National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Roshin Philip
Student ID: 23214759

School of Computing
National College of Ireland

Supervisor: Prof. Punit Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Roshin Philip
Student ID:	23214759
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof. Punit Gupta
Submission Due Date:	11/07/2025
Project Title:	Configuration Manual
Word Count:	1,685
Page Count:	7

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:	Roshin Philip
Date:	10th 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

Roshin Philip
23214759

1 Software Requirements

1.1 Java Development Kit

- **Required Version:** JDK 21.0.7 (Oracle Corporation (2024)
- **Purpose:** Required to compile and run CloudSim simulation code.
- **Verification:** Run java -version in your terminal to ensure the installed version matches.

1.2 CloudSim Toolkit

- **Version:** CloudSim 3.0.3 (CloudSim Project (2023))
- **Source:** https://github.com/Cloudslab/cloudsim
- **Setup:**
  - Download and extract the source code.
  - Ensure the following directories are present:
    - sources/org/cloudbus/cloudsim/
    - examples/org/cloudbus/cloudsim/examples/

2 Simulation Files and Folder Structure

Six customized Java simulation files are used to model different compute configurations (CPU, GPU, PPU) under both zero-delay and with-delay conditions:

All files are placed inside:

`examples/org/cloudbus/cloudsim/examples/custom/`

Each file defines the datacenter, VM, workload configuration, and prints compute + transfer time breakdowns (Data-in, Compute, Data-out, Total).

Device	**Zero-Delay File**	**With-Delay File**
CPU	CpuDatacenterZeroDelay.java	CpuDatacenterWithDelay.java
GPU	GpuDatacenterZeroDelay.java	GpuDatacenterExample.java
PPU	PpuDatacenterZeroDelay.java	PpuDatacenterWithDelay.java

3 Implementation Summary

- **VM and Host Configurations:**
  - CPU: 5,000,000 MIPS
  - GPU: 15,000,000 MIPS (proxy for NVIDIA V100)
  - PPU: 138,000,000 MIPS (based on Chen and et al. (2023) 9.2× speedup)
- **Workloads Simulated:**
  - Matrix sizes: 2000×2000, 3000×3000, 5000×5000
  - Cloudlet length formula: $2 \times N^2$ (for matrix × matrix multiplication)
- **Delay Modeling:**
  - **Zero Delay**: Simulates ideal local environment
  - **With Delay**: Uses custom DelayBroker to simulate:
    - 10 ms one-way latency
    - 1 Gbps bandwidth
    - 1 MiB input and output transfer size
- **Utilization Models:** UtilizationModelFull used for CPU, RAM, and bandwidth

4 Compilation Instructions

This project was developed and tested using Java JDK version 21.0.7 and compiled using Eclipse IDE. However, the project can also be compiled and executed via the terminal or any standard Java-compatible IDE. The project uses CloudSim 3.0.3 and requires specific JAR files to be included in the classpath.

Required JAR Files:

Ensure the following JAR files are available in your working directory (preferably in a folder named jars/):

- cloudsim-3.0.3.jar
- cloudsim-3.0.3-sources.jar
- cloudsim-examples-3.0.3.jar
- cloudsim-examples-3.0.3-sources.jar
- commons-math3-3.6.1.jar (located in /Downloads or moved to the jars/ folder)

4.1 Terminal Compilation

Navigate to the directory containing your Java files and run the following command to compile:

```
javac -cp "jars/cloudsim-3.0.3.jar:\
jars/cloudsim-3.0.3-sources.jar:\
jars/cloudsim-examples-3.0.3.jar:\
jars/cloudsim-examples-3.0.3-sources.jar:\
/Users/<your_user>/Downloads/commons-math3-3.6.1.jar" \
YourFile.java
```

Replace YourFile.java with the actual Java file you want to compile, such as:

```
 javac -cp "jars/cloudsim-3.0.3.jar:\
jars/cloudsim-3.0.3-sources.jar:\
jars/cloudsim-examples-3.0.3.jar:\
jars/cloudsim-examples-3.0.3-sources.jar:\
/Users/<your_user>/Downloads/commons-math3-3.6.1.jar" \
GpuDatacenterExample.java
```

4.2 Running the Program

To run the compiled class, use:

```
java -cp "jars/cloudsim-3.0.3.jar:\
jars/cloudsim-3.0.3-sources.jar:\
jars/cloudsim-examples-3.0.3.jar:\
jars/cloudsim-examples-3.0.3-sources.jar:\
/Users/<your_user>/Downloads/commons-math3-3.6.1.jar:." \
org.cloudbus.cloudsim.examples.custom.GpuDatacenterExample
```

4.3 Eclipse Build Path

If working inside Eclipse (Eclipse Foundation (2025):

- Use Java Build Path > Libraries to include all the JAR files mentioned above.
- Set the JRE System Library to JavaSE-21 [JDK 21.0.7].
- Ensure that all simulation files (GpuDatacenterExample.java, etc.) are placed under the correct package (e.g., org.cloudbus.cloudsim.examples.custom).

5 Execution Guide

To execute the simulation experiments, ensure that the project is successfully compiled with all required .jar dependencies. The simulation consists of six main Java classes, each corresponding to a specific processor type (CPU, GPU, PPU) and network scenario (zero-delay, with-delay). Each file contains an independent main() method that can be executed directly from the Eclipse IDE.

5.1 Executable Classes and Purpose

Table 1: Simulation Class Descriptions

Class Name	Description
`CpuDatacenterZeroDelay.java`	Simulates CPU execution without network delay (pure compute performance).
`CpuDatacenterWithDelay.java`	Simulates CPU execution with network delay (end-to-end evaluation).
`GpuDatacenterZeroDelay.java`	Simulates GPU execution without network delay.
`GpuDatacenterExample.java`	Simulates GPU execution with network delay (uses DelayBroker).
`PpuDatacenterZeroDelay.java`	Simulates PPU execution without network delay.
`PpuDatacenterWithDelay.java`	Simulates PPU execution with network delay (uses DelayBroker).

Each of these classes includes:

- Datacenter and host setup (including VM and PE MIPS).
- Definition of a matrix-based workload using Cloudlet.
- Optional delay modeling through DelayBroker where applicable.
- Print statements showing detailed timing metrics (data-in, compute, data-out, total time).

5.2 How to Run the Simulations

1. **Open the File**

   Open the desired class file (e.g., GpuDatacenterExample.java) in Eclipse.

2. **Run the Main Method**

   Right-click on the file and select **Run As → Java Application**. The simulation will begin execution in the Eclipse console.

3. **Observe the Output**

   The output will print time breakdowns such as:

   - Data-in time (network transmission)
   - Compute time (based on MIPS and matrix size)
   - Data-out time
   - Total execution time

   These values are used for evaluating device performance and calculating speedups.

5.3 Output Interpretation Example

For a 5000×5000 matrix workload with GPU (with delay), you might observe:

```
    === GPU NETWORK DELAY RUN ===
Data-in    : 0.100 s
Compute    : 3.333 s
Data-out   : 0.028 s
Total time : 3.462 s
```

These logs allow quantitative comparison of performance across CPU, GPU, and PPU devices.

5.4 Network delays

For simulations involving network delays:

- DelayBroker is used instead of the default DatacenterBroker.
- It calculates transmission time based on:
  - **Latency**: 10 ms one-way
  - **Bandwidth**: 1 Gbps
  - **Cloudlet file size**: 1 MiB (input and output)

The delay is injected at submission and return stages to model real-world conditions more accurately.

6 Results Visualization and Graph Generation

Once the simulations have been executed and output data has been collected, graphs can be generated to visualize performance trends across CPU, GPU, and PPU configurations. The recommended process is:

- **Export Simulation Results**

  Copy the simulation output values (Data-in, Compute, Data-out, Total time) from the console or log into a CSV file.

- **Use Python for Plotting**

  Install Python (3.9 or higher) and the required libraries(Python Software Foundation (2025),Hunter and et al. (2025):

  ```
  pip install pandas matplotlib
  ```

- **Example Python Code – Total execution time by matrix size**

7 Assumptions and Constraints

This section specifies the conditions and limitations under which the simulations were conducted, ensuring reproducibility of the results.

```python
import matplotlib.pyplot as plt

# Matrix sizes
matrix_sizes = [2000, 3000, 5000]

# Total execution times in seconds (example values — replace with your actual results)
cpu_total = [1.728, 3.728, 10.128]
gpu_total = [0.633, 1.328, 3.462]
ppu_total = [0.0, 0.259, 0.491]

plt.figure(figsize=(8, 6))

# Plot lines for CPU, GPU, and PPU
plt.plot(matrix_sizes, cpu_total, marker='o', label='CPU', linewidth=2)
plt.plot(matrix_sizes, gpu_total, marker='o', label='GPU', linewidth=2)
plt.plot(matrix_sizes, ppu_total, marker='o', label='PPU', linewidth=2)

# Chart labels and title
plt.title("Total Execution Time by Matrix Size", fontsize=14)
plt.xlabel("Matrix Size (N x N)", fontsize=12)
plt.ylabel("Total Execution Time (s)", fontsize=12)
plt.xticks(matrix_sizes)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend()

# Save the plot
plt.savefig("total_execution_time.png", dpi=300)
plt.show()
```

Figure 1: Total execution time by matrix size code snippet

7.1 Assumptions

- Software Environment: Simulations were executed using CloudSim 3.0.3 with Java Runtime Environment (JRE) 21.0.7.
- Datacentre Structure: A single datacentre was modeled with one physical host and one virtual machine (VM) per run.
- Workload Type: The workload consisted solely of matrix–matrix multiplication, with the computational requirement calculated as $2 \times N^2$ million instructions (MI).
- Performance Ratings: MIPS values for CPU, GPU, and PPU were fixed across all runs to represent consistent device capabilities.
- Network Settings:
  - Zero-delay runs assumed instantaneous data transfer.
  - With-delay runs used a fixed one-way latency of 10 ms and bandwidth of 1 Gbps for both input and output phases.
- Cloudlet Sizes: Input and output sizes were fixed at 1 MiB to maintain uniformity in network traffic.

7.2 Constraints

- Discrete-Event Limitations: Ultra-fast compute times on high-MIPS PPUs for very small workloads (e.g., 2000×2000) occasionally produced near-zero or negative values due to event resolution limits. For analysis, these were excluded from comparative speedup calculations.
- Workload Range: Results and interpretations focused on matrix sizes $\geq 3000 \times 3000$ to ensure compute time outweighed network overhead.
- Device Representation: MIPS values were proxies for real-world processors and do not account for full hardware complexity such as cache hierarchy or thermal limits.

- Simplified Networking: The model excluded dynamic network effects such as jitter, packet loss, or congestion.
- Scope Limitation: Energy consumption and cost modeling were beyond the scope of this simulation.

References

Chen, M. and et al. (2023). Photonic integrated circuit for matrix inversions and multiplications, *49th European Conference on Optical Communications (ECOC 2023)*, pp. 867–869.

CloudSim Project (2023). Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services, `https://github.com/Cloudslab/cloudsim`.

Eclipse Foundation (2025). Eclipse ide for java developers, `https://www.eclipse.org/downloads/`.

Hunter, J. D. and et al. (2025). Matplotlib documentation, `https://matplotlib.org/stable/index.html`.

Oracle Corporation (2024). Java platform, standard edition documentation – jdk 21, `https://docs.oracle.com/en/java/javase/21/`.

Python Software Foundation (2025). pandas documentation, `https://pandas.pydata.org/docs/`.