key issues are frequently encountered:

1. **Lack of Accurate Hardware-Level Monitoring:** Current hardware-based monitoring and reporting systems often struggle to provide the necessary precision to meet the unique energy consumption requirements of individual virtual machines (VMs) and their applications. These tools are primarily designed for standalone systems and do not readily adapt to the complexities of virtualized environments.
2. **Limitations of Guest Operating Systems:** Guest operating systems within VMs, while more suitable for fine-grained energy monitoring and reporting, face limitations in aggregating and reporting energy usage across the entire VM. VM isolation and the lack of awareness among VMs present complexities in precise energy monitoring and reporting in virtualized systems.

3. **Unifying Energy Monitoring and Reporting Solution:** The overarching challenge is to develop a unified energy monitoring and reporting solution capable of providing detailed insights into the energy consumption of virtual machines, applications within them, and the underlying hardware. This solution must operate at various granularities and offer actionable data for optimizing resource utilization while ensuring high accuracy and reliability.

1.4 Objectives of our work

The primary objective of this project is to develop an intuitive, external GUI tool designed to facilitate the accurate calculation and visual representation of power consumption for virtualized environments. This tool will enable users to input initialization details for both the host server and guest VMs, including parameters such as the number of VMs, runtime, RAM allocation, and other relevant factors. The overarching goals are as follows:

1. **Precise Power Consumption Calculation:** Develop a powerful computational engine that accurately calculates the power consumption of both the host server and individual guest VMs. This engine will take into account parameters such as CPU usage, RAM allocation, storage, and other hardware-specific details to provide precise energy consumption estimations.
2. **Intuitive GUI Interface:** Create a user-friendly graphical interface that allows users to easily input initialization details for the host and guest VMs. This interface should be intuitive and straightforward, making it accessible to users with varying degrees of technical expertise.
3. **Real-Time Meter Visualizations:** Implement a graphical representation module that provides real-time visual displays of power consumption, resembling the design of meters. Users will be able to see power usage presented in a format that allows for quick and intuitive understanding of power consumption levels.

Chapter 2
RELATED WORK INVESTIGATION

2.1 Energy Monitoring & Reporting for Virtual Machines

Virtual Machines (VMs) are essential in contemporary data centers and server environments in the age of cloud computing. These VMs give businesses the ability to scale applications, allocate resources more effectively, and spend less on hardware. However, having thorough energy consumption monitoring tools is crucial for maximizing both performance and sustainability. The main topic of developing a thorough energy consumption monitoring tool for virtual machines is explored in this article.

Data Gathering: Gathering pertinent data is the initial stage in developing an energy consumption monitoring tool for virtual machines. This entails tracking the host server's hardware resources, such as CPU, RAM, and network traffic, and, most crucially, calculating the server's power consumption. Specialized hardware or software solutions can be used for power monitoring. The combination of these data sources offers a comprehensive understanding of the energy usage incurred by VM activities.

Data Aggregation and Analysis: Data collection, aggregation, and analysis are required steps after data collection. Consideration should be given to VM-specific data, such as the number of VMs running on a host, their resource allocations, and their activity levels. This information is essential for figuring out how energy-efficient VMs are and identifying areas for improvement. Trends, abnormalities, and chances for optimization can be found via advanced analytics.

Resource Allocation and Scheduling: The next step is to reduce energy consumption by optimizing VM resource scheduling and allocation. This entails allocating resources in accordance with the energy profiles of the VMs, combining workloads to cut down on server downtime, and ensuring that VMs are powered down when not in use. Energy efficiency can be further improved by making dynamic changes to VM resource allocations.

Data Visualization and Reporting: Data from energy monitoring should be displayed in an easy-to-understand manner. Dashboards displaying energy consumption metrics, resource utilization, and efficiency indicators can be made using data visualization tools. To track long-term trends and advancements in energy efficiency, reports can be produced daily, weekly, or monthly.

Integration with Management Tools: Tools for cloud orchestration and virtualization management should be easily integrated with the energy monitoring tool. With the help of this integration, administrators can automate energy-saving procedures and make real-time adjustments to VMs based on data about energy consumption.

Machine Learning and Predictive Analysis: Machine learning algorithms can be used as the system develops to forecast energy consumption trends and provide proactive energy optimization recommendations. Machine learning models can help to cut down on energy waste by continuously adapting to changing workloads.

2.2 Literature Review

In this study, different Energy-efficient Virtual Machine placement techniques are presented.

Perla Ravi Theja has proposed an Adaptive-Genetic algorithm that tracks host underload and overload detection, performs minimum migration time VM selection and places VM to host such that energy consumption and SLA violation are reduced. This algorithm uses local regression and interquartile range schemes to track CPU utilization dynamically.

M H Malekloo, N Kara has proposed Multi-objective Ant Colony Optimization algorithm which focused on green cloud computing. This approach attains lower energy consumption in VM placement, reduces CPU energy wastage and also the energy cost required for the communication of traffic load among the VMs. It uses two complementary algorithms: ACO VMs placement and Multi-objective ACO VM consolidation which consolidates VMs on to less number of PMs(Physical machines).

F Abdessamia proposed a PSO-based VM placement algorithm for heterogeneous environment. This technique is modified one of PSO which performs better when compared to existing first-fit, best-fit and worst-fit algorithms and reduces the energy consumption. Algorithm is experimented using cloudsim tool.

Xiong Fu used binary particle swarm optimization(BPSO) algorithm for energy-efficient VM placement, which uses fitness function to reduce number of active hosts in the data center. This approach focuses on CPU utilization and disk resources of host to reduce energy consumption.

A Ibrahim and M Noshy proposed a power-aware VM placement algorithm based on particle swarm optimization (PAPSO) which maps VM to near and best appropriate Host. This technique uses minimization fitness function to place VMs to minimal number of Hosts to reduce energy consumption and number of active hosts in data center. It also reduces number of VM migrations and experiment is deployed in Cloudsim. Proposed algorithm is compared with existing PABFD and PAPSO out-performs PABFD in metrics: energy consumption, number of VM migrations, number of active servers, SLA violation rate.

2.3 Pros and cons of the stated Approaches/Methods

a. Hypervisor-based Monitoring:

- **Pros:**

  1. Direct access to VM-level data: Hypervisors can provide real-time data on each VM's energy consumption, making it highly accurate.
  2. Low impact on VM performance: Hypervisors are designed to minimize overhead, ensuring minimal impact on VM workloads.

- **Cons:**

  1. Limited to supported hypervisors: Not all hypervisors offer energy consumption monitoring capabilities, restricting its applicability.
  2. Vendor-specific: Different hypervisor vendors may have proprietary solutions, limiting cross-platform compatibility.

b. Agent-based Monitoring:

- **Pros:**

  1. Versatility: Agents can be deployed on various VMs, irrespective of the underlying hypervisor or cloud platform.
  2. Granular data collection: Agents can collect detailed data about not only energy consumption but also system-level metrics.

- **Cons:**
    1. Resource overhead: Agents can introduce some overhead, especially in terms of CPU and memory usage on the VM.
    2. Maintenance: Managing and updating agents across numerous VMs can be cumbersome and resource-intensive.

c. Power Metering Infrastructure:

- **Pros:**
    1. Direct hardware-level data: Power meters placed at the server level provide accurate, real-time energy consumption data.
    2. Platform-agnostic: This approach works across different hypervisors and hardware platforms.
- **Cons:**
    1. Deployment complexity: Installing and maintaining power meters can be challenging, especially in existing data centers.
    2. Cost: The initial setup cost and maintenance of power metering infrastructure can be substantial.

d. Software-defined Networking (SDN) Approach:

- **Pros:**
    1. Network-level insights: SDN solutions can indirectly monitor energy consumption by analyzing network traffic patterns.
    2. No impact on VMs: This method does not introduce any overhead on VMs themselves.
- **Cons:**
    1. Indirect measurement: The data collected via SDN may not be as precise as direct monitoring methods.
    2. Limited to network-based insights: SDN primarily provides network-related data, not VM-specific consumption.

Chapter 3
TECHNICAL IMPLEMENTATION & ANALYSIS

3.1 Outline

The application we are creating is a tool that will monitor the energy consumption of the provided configuration. These configurations include:

- No of host
- No of host cores
- Host ram
- Host storage
- Host bandwidth
- No of VM
- No of VM cores
- VM ram
- VM storage
- VM bandwidth
- No of cloudlets
- No of Cloudlets cores
- Cloudlets length

This tool utilizes the following tools and frameworks to achieve the desired energy consumption based on the user-inputted parameters mentioned above:

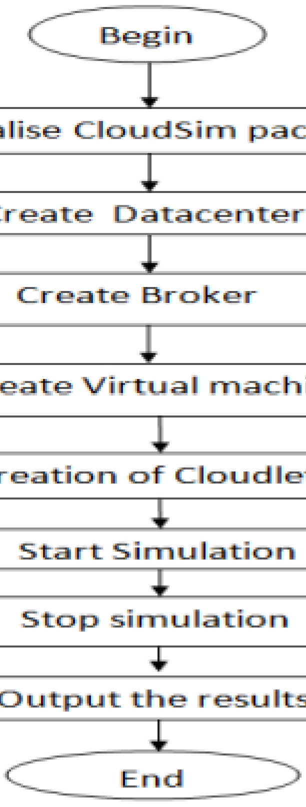
- Cloudsim Plus: A Java 17 simulation framework that simplifies Cloud computing modelling and experimentation. It empowers developers to focus on design without worrying about low-level infrastructure details, based on Cloudsim.
- Medusa: A gauge library for JavaFX that enables the creation of visually appealing and interactive gauges and displays.
- OpenJFX (JavaFX): A framework for building rich client applications using Java. It's commonly used for creating graphical user interfaces (GUIs) in Java applications.

- Maven: A project management and build automation tool. It simplifies the build process and handles project dependencies.
    - Let’s dive deep into these components in the upcoming sections.

3.2 Technical coding and code solutions

As mentioned in the earlier section that we be using Cloud Simplus framework for simulation purposes lets breakdown the components of cloud Simplus for a better understanding of the application:

- Datacentre: it represents a location of a cloud infrastructure with its VM and hosts.
- Hosts: Hosts are physical machines or servers in the datacentre that run multiple VMs. They are responsible for executing user tasks and managing the resources allocated to VMs.
- Virtual Machine: VMs are virtualized computing instances that run on hosts. Cloudsim Plus supports different VM allocation policies.
- Cloudlets: they are the cloud applications or user defined task which will be executed in a cloud environment
- Brokers: They act as a intermediate between the cloud user and the datacentre where it submit the cloudlets to the datacentre and handle the I/O
- Simulation: simulation entities and events to model the passage of time, scheduling of tasks, and other aspects of the simulation process.
- Policies: these policies play a key role as they are used to allocate resources for VM
- DatacenterBroker: these brokers are specialized in submits cloudlets to the datacentre on behalf of cloud users. It can use different VM allocation and cloudlet scheduling policies.
- PE: Processing Element they are the core of a host or the virtual machine they generally represent CPU.

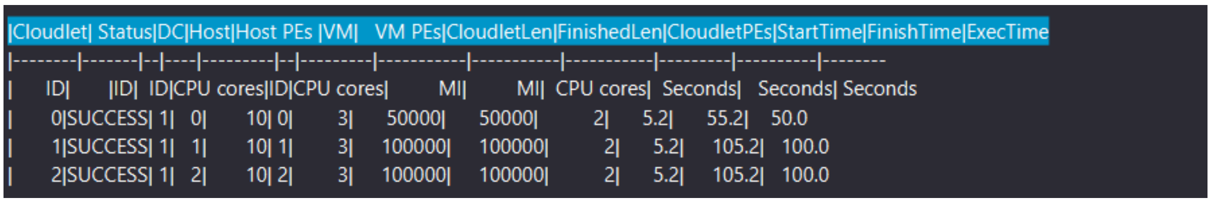


The diagram above shows the basic working procedure of CloudSim. The first step is to initialize the CloudSim package using the **init()** method, which initializes the entities and variables of CloudSim. The second step involves creating a Datacenter using base classes. Here we are creating a datacentre virtually because CloudSim is a toolkit, and we are analyzing it with our own configurations. After creating Datacentres, hosts are constructed with attributes such as ID, bandwidth, storage, PE list (number of CPUs), and the type of scheduler. Next, we create brokers, which serve as an interface between a user and a provider. Then, VMs and cloudlets are submitted by the user. To start the simulation, we call the **startSimulation** function, which initiates the simulation process. After completing all the tasks, we stop the simulation step by step using the **stopSimulation** method. The output is displayed in the command prompt with each event.

Now, we can proceed with the code explanation part to understand the functionality of the application.

Let’s Starts with a power model class from cloudsimplus example then based on that we start to work on the backend of the application we start by creating a project in the IDE with maven then we will use the pom.xml to add the required plugins and dependencies such as Medusa, cloudsimplus and openjfx.

We will start by focus on a backend of program which with the help of a cloudsimplus example we are able to learn on how to use cloudsimplus once we aredone.
we start by creating a class called as energymonitoringapplications.java then based on the power example we are receiving the output like this:



```
|Cloudlet| Status|DC|Host|Host PEs |VM|   VM PEs|CloudletLen|FinishedLen|CloudletPEs|StartTime|FinishTime|ExecTime
|--------|-------|--|----|---------|--|---------|-----------|-----------|-----------|---------|----------|--------
|      ID|       |ID|  ID|CPU cores|ID|CPU cores|         MI|         MI|  CPU cores|  Seconds|   Seconds| Seconds
|       0|SUCCESS| 1|   0|       10| 0|        3|      50000|      50000|          2|      5.2|      55.2|    50.0
|       1|SUCCESS| 1|   1|       10| 1|        3|     100000|     100000|          2|      5.2|     105.2|   100.0
|       2|SUCCESS| 1|   2|       10| 2|        3|     100000|     100000|          2|      5.2|     105.2|   100.0
```

Then we use a the received values such as time and a power consumption to compute the energy consumption of the VM/host using the formula,

(E=Pt)

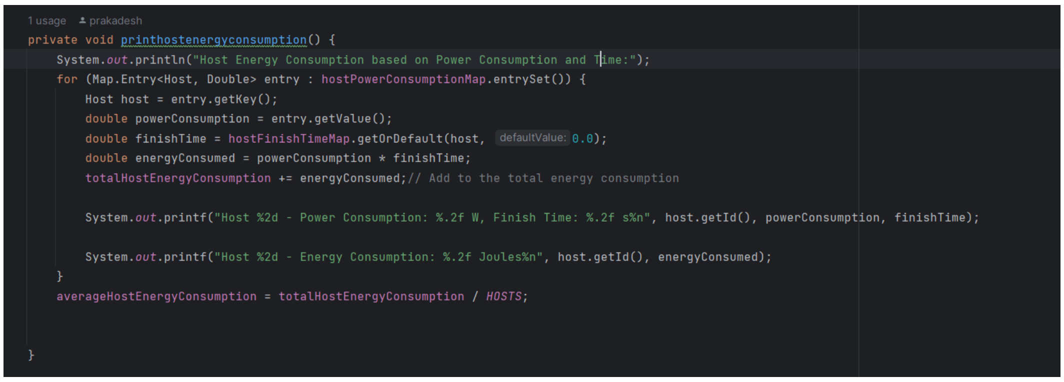
P= power consumption

T = time

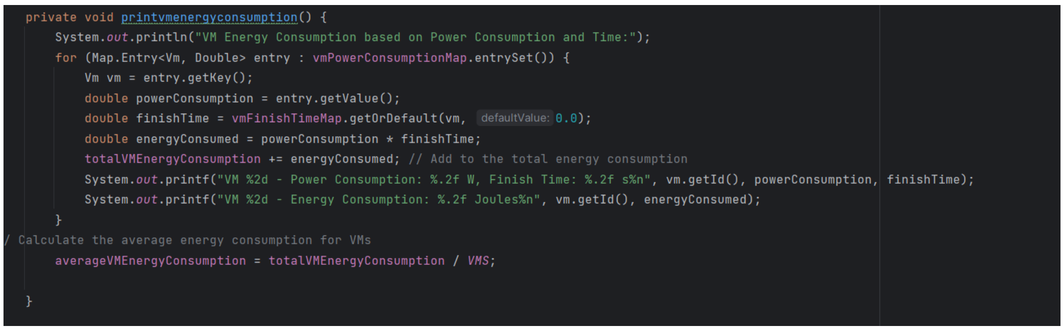
Then we will create a constructor to apply this in code we will use hashmap to store the time of host and VM in a separate hashmap call it as VMFinishTimeMap and hostFinishTimeMap.

And also do the same to store the power consumption in a separate hashmap call it as hostPowerConsumptionMap and VMPowerConsumptionMap.

Then we create two constructors called as print VMenergyconsumption and printhostenergyconsumption in the EnerymonitoringApplication class.

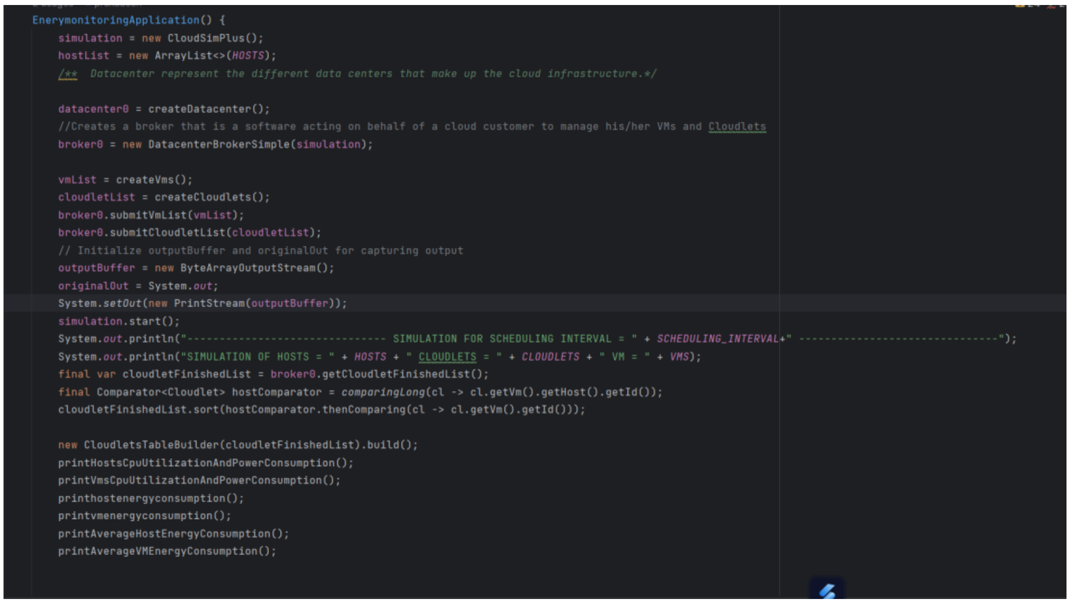


```
1 usage  prakadesh
private void printhostenergyconsumption() {
    System.out.println("Host Energy Consumption based on Power Consumption and Time:");
    for (Map.Entry<Host, Double> entry : hostPowerConsumptionMap.entrySet()) {
        Host host = entry.getKey();
        double powerConsumption = entry.getValue();
        double finishTime = hostFinishTimeMap.getOrDefault(host,  defaultValue: 0.0);
        double energyConsumed = powerConsumption * finishTime;
        totalHostEnergyConsumption += energyConsumed;// Add to the total energy consumption

        System.out.printf("Host %2d - Power Consumption: %.2f W, Finish Time: %.2f s%n", host.getId(), powerConsumption, finishTime);

        System.out.printf("Host %2d - Energy Consumption: %.2f Joules%n", host.getId(), energyConsumed);
    }
    averageHostEnergyConsumption = totalHostEnergyConsumption / HOSTS;


}
```



```
private void printvmenergyconsumption() {
    System.out.println("VM Energy Consumption based on Power Consumption and Time:");
    for (Map.Entry<Vm, Double> entry : vmPowerConsumptionMap.entrySet()) {
        Vm vm = entry.getKey();
        double powerConsumption = entry.getValue();
        double finishTime = vmFinishTimeMap.getOrDefault(vm,  defaultValue: 0.0);
        double energyConsumed = powerConsumption * finishTime;
        totalVMEnergyConsumption += energyConsumed; // Add to the total energy consumption
        System.out.printf("VM %2d - Power Consumption: %.2f W, Finish Time: %.2f s%n", vm.getId(), powerConsumption, finishTime);
        System.out.printf("VM %2d - Energy Consumption: %.2f Joules%n", vm.getId(), energyConsumed);
    }
/ Calculate the average energy consumption for VMs
    averageVMEnergyConsumption = totalVMEnergyConsumption / VMS;

}
```

Then we will add these constructor to be called in start (EnerymonitoringApplication)function like this



```
EnerymonitoringApplication() {
    simulation = new CloudSimPlus();
    hostList = new ArrayList<>(HOSTS);
    /**  Datacenter represent the different data centers that make up the cloud infrastructure.*/

    datacenter0 = createDatacenter();
    //Creates a broker that is a software acting on behalf of a cloud customer to manage his/her VMs and Cloudlets
    broker0 = new DatacenterBrokerSimple(simulation);

    vmList = createVms();
    cloudletList = createCloudlets();
    broker0.submitVmList(vmList);
    broker0.submitCloudletList(cloudletList);
    // Initialize outputBuffer and originalOut for capturing output
    outputBuffer = new ByteArrayOutputStream();
    originalOut = System.out;
    System.setOut(new PrintStream(outputBuffer));
    simulation.start();
    System.out.println("------------------------------ SIMULATION FOR SCHEDULING INTERVAL = " + SCHEDULING_INTERVAL+" ------------------------------");
    System.out.println("SIMULATION OF HOSTS = " + HOSTS + " CLOUDLETS = " + CLOUDLETS + " VM = " + VMS);
    final var cloudletFinishedList = broker0.getCloudletFinishedList();
    final Comparator<Cloudlet> hostComparator = comparingLong(cl -> cl.getVm().getHost().getId());
    cloudletFinishedList.sort(hostComparator.thenComparing(cl -> cl.getVm().getId()));

    new CloudletsTableBuilder(cloudletFinishedList).build();
    printHostsCpuUtilizationAndPowerConsumption();
    printVmsCpuUtilizationAndPowerConsumption();
    printhostenergyconsumption();
    printvmenergyconsumption();
    printAverageHostEnergyConsumption();
    printAverageVMEnergyConsumption();
```

Then we will receive the output like this :

```
VM Energy Consumption based on Power Consumption and Time:

        VM  2 - Power Consumption: 17.50 W, Finish Time: 105.32 s

        VM  2 - Energy Consumption: 1843.10 Joules
```

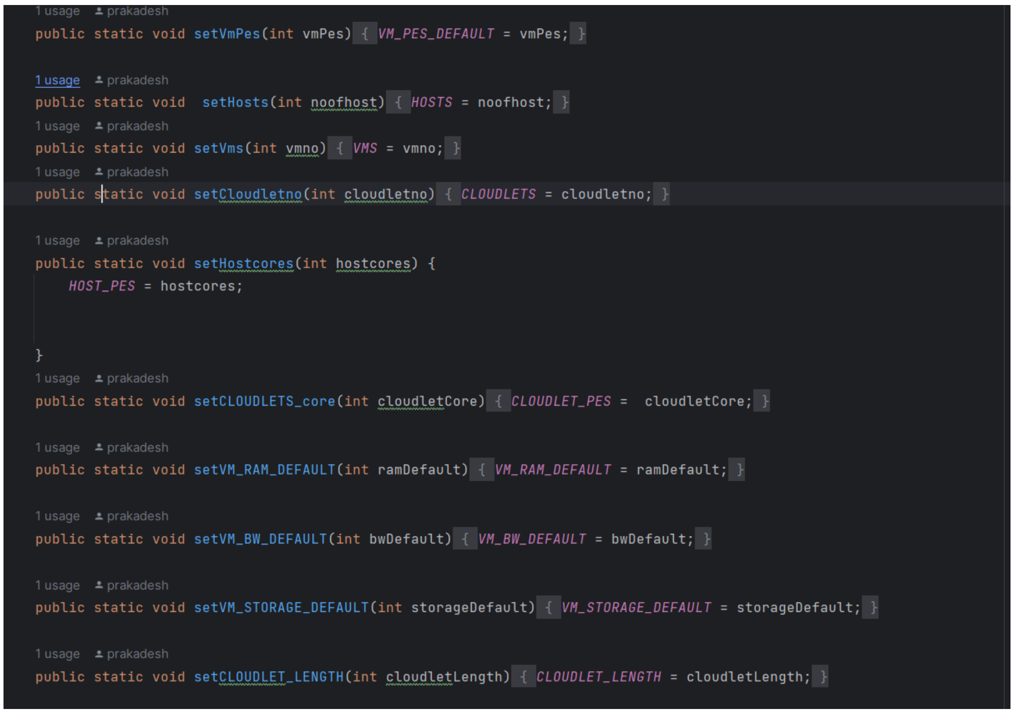
```
Host Energy Consumption based on Power Consumption and Time:

Host  2 - Power Consumption: 35.00 W, Finish Time: 105.32 s

Host  2 - Energy Consumption: 3686.20 Joules

Host  4 - Power Consumption: 35.00 W, Finish Time: 105.32 s

Host  4 - Energy Consumption: 3686.20 Joules
```

Now we got the backend ready for the initial phase. Where once we define a value for configuration such as VM no host no VM pes host pes we can provide them with a output of the energy consumption of each host and VM along with the time taken and power consumption

Then we will send the output of class energymonitoringapplications to the CustomOutputStream class which will display result in the GUI

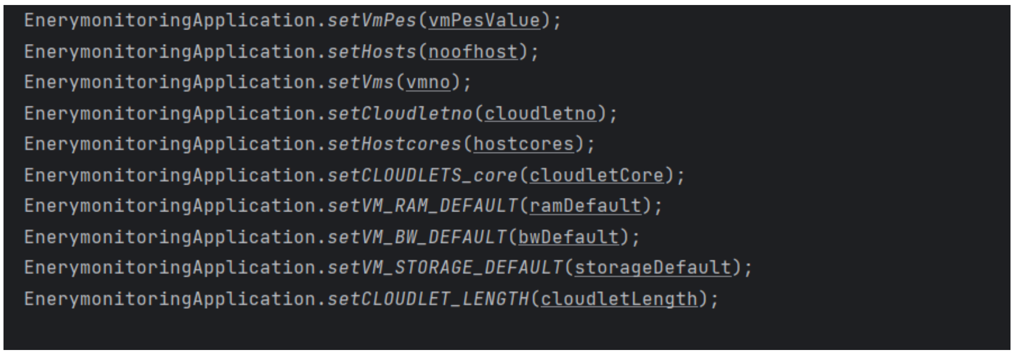
Also we will create a getter method which will send the average energy consumption of host and VM when called

We will use setter method to define the configuration for the energymonitoringapplications class like this :



```
1 usage   prakadesh
public static void setVmPes(int vmPes) { VM_PES_DEFAULT = vmPes; }

1 usage   prakadesh
public static void  setHosts(int noofhost) { HOSTS = noofhost; }
1 usage   prakadesh
public static void setVms(int vmno) { VMS = vmno; }
1 usage   prakadesh
public static void setCloudletno(int cloudletno) { CLOUDLETS = cloudletno; }

1 usage   prakadesh
public static void setHostcores(int hostcores) {
    HOST_PES = hostcores;


}
1 usage   prakadesh
public static void setCLOUDLETS_core(int cloudletCore) { CLOUDLET_PES =  cloudletCore; }

1 usage   prakadesh
public static void setVM_RAM_DEFAULT(int ramDefault) { VM_RAM_DEFAULT = ramDefault; }

1 usage   prakadesh
public static void setVM_BW_DEFAULT(int bwDefault) { VM_BW_DEFAULT = bwDefault; }

1 usage   prakadesh
public static void setVM_STORAGE_DEFAULT(int storageDefault) { VM_STORAGE_DEFAULT = storageDefault; }

1 usage   prakadesh
public static void setCLOUDLET_LENGTH(int cloudletLength) { CLOUDLET_LENGTH = cloudletLength; }
```

These can be called to update the value like these:



```
EnerymonitoringApplication.setVmPes(vmPesValue);
EnerymonitoringApplication.setHosts(noofhost);
EnerymonitoringApplication.setVms(vmno);
EnerymonitoringApplication.setCloudletno(cloudletno);
EnerymonitoringApplication.setHostcores(hostcores);
EnerymonitoringApplication.setCLOUDLETS_core(cloudletCore);
EnerymonitoringApplication.setVM_RAM_DEFAULT(ramDefault);
EnerymonitoringApplication.setVM_BW_DEFAULT(bwDefault);
EnerymonitoringApplication.setVM_STORAGE_DEFAULT(storageDefault);
EnerymonitoringApplication.setCLOUDLET_LENGTH(cloudletLength);
```

Then we can start with a GUI part where a user can input these configurations in the GUI rather than having to define it in a code as a variable we use the above setter method to communicate between frontend and backend.

As discussed earlier we will be using JavaFX for GUI,

- Lets start by creating a class called as Graphical_User_Interface
- We will make this class as a main class so that once a run is selected then the GUI is initialized we can make this class as a main class by adding it in the pom.xml file
- Once we have completed with defining it as a main class then we start by defining the buttons and ui elements to be used such as TextArea , Slider,gauges from medusa package
- We declare a main class with launch config
- Then a start constructor for primary stage

We will initialize the components

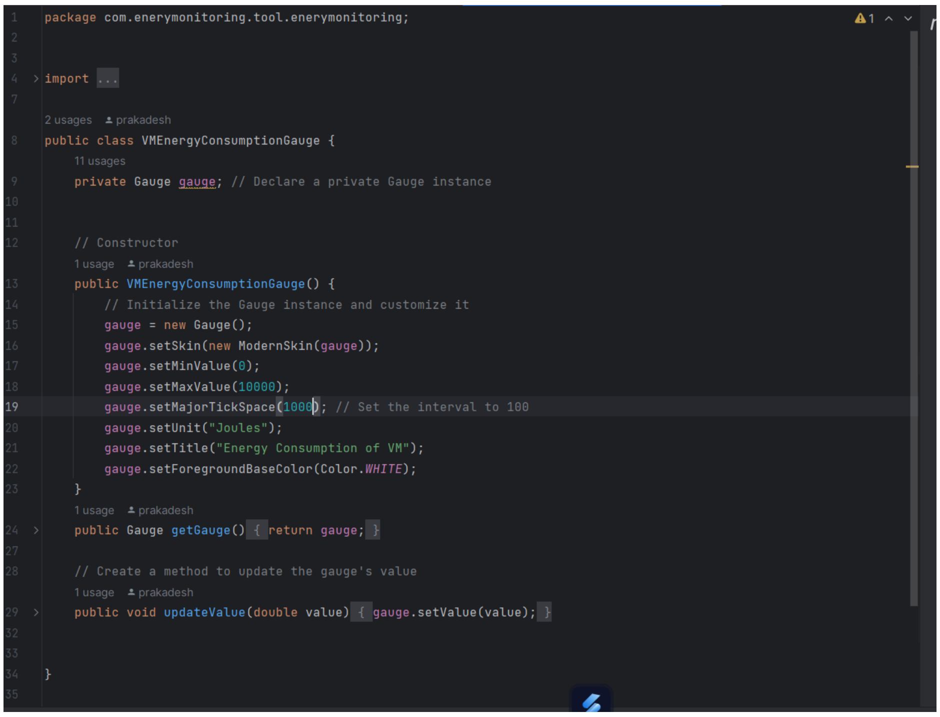
- Gauge for host and VM to showcase energy consumption
- Then a ConfigurationManager will be initialized so that users can select from a list of predefined configurations which can be selected with the help of a combobox
- Then config the combobox on action functions
- Then initialize a vbox (vertical box) to hold the combobox
- Then Hbox (horizontal box) to hold the Two gauge and a about us button

- About us page will contain info regarding the team member and project information

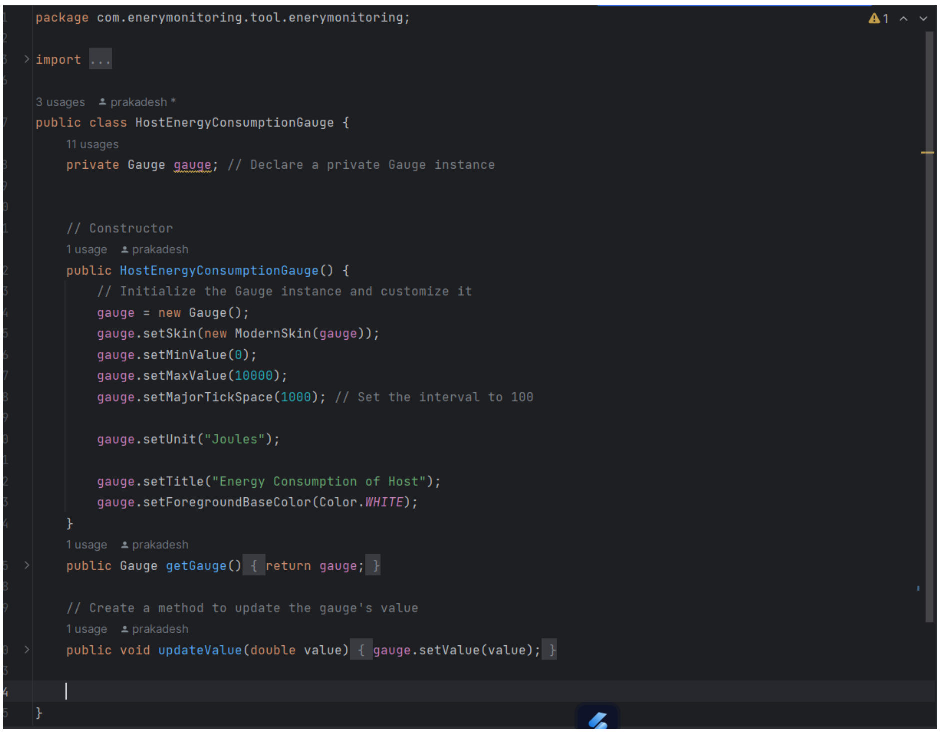
- Then we will create a sliders with label on each such as
  - VM_core
  - VM_no
  - Host_core
  - Host_no
  - Cloudlets_core
  - Cloudlets_no
  - VM_ram
  - VM_bandwith
  - VM_storage
  - Cloudlets_length
  - More…
- Then we use listeners for those sliders so that the user input parameters are being sent to the backend also to update the label with the position of the slider
- Then we will create a start button and name it then create another HBOX to hold the buttons
- Then we will create a final VBOX called as vbox_final which will hold the elements with their HBOXs or VBOXs
- Then define the margin for the final VBOX
- Then use dark theme for all components of FINAL_VBOX
- Then make the final vbox under a ScrollPane
- Then we close the constructor start of primary scene
- Then we will define the following action which need to be taken place when a user selects the startbutton
  - We will start by fetching all the sliders value
  - Then calculate a total hostcores , hostram,

  - Then with the help of these we will check if there was any resource allocation or resource constraints if there was any we will show a error in the textbox else we will procced
  - Then we will check if any slider value is null or zero if it then we us a default which are predefined for testing purposes
  - Then we will send those values to the backend using EnerymonitoringApplication.setVMPes(VMPesValue); EnerymonitoringApplication.setHosts(noofhost);
  - Then we will start the backend functions then we will fetch the values for host and VM energy consumption and set those values for there respective gauges
  - Then we will use the outputextarea class to show the output in a textarea from the output of the backend server
  - Closes the start button constructor
- Code snippet for other class which are used in the gui parts are

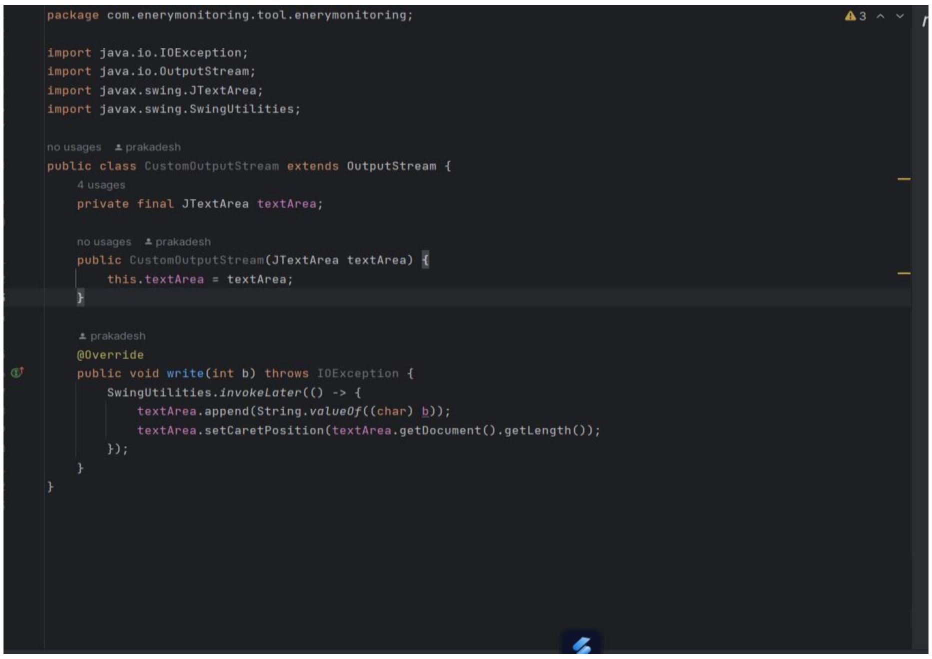
Given below with their functions,



```
package com.enerymonitoring.tool.enerymonitoring;


import ...

2 usages  prakadesh
public class VMEnergyConsumptionGauge {
    11 usages
    private Gauge gauge; // Declare a private Gauge instance


    // Constructor
    1 usage  prakadesh
    public VMEnergyConsumptionGauge() {
        // Initialize the Gauge instance and customize it
        gauge = new Gauge();
        gauge.setSkin(new ModernSkin(gauge));
        gauge.setMinValue(0);
        gauge.setMaxValue(10000);
        gauge.setMajorTickSpace(1000); // Set the interval to 100
        gauge.setUnit("Joules");
        gauge.setTitle("Energy Consumption of VM");
        gauge.setForegroundBaseColor(Color.WHITE);
    }
    1 usage  prakadesh
    public Gauge getGauge() { return gauge; }

    // Create a method to update the gauge's value
    1 usage  prakadesh
    public void updateValue(double value) { gauge.setValue(value); }


}
```



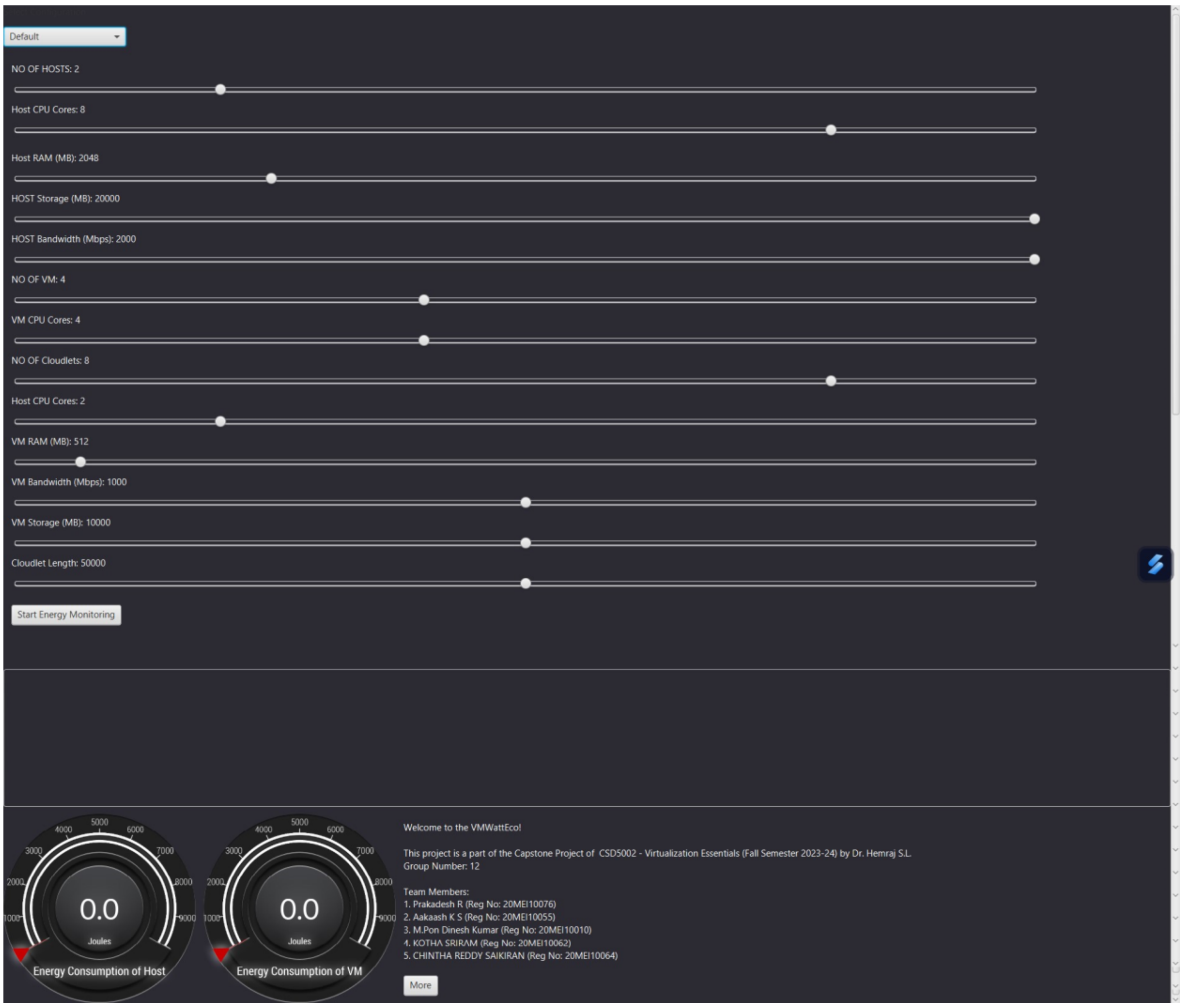
```java
package com.enerymonitoring.tool.enerymonitoring;

import ...

3 usages  prakadesh *
public class HostEnergyConsumptionGauge {
    11 usages
    private Gauge gauge; // Declare a private Gauge instance


    // Constructor
    1 usage  prakadesh
    public HostEnergyConsumptionGauge() {
        // Initialize the Gauge instance and customize it
        gauge = new Gauge();
        gauge.setSkin(new ModernSkin(gauge));
        gauge.setMinValue(0);
        gauge.setMaxValue(10000);
        gauge.setMajorTickSpace(1000); // Set the interval to 100

        gauge.setUnit("Joules");

        gauge.setTitle("Energy Consumption of Host");
        gauge.setForegroundBaseColor(Color.WHITE);
    }
    1 usage  prakadesh
    public Gauge getGauge() { return gauge; }

    // Create a method to update the gauge's value
    1 usage  prakadesh
    public void updateValue(double value) { gauge.setValue(value); }

    
}
```



```java
package com.enerymonitoring.tool.enerymonitoring;

import java.io.IOException;
import java.io.OutputStream;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;

no usages  prakadesh
public class CustomOutputStream extends OutputStream {
    4 usages
    private final JTextArea textArea;

    no usages  prakadesh
    public CustomOutputStream(JTextArea textArea) {
        this.textArea = textArea;
    }

    prakadesh
    @Override
    public void write(int b) throws IOException {
        SwingUtilities.invokeLater(() -> {
            textArea.append(String.valueOf((char) b));
            textArea.setCaretPosition(textArea.getDocument().getLength());
        });
    }
}
```

We use CustomOutputStream this class plays a major role as this is responsible for displaying the output of the code to the GUI this will fetch the output from energymonitoringapplications and send it to TextArea in Graphical_User_Interface.

3.3 Project Outcome



Default

NO OF HOSTS: 2

Host CPU Cores: 8

Host RAM (MB): 2048

HOST Storage (MB): 20000

HOST Bandwidth (Mbps): 2000

NO OF VM: 4

VM CPU Cores: 4

NO OF Cloudlets: 8

Host CPU Cores: 2

VM RAM (MB): 512

VM Bandwidth (Mbps): 1000

VM Storage (MB): 10000

Cloudlet Length: 50000

Start Energy Monitoring

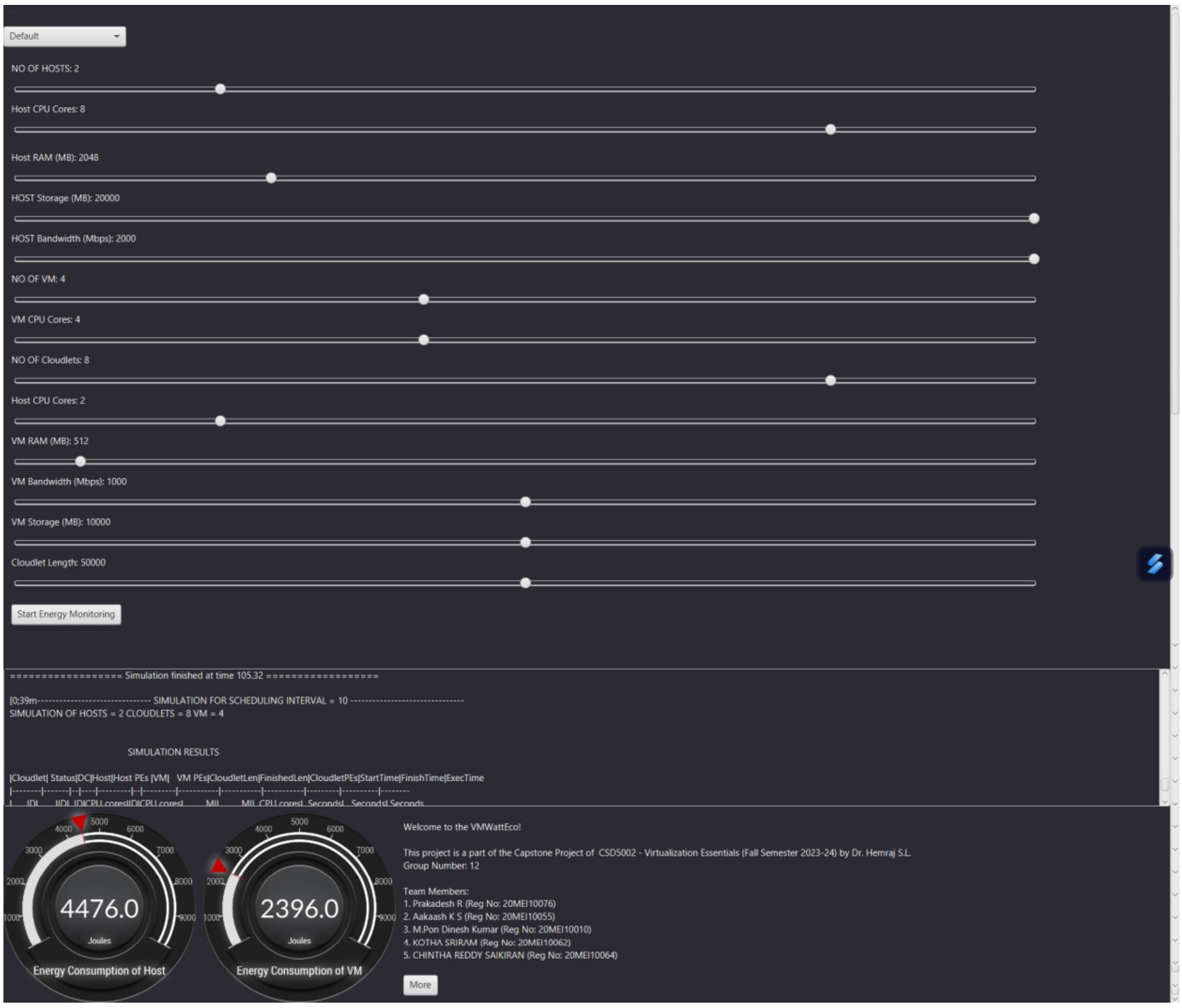
0.0 Joules
Energy Consumption of Host

0.0 Joules
Energy Consumption of VM

Welcome to the VMWattEco!

This project is a part of the Capstone Project of CSD5002 - Virtualization Essentials (Fall Semester 2023-24) by Dr. Hemraj S.L.
Group Number: 12

Team Members:
1. Prakadesh R (Reg No: 20MEI10076)
2. Aakaash K S (Reg No: 20MEI10055)
3. M.Pon Dinesh Kumar (Reg No: 20MEI10010)
4. KOTHA SRIRAM (Reg No: 20MEI10062)
5. CHINTHA REDDY SAIKIRAN (Reg No: 20MEI10064)



NO OF HOSTS: 2
Host CPU Cores: 8
Host RAM (MB): 2048
HOST Storage (MB): 20000
HOST Bandwidth (Mbps): 2000
NO OF VM: 4
VM CPU Cores: 4
NO OF Cloudlets: 8
Host CPU Cores: 2
VM RAM (MB): 512
VM Bandwidth (Mbps): 1000
VM Storage (MB): 10000
Cloudlet Length: 50000

```
================== Simulation finished at time 105.32 ==================

[0;39m------------------------------ SIMULATION FOR SCHEDULING INTERVAL = 10 ------------------------------
SIMULATION OF HOSTS = 2 CLOUDLETS = 8 VM = 4


                              SIMULATION RESULTS

|Cloudlet| Status|DC|Host|Host PEs |VM|   VM PEs|CloudletLen|FinishedLen|CloudletPEs|StartTime|FinishTime|ExecTime
|--------|-------|--|----|---------|--|---------|-----------|-----------|-----------|---------|----------|--------
|      ID|       |ID|  ID|CPU cores|ID|CPU cores|         MI|         MI|  CPU cores|  Seconds|   Seconds| Seconds
```

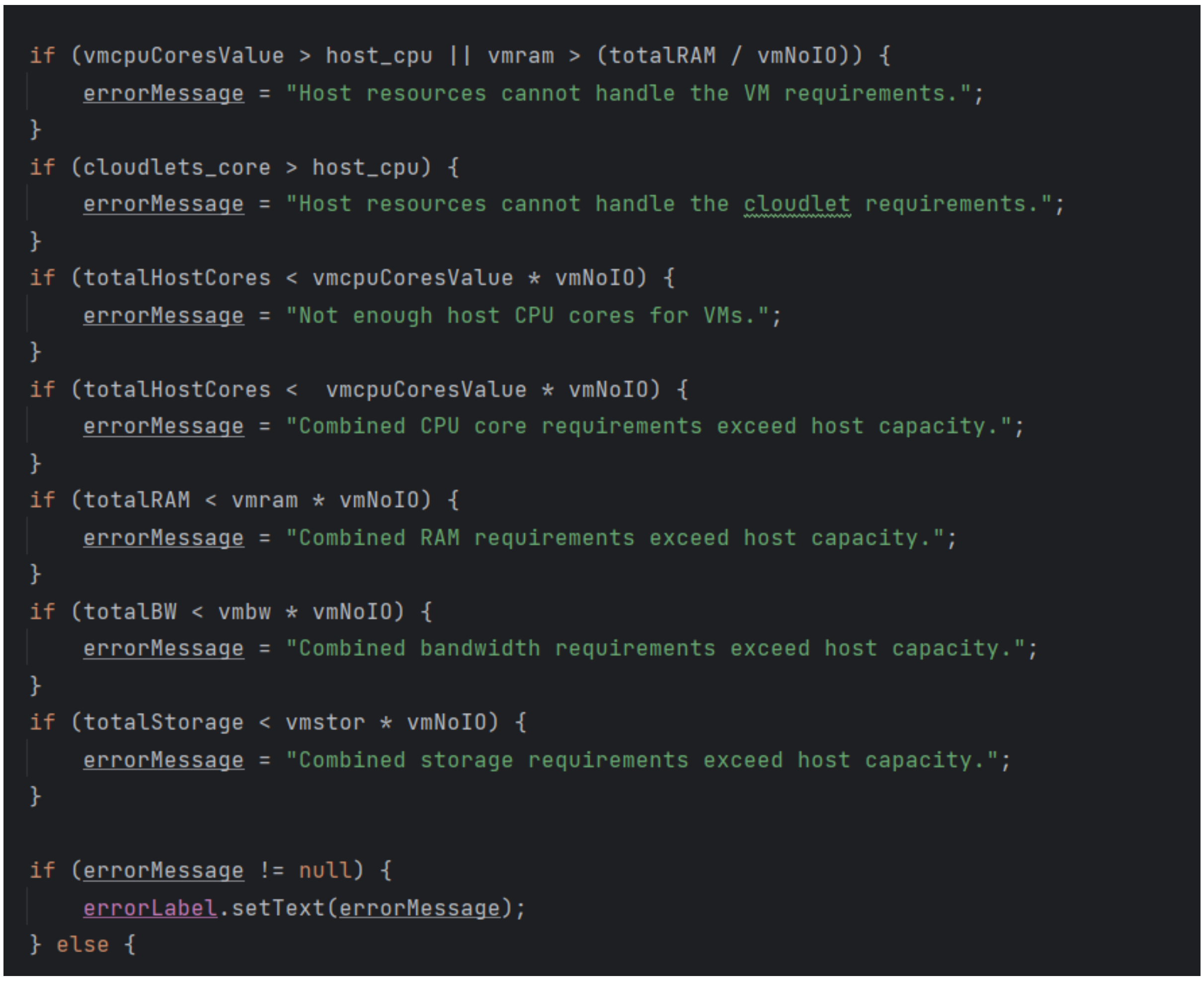
Welcome to the VMWattEco!

This project is a part of the Capstone Project of CSD5002 - Virtualization Essentials (Fall Semester 2023-24) by Dr. Hemraj S.L.
Group Number: 12

Team Members:
1. Prakadesh R (Reg No: 20MEI10076)
2. Aakaash K S (Reg No: 20MEI10055)
3. M.Pon Dinesh Kumar (Reg No: 20MEI10010)
4. KOTHA SRIRAM (Reg No: 20MEI10062)
5. CHINTHA REDDY SAIKIRAN (Reg No: 20MEI10064)

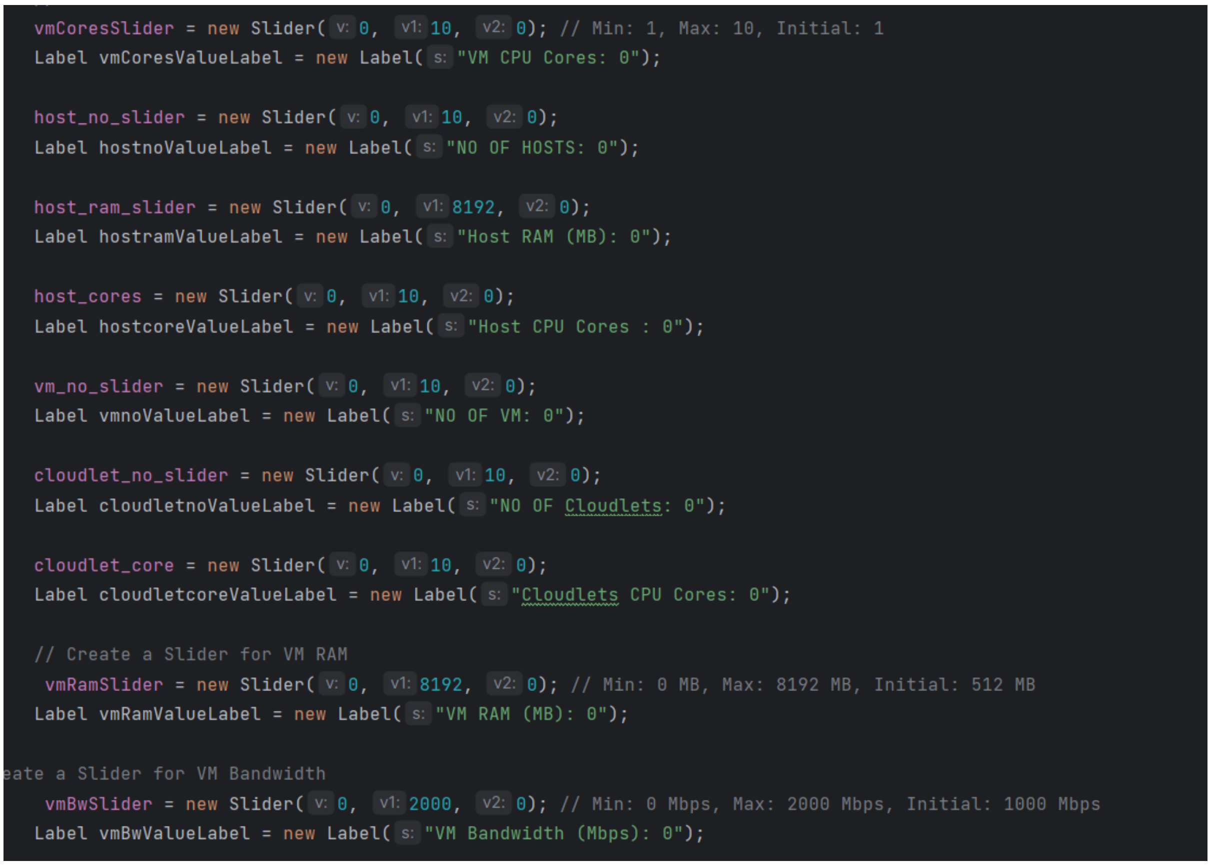
3.4 Test and validation

We will be using test case to test the application performance and error handling logics to ensure smooth operation of the application so that they can handle various issues a such as resource constraints issue where which will lead to condition where the application run in a loop which result in application crash which was major issue faced during the initial phase of the process to overcome that we had a condition to handle resource constraints issues.



```
if (vmcpuCoresValue > host_cpu || vmram > (totalRAM / vmNoIO)) {
    errorMessage = "Host resources cannot handle the VM requirements.";
}
if (cloudlets_core > host_cpu) {
    errorMessage = "Host resources cannot handle the cloudlet requirements.";
}
if (totalHostCores < vmcpuCoresValue * vmNoIO) {
    errorMessage = "Not enough host CPU cores for VMs.";
}
if (totalHostCores <  vmcpuCoresValue * vmNoIO) {
    errorMessage = "Combined CPU core requirements exceed host capacity.";
}
if (totalRAM < vmram * vmNoIO) {
    errorMessage = "Combined RAM requirements exceed host capacity.";
}
if (totalBW < vmbw * vmNoIO) {
    errorMessage = "Combined bandwidth requirements exceed host capacity.";
}
if (totalStorage < vmstor * vmNoIO) {
    errorMessage = "Combined storage requirements exceed host capacity.";
}

if (errorMessage != null) {
    errorLabel.setText(errorMessage);
} else {
```

These condition is helpful in prevention against various kinds of input based attacks such as condition where a user input letter instead of a integer which leads to crash we also made the decision of replacing the text field with a siders with a min and max values so that we can prevent these kind condition in future.

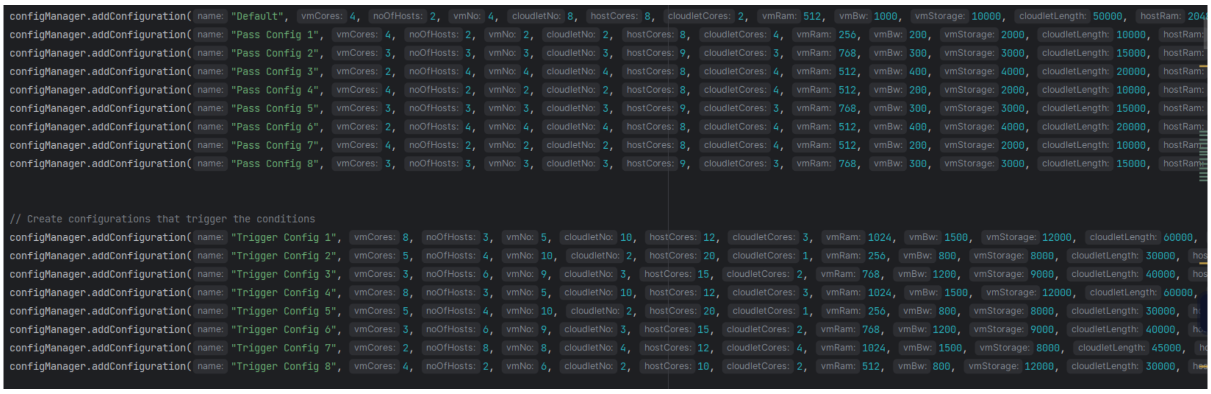


```
  vmCoresSlider = new Slider( v: 0,  v1: 10,  v2: 0); // Min: 1, Max: 10, Initial: 1
  Label vmCoresValueLabel = new Label( s: "VM CPU Cores: 0");

  host_no_slider = new Slider( v: 0,  v1: 10,  v2: 0);
  Label hostnoValueLabel = new Label( s: "NO OF HOSTS: 0");

  host_ram_slider = new Slider( v: 0,  v1: 8192,  v2: 0);
  Label hostramValueLabel = new Label( s: "Host RAM (MB): 0");

  host_cores = new Slider( v: 0,  v1: 10,  v2: 0);
  Label hostcoreValueLabel = new Label( s: "Host CPU Cores : 0");

  vm_no_slider = new Slider( v: 0,  v1: 10,  v2: 0);
  Label vmnoValueLabel = new Label( s: "NO OF VM: 0");

  cloudlet_no_slider = new Slider( v: 0,  v1: 10,  v2: 0);
  Label cloudletnoValueLabel = new Label( s: "NO OF Cloudlets: 0");

  cloudlet_core = new Slider( v: 0,  v1: 10,  v2: 0);
  Label cloudletcoreValueLabel = new Label( s: "Cloudlets CPU Cores: 0");

  // Create a Slider for VM RAM
   vmRamSlider = new Slider( v: 0,  v1: 8192,  v2: 0); // Min: 0 MB, Max: 8192 MB, Initial: 512 MB
  Label vmRamValueLabel = new Label( s: "VM RAM (MB): 0");

eate a Slider for VM Bandwidth
   vmBwSlider = new Slider( v: 0,  v1: 2000,  v2: 0); // Min: 0 Mbps, Max: 2000 Mbps, Initial: 1000 Mbps
  Label vmBwValueLabel = new Label( s: "VM Bandwidth (Mbps): 0");
```

Also, these sliders provide the user with an interactive way and helpful in securing the application.

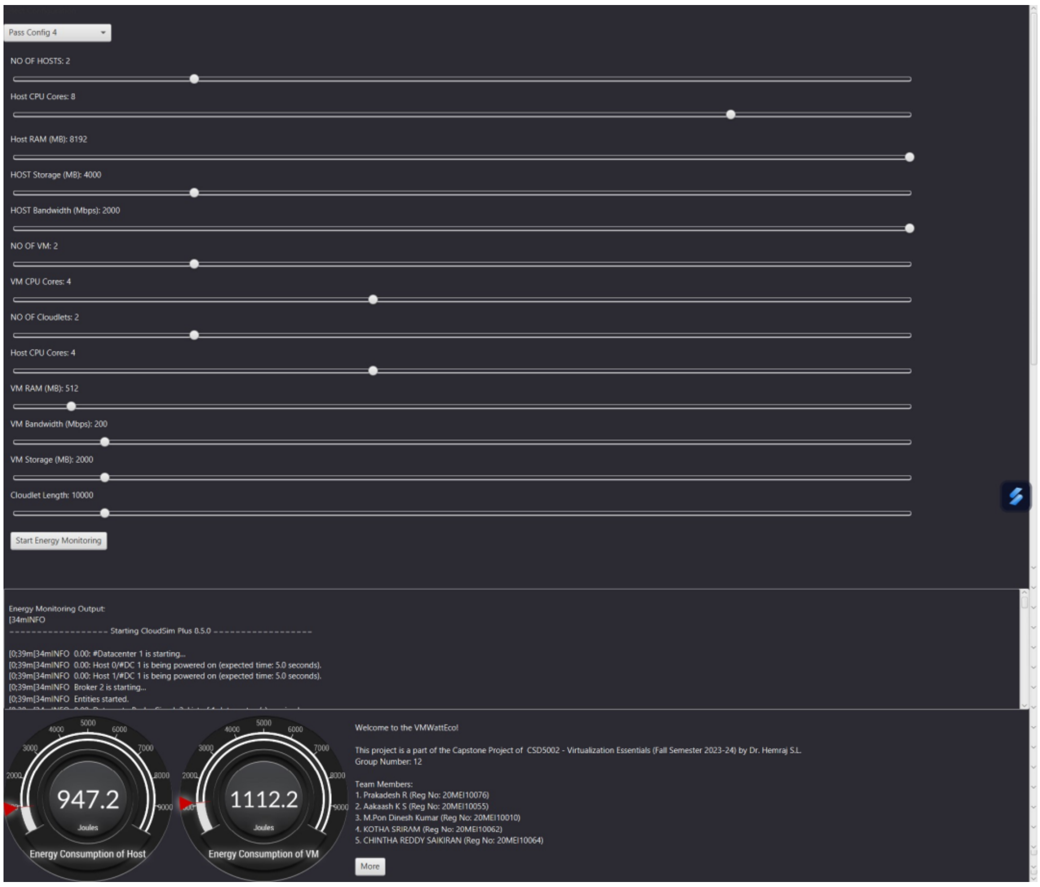
To the test the application whether these conditions are working properly and the application is working correctly we can use test case these cases are being pre-defined in the code for testing purposes using a configuration manager these configuration managers are providing with a combo box to show the list of available configurations to be used along with test configuration:

Here are some of the test configurations:



```
configManager.addConfiguration( name: "Default",  vmCores: 4,  noOfHosts: 2,  vmNo: 4,  cloudletNo: 8,  hostCores: 8,  cloudletCores: 2,  vmRam: 512,  vmBw: 1000,  vmStorage: 10000,  cloudletLength: 50000,  hostRam: 204
configManager.addConfiguration( name: "Pass Config 1",  vmCores: 4,  noOfHosts: 2,  vmNo: 2,  cloudletNo: 2,  hostCores: 8,  cloudletCores: 4,  vmRam: 256,  vmBw: 200,  vmStorage: 2000,  cloudletLength: 10000,  hostRam:
configManager.addConfiguration( name: "Pass Config 2",  vmCores: 3,  noOfHosts: 3,  vmNo: 3,  cloudletNo: 3,  hostCores: 9,  cloudletCores: 3,  vmRam: 768,  vmBw: 300,  vmStorage: 3000,  cloudletLength: 15000,  hostRam:
configManager.addConfiguration( name: "Pass Config 3",  vmCores: 2,  noOfHosts: 4,  vmNo: 4,  cloudletNo: 4,  hostCores: 8,  cloudletCores: 4,  vmRam: 512,  vmBw: 400,  vmStorage: 4000,  cloudletLength: 20000,  hostRam:
configManager.addConfiguration( name: "Pass Config 4",  vmCores: 4,  noOfHosts: 2,  vmNo: 2,  cloudletNo: 2,  hostCores: 8,  cloudletCores: 4,  vmRam: 512,  vmBw: 200,  vmStorage: 2000,  cloudletLength: 10000,  hostRam:
configManager.addConfiguration( name: "Pass Config 5",  vmCores: 3,  noOfHosts: 3,  vmNo: 3,  cloudletNo: 3,  hostCores: 9,  cloudletCores: 3,  vmRam: 768,  vmBw: 300,  vmStorage: 3000,  cloudletLength: 15000,  hostRam:
configManager.addConfiguration( name: "Pass Config 6",  vmCores: 2,  noOfHosts: 4,  vmNo: 4,  cloudletNo: 4,  hostCores: 8,  cloudletCores: 4,  vmRam: 512,  vmBw: 400,  vmStorage: 4000,  cloudletLength: 20000,  hostRam:
configManager.addConfiguration( name: "Pass Config 7",  vmCores: 4,  noOfHosts: 2,  vmNo: 2,  cloudletNo: 2,  hostCores: 8,  cloudletCores: 4,  vmRam: 512,  vmBw: 200,  vmStorage: 2000,  cloudletLength: 10000,  hostRam
configManager.addConfiguration( name: "Pass Config 8",  vmCores: 3,  noOfHosts: 3,  vmNo: 3,  cloudletNo: 3,  hostCores: 9,  cloudletCores: 3,  vmRam: 768,  vmBw: 300,  vmStorage: 3000,  cloudletLength: 15000,  hostRam


// Create configurations that trigger the conditions
configManager.addConfiguration( name: "Trigger Config 1",  vmCores: 8,  noOfHosts: 3,  vmNo: 5,  cloudletNo: 10,  hostCores: 12,  cloudletCores: 3,  vmRam: 1024,  vmBw: 1500,  vmStorage: 12000,  cloudletLength: 60000,
configManager.addConfiguration( name: "Trigger Config 2",  vmCores: 5,  noOfHosts: 4,  vmNo: 10,  cloudletNo: 2,  hostCores: 20,  cloudletCores: 1,  vmRam: 256,  vmBw: 800,  vmStorage: 8000,  cloudletLength: 30000,  hos
configManager.addConfiguration( name: "Trigger Config 3",  vmCores: 3,  noOfHosts: 6,  vmNo: 9,  cloudletNo: 3,  hostCores: 15,  cloudletCores: 2,  vmRam: 768,  vmBw: 1200,  vmStorage: 9000,  cloudletLength: 40000,  hos
configManager.addConfiguration( name: "Trigger Config 4",  vmCores: 8,  noOfHosts: 3,  vmNo: 5,  cloudletNo: 10,  hostCores: 12,  cloudletCores: 3,  vmRam: 1024,  vmBw: 1500,  vmStorage: 12000,  cloudletLength: 60000,
configManager.addConfiguration( name: "Trigger Config 5",  vmCores: 5,  noOfHosts: 4,  vmNo: 10,  cloudletNo: 2,  hostCores: 20,  cloudletCores: 1,  vmRam: 256,  vmBw: 800,  vmStorage: 8000,  cloudletLength: 30000,  h
configManager.addConfiguration( name: "Trigger Config 6",  vmCores: 3,  noOfHosts: 6,  vmNo: 9,  cloudletNo: 3,  hostCores: 15,  cloudletCores: 2,  vmRam: 768,  vmBw: 1200,  vmStorage: 9000,  cloudletLength: 40000,  h
configManager.addConfiguration( name: "Trigger Config 7",  vmCores: 2,  noOfHosts: 8,  vmNo: 8,  cloudletNo: 4,  hostCores: 12,  cloudletCores: 4,  vmRam: 1024,  vmBw: 1500,  vmStorage: 8000,  cloudletLength: 45000,  h
configManager.addConfiguration( name: "Trigger Config 8",  vmCores: 4,  noOfHosts: 2,  vmNo: 6,  cloudletNo: 2,  hostCores: 10,  cloudletCores: 2,  vmRam: 512,  vmBw: 800,  vmStorage: 12000,  cloudletLength: 30000,  h
```

Let’s take a look at some output of the above configuration Pass Config 4



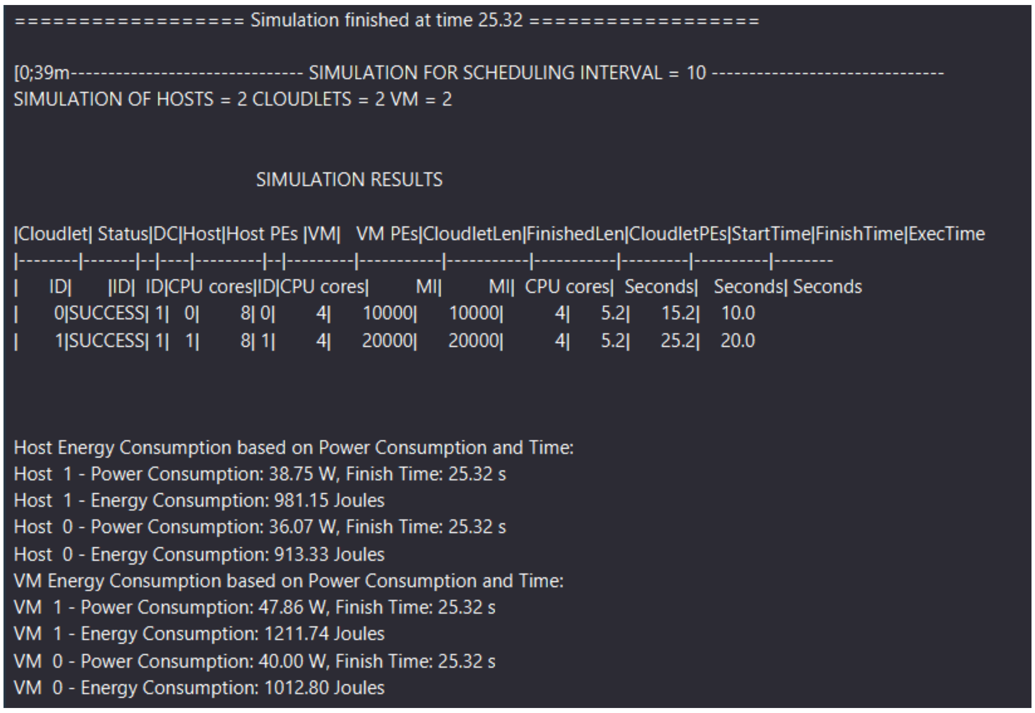
NO OF HOSTS: 2
Host CPU Cores: 8
Host RAM (MB): 8192
HOST Storage (MB): 4000
HOST Bandwidth (Mbps): 2000
NO OF VM: 2
VM CPU Cores: 4
NO OF Cloudlets: 2
Host CPU Cores: 4
VM RAM (MB): 512
VM Bandwidth (Mbps): 200
VM Storage (MB): 2000
Cloudlet Length: 10000

```
Energy Monitoring Output:
[34mINFO
------------------ Starting CloudSim Plus 8.5.0 ------------------

[0;39m[34mINFO  0.00: #Datacenter 1 is starting...
[0;39m[34mINFO  0.00: Host 0/#DC 1 is being powered on (expected time: 5.0 seconds).
[0;39m[34mINFO  0.00: Host 1/#DC 1 is being powered on (expected time: 5.0 seconds).
[0;39m[34mINFO  Broker 2 is starting...
[0;39m[34mINFO  Entities started.
```

Welcome to the VMWattEco!

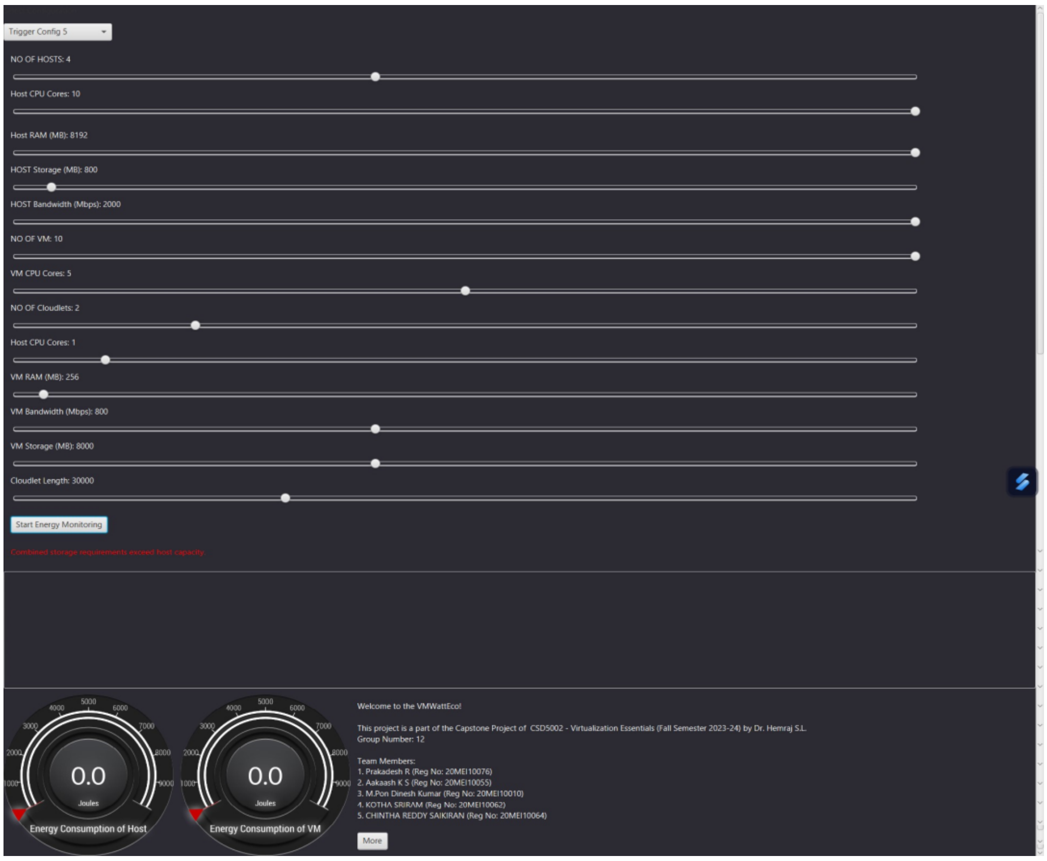
This project is a part of the Capstone Project of CSD5002 - Virtualization Essentials (Fall Semester 2023-24) by Dr. Hemraj S.L.
Group Number: 12

Team Members:
1. Prakadesh R (Reg No: 20MEI10076)
2. Aakaash K S (Reg No: 20MEI10055)
3. M.Pon Dinesh Kumar (Reg No: 20MEI10010)
4. KOTHA SRIRAM (Reg No: 20MEI10062)
5. CHINTHA REDDY SAIKIRAN (Reg No: 20MEI10064)



```
================== Simulation finished at time 25.32 ==================

[0;39m------------------------------ SIMULATION FOR SCHEDULING INTERVAL = 10 ------------------------------
SIMULATION OF HOSTS = 2 CLOUDLETS = 2 VM = 2


                              SIMULATION RESULTS

|Cloudlet| Status|DC|Host|Host PEs |VM|   VM PEs|CloudletLen|FinishedLen|CloudletPEs|StartTime|FinishTime|ExecTime
|--------|-------|--|----|---------|--|---------|-----------|-----------|-----------|---------|----------|--------
|      ID|       |ID|  ID|CPU cores|ID|CPU cores|         MI|         MI|  CPU cores|  Seconds|   Seconds| Seconds
|       0|SUCCESS| 1|   0|        8| 0|        4|      10000|      10000|          4|      5.2|      15.2|   10.0
|       1|SUCCESS| 1|   1|        8| 1|        4|      20000|      20000|          4|      5.2|      25.2|   20.0



Host Energy Consumption based on Power Consumption and Time:
Host  1 - Power Consumption: 38.75 W, Finish Time: 25.32 s
Host  1 - Energy Consumption: 981.15 Joules
Host  0 - Power Consumption: 36.07 W, Finish Time: 25.32 s
Host  0 - Energy Consumption: 913.33 Joules
VM Energy Consumption based on Power Consumption and Time:
VM  1 - Power Consumption: 47.86 W, Finish Time: 25.32 s
VM  1 - Energy Consumption: 1211.74 Joules
VM  0 - Power Consumption: 40.00 W, Finish Time: 25.32 s
VM  0 - Energy Consumption: 1012.80 Joules
```

Trigger Config 5;


Trigger Config 5

NO OF HOSTS: 4

Host CPU Cores: 10

Host RAM (MB): 8192

HOST Storage (MB): 800

HOST Bandwidth (Mbps): 2000

NO OF VM: 10

VM CPU Cores: 5

NO OF Cloudlets: 2

Host CPU Cores: 1

VM RAM (MB): 256

VM Bandwidth (Mbps): 800

VM Storage (MB): 8000

Cloudlet Length: 30000

Start Energy Monitoring

Combined storage requirements exceed host capacity.

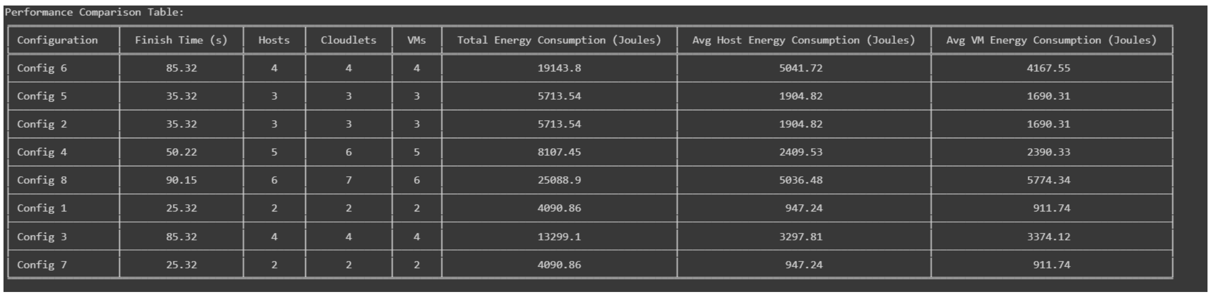
Welcome to the VMWattEco!

This project is a part of the Capstone Project of CSD5002 - Virtualization Essentials (Fall Semester 2023-24) by Dr. Hemraj S.L.
Group Number: 12

Team Members:
1. Prakadesh R (Reg No: 20MEI10076)
2. Aakaash K S (Reg No: 20MEI10055)
3. M.Pon Dinesh Kumar (Reg No: 20MEI10010)
4. KOTHA SRIRAM (Reg No: 20MEI10062)
5. CHINTHA REDDY SAIKIRAN (Reg No: 20MEI10064)

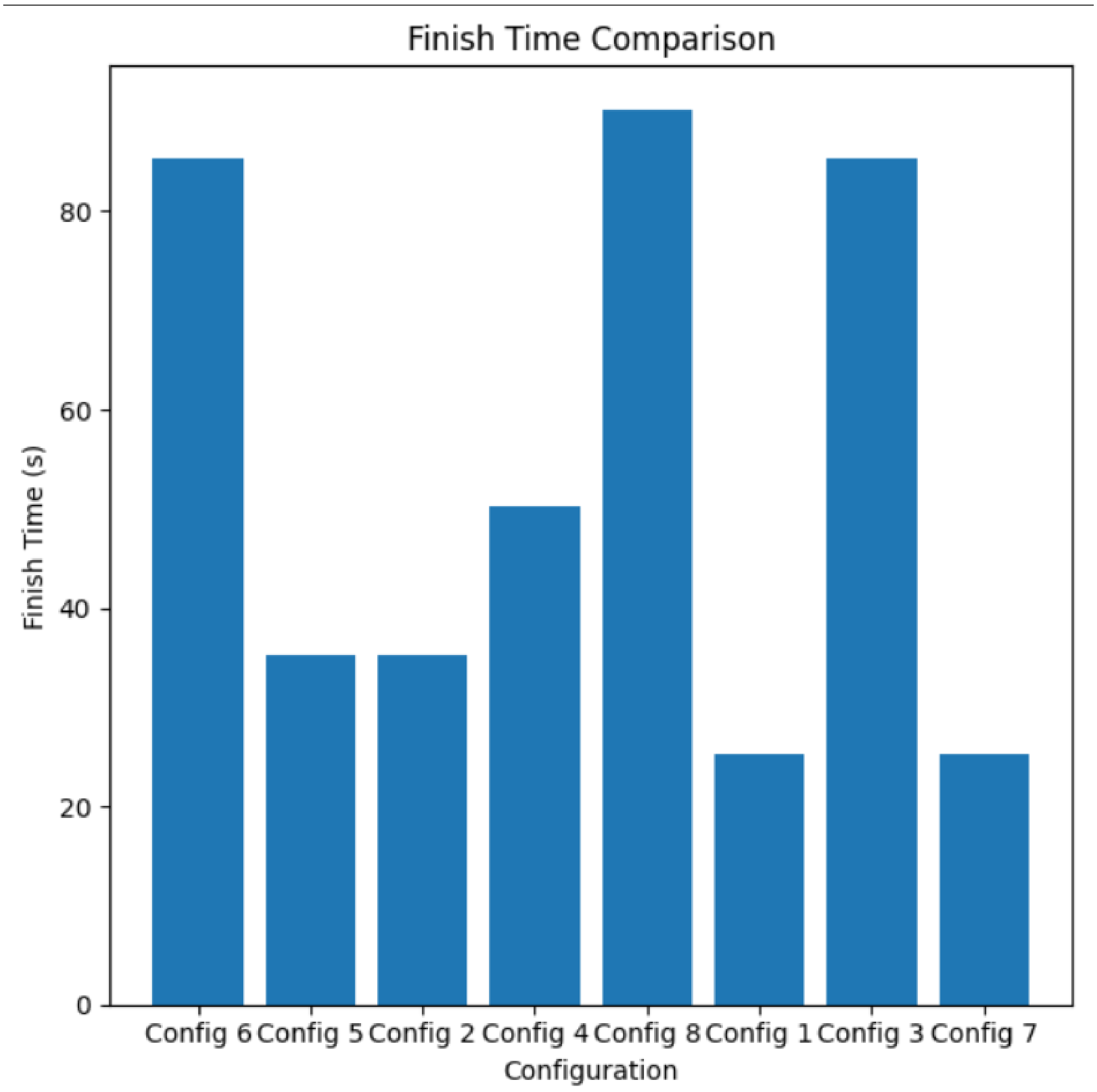
3.5 Performance Analysis

To analyse the performance of the application we are using pre-configured configuration from configuration manager then we compare them in graphical form for better understanding.


```
Performance Comparison Table:
| Configuration | Finish Time (s) | Hosts | Cloudlets | VMs | Total Energy Consumption (Joules) | Avg Host Energy Consumption (Joules) | Avg VM Energy Consumption (Joules) |
| Config 6      |      85.32      |   4   |     4     |  4  |              19143.8              |                5041.72               |              4167.55               |
| Config 5      |      35.32      |   3   |     3     |  3  |              5713.54              |                1904.82               |              1690.31               |
| Config 2      |      35.32      |   3   |     3     |  3  |              5713.54              |                1904.82               |              1690.31               |
| Config 4      |      50.22      |   5   |     6     |  5  |              8107.45              |                2409.53               |              2390.33               |
| Config 8      |      90.15      |   6   |     7     |  6  |              25088.9              |                5036.48               |              5774.34               |
| Config 1      |      25.32      |   2   |     2     |  2  |              4090.86              |                 947.24               |               911.74               |
| Config 3      |      85.32      |   4   |     4     |  4  |              13299.1              |                3297.81               |              3374.12               |
| Config 7      |      25.32      |   2   |     2     |  2  |              4090.86              |                 947.24               |               911.74               |
```

1. Finish Time Comparison:

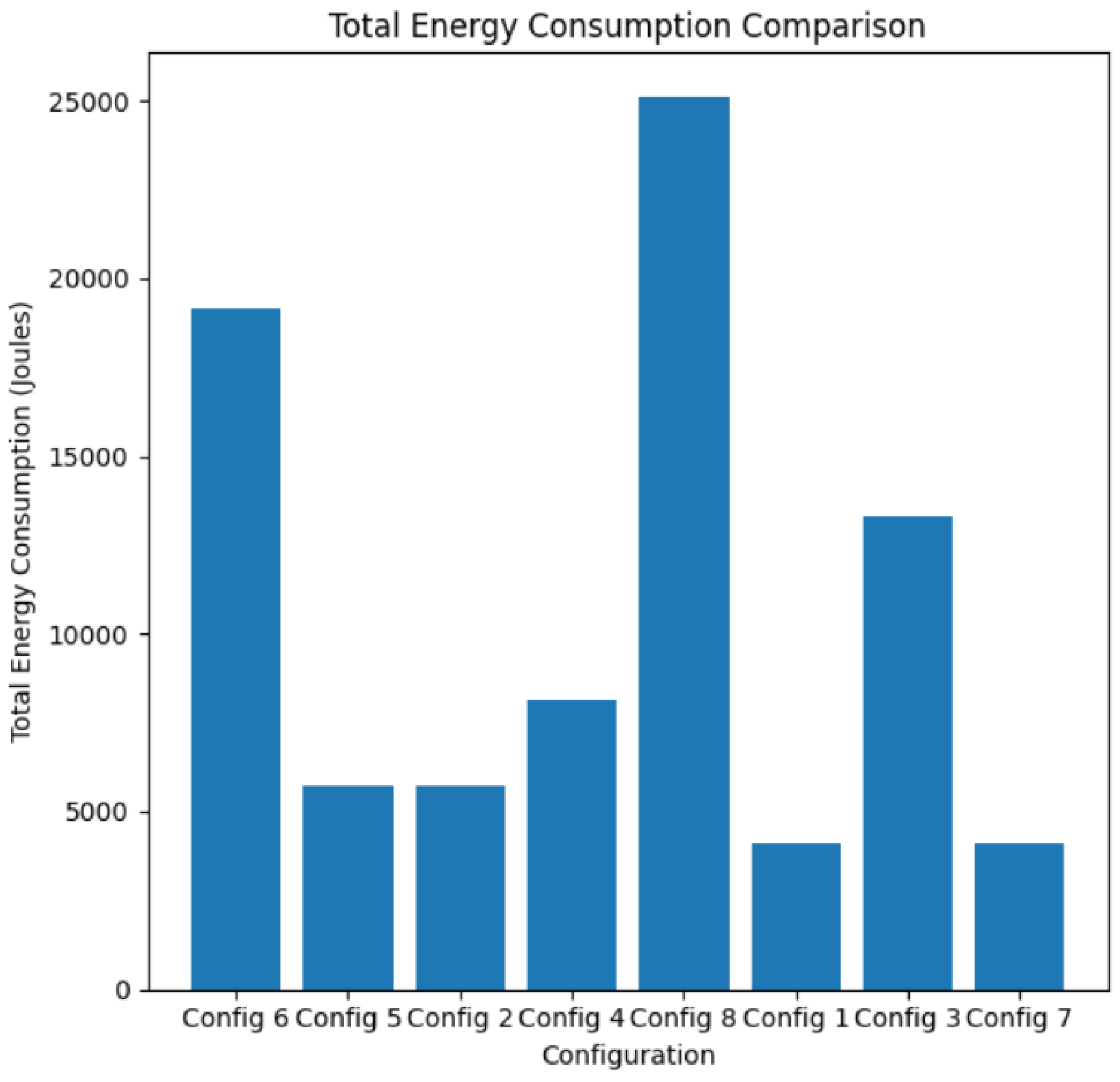
We started by comparing the finish times for each configuration. Which determines how much time it take for the simulation completes the cloudlets tasks.



From our observation, we can see that Config 6 and Config 8 have the longest finish times, indicating that these configurations require more time to complete the simulations. Conversely, Config 5 and Config 1 have shorter finish times, indicating more efficient task completion.

2. Total Energy Consumption Comparison:

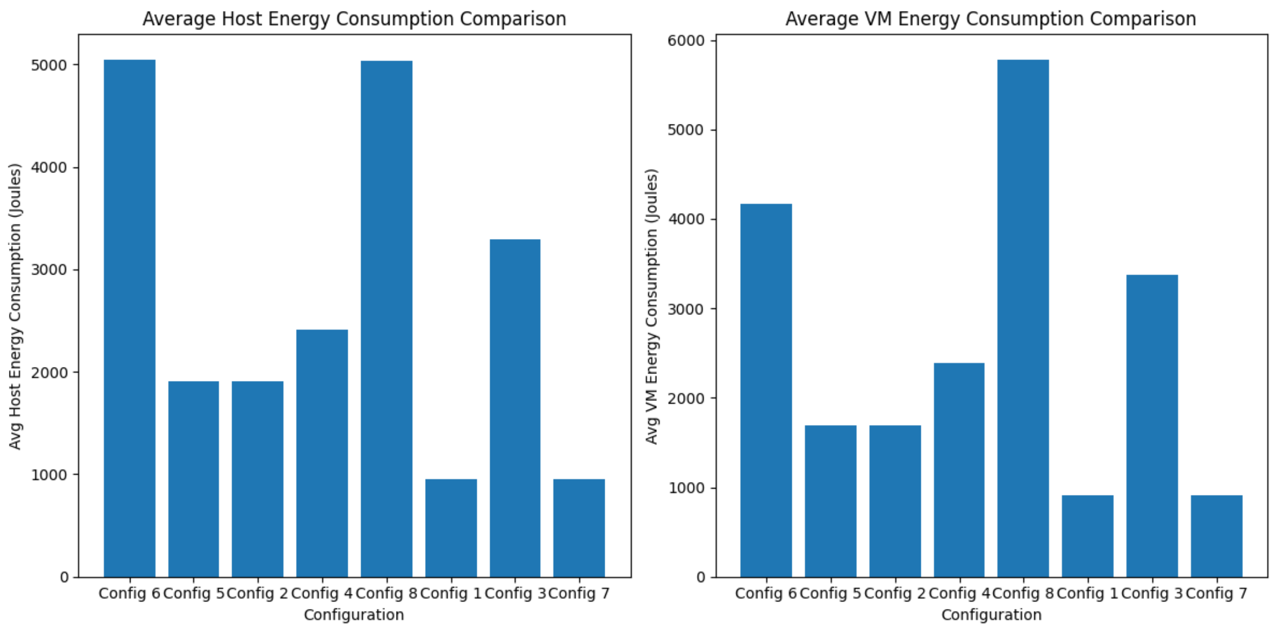
Next, we go with the total energy consumption for each configuration. Energy consumption is a energy used by the configuration in a simulator.



Config 6 and Config 8 have the highest total energy consumption, indicating a higher energy demand for these configurations. In contrast, Config 5 and Config 1 exhibit lower energy consumption, making them more energy-efficient.

3. Average Host and VM Energy Consumption:

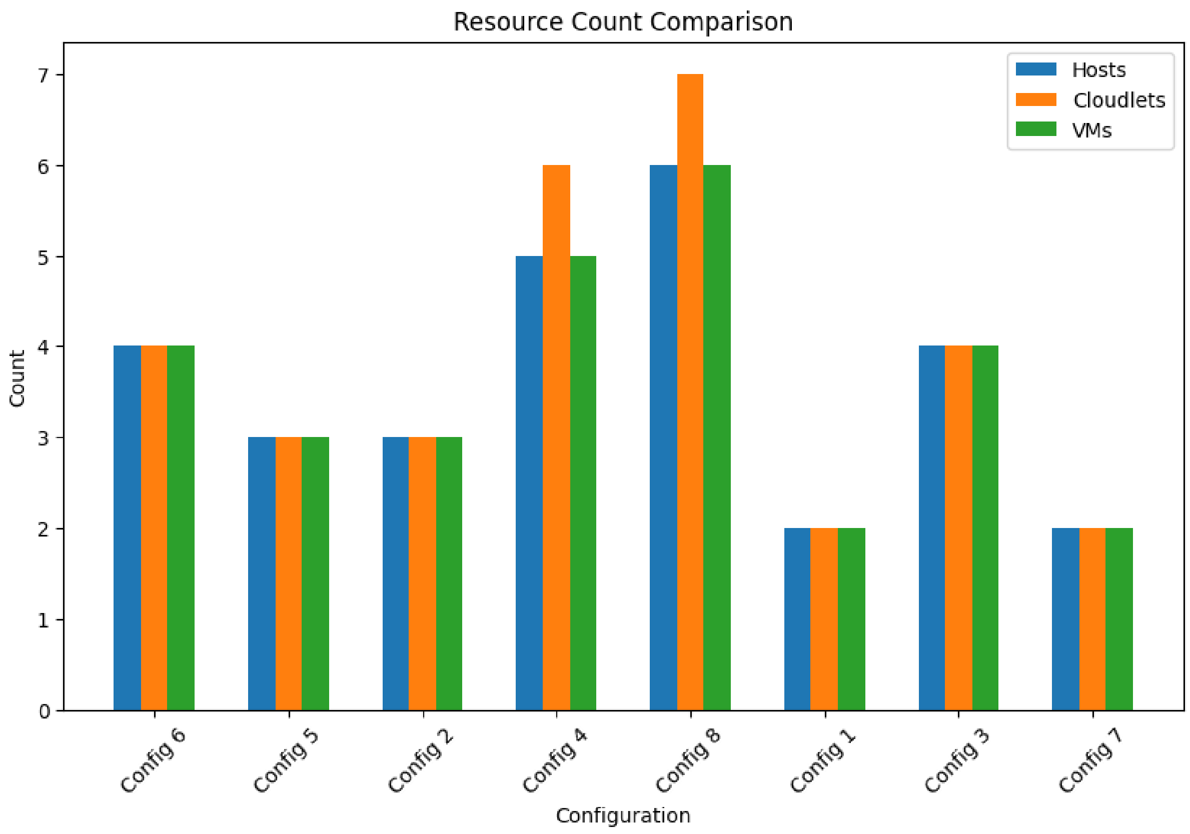
Next we have energy consumption on the basis of VM and host in a configuration in last section we have seen the combination of energy consumption here we discuss on the average energy consumption of host and VM energy consumption.



The chart illustrates that Config 6 and Config 8 have the highest average host and VM energy consumption. On the other hand, Config 5 and Config 1 are more efficient in terms of energy usage.

4. Resource Comparison:

In this we will compare each configuration along with there no of hosts , cloudlets and Virtual Machines.



Config 6 and Config 8 require a higher number of hosts, which can be crucial in assessing resource scalability and cost-effectiveness.

In summary, the performance analysis reveals that Config 5 and Config 1 outperform other configurations with shorter finish times, lower energy consumption, and efficient resource usage. Config 6 and Config 8, while achieving the desired simulation goals, come at the cost of increased finish times, energy consumption, and higher host requirements.

Chapter 4
CONCLUSION AND RECOMMENDATIONS

4.1 Outline

We had successfully developed a system capable of recording and analysing the real-time data relating to power usage, resource utilization, and other important factors in a data centre or server environment.

While developing the energy consumption monitoring tool for virtual machines (VMs), the programme offers the administrators useful insights into energy usage patterns and ideas for optimizing the energy use, which can lead to cost savings and improved environmental sustainability.

4.2 Limitation/Constraints

1) While our energy consumption and monitoring solution for virtual machines has many advantages and it is very crucial to understand the limitations and constraints. Initially, the system is dependent on the installation of hardware sensors which, while necessary for real-time data collection, can be an expensive and infrastructure-altering endeavor.

2) Furthermore, the compatibility of the tool is an issue because its integration strongly depends on the specific virtualization platforms and hardware configurations. Thus, restricting its applicability in heterogeneous environments.

3) The accuracy of energy consumption estimates may also be impacted by the fluctuations in sensor data and complicate the structure of the virtualized systems.

4) Security issues including data protection and access control must be vigilantly addressed.

5) Finally, scalability is an ongoing issue, especially as the number of VMs and sensors grows and demands a dynamic scaling method for effective management and performance.

6) Constraints highlight the areas where the tool's development and implementation should be improved.

4.3 Future Enhancements

- In the future plan, we want to improve our energy monitoring tool by finding more affordable ways to measure energy consumption, so that it does not cost too much.
- We also make sure that our tool is compatible to a variety of computer systems and not just to specific systems.
- We use the computer information to help us to predict and recommend better ways to save energy.
- We will also make sure that our tool is more accurate in numbers. We want to make it very easy to use.
- By doing these changes we can save electricity, money and the environment.

References

[1] P. R. Theja and S. K. K. Babu, “An Evolutionary Computing based Energy Efficient VM Consolidation Scheme for Optimal Resource Utilization and QoS Assurance”, Indian Journal of Science and Technology.

[2] M.-H. Malekloo, N. Kara, and M. El Barachi, “An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments", Sustain. Comput., Information.

[3] M. Soltanshahi, R. Asemi, and N. Shafiei, “Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers", Heliyon,

[4] H. Wang and H. Tianfield, “Energy-aware dynamic virtual machine consolidation for cloud datacenters", IEEE Access.

[5] M. Mohammadhosseini, A. Toroghi Haghighat, and E. Mahdipour, “An efficient energy-aware method for virtual machine placement in cloud data centers using the cultural algorithm", J. Supercomputer.

[6] N. Garg, D. Singh, and M. S. Goraya, “Power and resource-aware VM placement in cloud environment", in Proc. IEEE 8th Int. Advance Computing.Conference. (IACC).

[7] Liu, Xiao-Fang, et al. “An energy efficient ant colony system for virtual machine placement in cloud computing”, IEEE Transactions on Evolutionary Computation.

[8] An-ping Xiong and Chun-xiang Xu, “Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center”, Mathematical Problems in Engineering.

[9] Abdessamia, Y. Tai, W. Zhang, and M. Shafiq, “An Improved particle swarm optimization for energy-efficiency virtual machine placement”, Published in IEEE International Conference on Cloud Computing Research and Innovation (ICCCRI).

[10] Xiong Fu, Qing Zhao, Junchang Wang, Lin Zhang, and Lei Qiao,“Energy-aware VM initial placement strategy based on BPSO in cloud computing”, Scientific Programming.

Acknowledgements

We would like to extend our heartfelt gratitude to the following individuals and organizations for their invaluable contributions and support in the development of our energy consumption monitoring tool for VMs:

Our academic advisor**, Dr. Hemraj S. Lamkuche** provided mentorship, guidance, and unwavering support throughout this research. His expertise and insights were critical in shaping the direction of this project.

We extend our appreciation to the dedicated members of our research team who tirelessly worked on the tool's development. Their passion and collaborative spirit were fundamental to the creation of this project.

VIT Bhopal University
October 25, 2023