National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Savitanshu Somvanshi
Student ID: 23213949

School of Computing
National College of Ireland

Supervisor: Prof. Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Savitanshu Somvanshi
Student ID:	23213949
Programme:	Masters in Cloud Computing **Year:** 2024
Module:	MSc Research Project
Lecturer:	Prof. Vikas Sahni
Submission Due Date:	11/08/2025
Project Title:	Processing-in-Memory (PIM) for Cloud Computing: Optimizing Performance and Energy Efficiency
Word Count:	1108 **Page Count:** 5

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: Savitanshu Somvanshi

Date: 11th August 2025

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

Savitanshu Somvanshi
Student ID: 23213949

1 Software Requirements

This section outlines the exact software tools and environments required to replicate the simulation-based experiments performed in this research.

1.1 Java Development Environment

Java Development Kit (JDK)

- Required Version: Java SE 8
- Why: CloudSim 3.0.3 is only compatible with Java 8. Later versions introduce breaking changes in API usage.
- Verify Installation: In bash terminal hit : “java -version” . Should output something like → java version "1.8.0_352"

1.2 CloudSim Simulation Toolkit

- **Toolkit: CloudSim 3.0.3**
- Source: https://github.com/Cloudslab/cloudsim
- Installation Steps:
  1. Clone the repository or download the ZIP.
  2. Extract or open the project folder.
  3. Ensure the following directories exist:
     a) /sources/org/cloudbus/cloudsim/
     b) /examples/org/cloudbus/cloudsim/examples/

1.3 Project file Modifications

This section explains how to set up the custom files and folder structure inside the CloudSim toolkit to enable the heterogeneous CPU+PIM scheduler simulation. This step is essential for reproducing the experiment accurately.

1.3.1 Create Custom scheduler package

- Inside the sources/ directory of CloudSim, navigate to org/ and create a new folder named scheduler.
- Inside the scheduler/ folder, create the following new Java files:
- PIMScheduler.java
- BaselineScheduler.java

These two files implement the logic for dynamic (adaptive) and static (fixed threshold) job scheduling, respectively.

1.3.2 Add PIM Host Configuration File

- Inside sources/org/, create a new folder named pimsim.
- Inside pimsim/, create one file: HeterogeneousHostConfig.java

This file defines the creation of both CPU and PIM hosts used in the simulation. It returns list of pre-configured Host objects for use in the datacenter setup.

1.3.3 Modify Main Simulation File

- Navigate to: examples/org/cloudbus/cloudsim/examples/
- Inside this folder, open (or create if not already present): CloudSimExample1.java
- Replace or modify the code to include: Import statements for scheduler.PIMScheduler, scheduler.BaselineScheduler, and pimsim.HeterogeneousHostConfig.
- A simulation logic that:
    - Creates 100 randomized cloudlets with RAM, length, and deadline values.
    - Assigns cloudlets to VMs based on either the dynamic or static scheduler (toggle available).
    - Writes simulation results to results_dynamic.csv or results_static.csv.

This is the entry point for the simulation. It contains the toggle for switching between the two schedulers and drives all VM/job setup and data export logic.

By the end of all the operations the folder structure should look like below

```
CloudSim/
├── sources/
│   ├── org/
│   │   ├── cloudbus/
│   │   │   └── cloudsim/            ← (default CloudSim code)
│   │   ├── scheduler/               ← (you create this)
│   │   │   ├── PIMScheduler.java
│   │   │   └── BaselineScheduler.java
│   │   └── pimsim/                  ← (you create this)
│   │       └── HeterogeneousHostConfig.java
├── examples/
│   ├── org/cloudbus/cloudsim/examples/
│   │   └── CloudSimExample1.java    ← (modified to run experiment)
```

2 Compilation and Execution Instructions

This section outlines how to compile and run the CloudSim simulation after all custom files and logic have been set up. Follow these steps carefully to ensure a successful execution.

- Open a terminal and navigate to the root directory of the CloudSim project using: cd /path/to/CloudSim
- Make sure that the jars/ directory (included by default in CloudSim) contains the following essential .jar files:

    - cloudsim-3.0.jar
    - commons-math3-3.6.1.jar
    - log4j-1.2.17.jar

- Must also ensure that your newly added scheduler/ and pimsim/ packages are located **under sources/org/** — this ensures they are compiled with the rest of the project.
- Compile all Java source files (including your custom ones) using: find sources/ -name "*.java" > sources_file_list.txt javac -cp jars/cloudsim-3.0.jar:jars/commons-math3-3.6.1.jar:jars/log4j-1.2.17.jar @sources_file_list.txt
- After compilation is successful, run the simulation using: java -cp jars/cloudsim-3.0.jar:jars/commons-math3-3.6.1.jar:jars/log4j-1.2.17.jar:sources/org.cloudbus.cloudsim.examples.CloudSimExample1
- If you face a ClassNotFoundException for any scheduler or custom package class, double-check:
    - That the Java file is placed inside the correct folder (matching its declared package).
    - That the -cp path includes sources/
- Upon successful execution, the terminal will print logs of cloudlet assignments, profiling decisions, predicted vs. actual execution times, and threshold adjustments (if dynamic scheduler is used).
- After the simulation completes, the following files will be automatically generated in the project root:
    - results_dynamic.csv (if dynamic scheduling is enabled)
    - results_static.csv (if static baseline is used)

3. Data Analysis and Graph Generation

This section outlines the steps to analyze simulation outputs and generate evaluation graphs using Google Colab. These graphs include threshold variation, prediction error trends, latency and energy comparisons, and box plots.

- Open Google Colab in your browser.
- Create a new notebook and rename it appropriately (e.g., PIM_Scheduler_Analysis.ipynb).
- In the first cell, import necessary Python libraries:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set(style="whitegrid")
```

- In the next cell, upload the output CSV files generated by CloudSim:

```
from google.colab import files

uploaded = files.upload()  # Upload both results_dynamic.csv and results_static.csv
```

- Once uploaded, load both files into pandas DataFrames:

dynamic_df = pd.read_csv("results_dynamic.csv")

static_df = pd.read_csv("results_static.csv")

- Tag each dataset with its scheduler type:

  dynamic_df['Scheduler'] = 'Dynamic'

  static_df['Scheduler'] = 'Static'

- Combine both into one dataframe for comparative analysis:

  combined_df = pd.concat([dynamic_df, static_df])

3.1 To generate the various graphs:

- **Threshold Variation Over Time :**

```
thresholds = dynamic_df[["CloudletID", "Threshold"]].drop_duplicates()

plt.figure(figsize=(10, 5))
plt.plot(thresholds["CloudletID"], thresholds["Threshold"], marker='o')
plt.title("Threshold Evolution Over Cloudlet Executions")
plt.xlabel("Cloudlet ID")
plt.ylabel("Threshold Value")
plt.grid(True)
plt.show()
```

- **Prediction Error Trend**

```
plt.figure(figsize=(10, 5))
plt.plot(dynamic_df["CloudletID"], dynamic_df["Error"], marker='o', color='orange')
plt.title("Prediction Error Over Time (Dynamic Scheduler)")
plt.xlabel("Cloudlet ID")
plt.ylabel("Prediction Error")
plt.grid(True)
plt.show()
```

- **Average Latency Comparison**

```
avg_latency = combined_df.groupby("Scheduler")["ActualTime"].mean()

avg_latency.plot(kind="bar", color=["skyblue", "salmon"])
plt.title("Average Latency: Static vs Dynamic")
plt.ylabel("Execution Time (s)")
plt.xticks(rotation=0)
plt.show()
```

- **Total Energy Comparison**

```
total_energy = combined_df.groupby("Scheduler")["Energy"].sum()

total_energy.plot(kind="bar", color=["green", "red"])
plt.title("Total Energy Consumption: Static vs Dynamic")
plt.ylabel("Total Energy (Joules)")
plt.xticks(rotation=0)
plt.show()
```

- **Latency Distribution**

```
sns.boxplot(data=combined_df, x="Scheduler", y="ActualTime", palette="pastel")
plt.title("Latency Distribution: Static vs Dynamic")
plt.ylabel("Execution Time (s)")
plt.show()
```

- **Prediction Error Distribution**

```
sns.boxplot(data=combined_df, x="Scheduler", y="Error", palette="muted")
plt.title("Prediction Error Distribution")
plt.ylabel("Prediction Error")
plt.show()
```

4. Assumptions

Files to Create/Modify

- CloudSimExample1.java: Main simulation file inside examples/org/cloudbus/cloudsim/examples
- PIMScheduler.java: Custom dynamic scheduler logic
- BaselineScheduler.java: Static scheduler logic
- HeterogeneousHostConfig.java: Adds CPU and PIM host configurations
- results_dynamic.csv & results_static.csv: Generated automatically after simulation ends
- Google Colab notebook for graph generation

Minimum VM Configuration Assumptions

- Minimum VM Configuration Assumptions
- CPU VM: MIPS ≥ 9000
- PIM VM: MIPS < 9000
- Scheduler Logic: Classifies cloudlets using RAM/Length ratio + deadline

Synthetic Cloudlet Parameters

- Number of Cloudlets: 100
- Length Range: 1000 to 500000 MI
- RAM Range: 256 MB to 4096 MB
- Deadline: Randomized between 10s to 100s
- Profiling: 10% simulation with noise between ±20% for prediction