National College of Ireland

Strategic Task Placement: A Comparative Analysis of Lightweight Scheduling in Edge–Fog–Cloud Systems

MSc Research Project
Cloud Computing

Raj Dhamdhere
Student ID: x23122498

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Raj Dhamdhere
Student ID:	x23122498
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	11/08/2025
Project Title:	Strategic Task Placement: A Comparative Analysis of Lightweight Scheduling in Edge–Fog–Cloud Systems
Word Count:	1321
Page Count:	8

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:	Raj Dhamdhere
Date:	13th September 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):	

Strategic Task Placement: A Comparative Analysis of Lightweight Scheduling in Edge–Fog–Cloud Systems

Raj Dhamdhere
x23122498

1 Introduction

This configuration manual provides a detailed procedure for installing and configuring all required modules of this project to provide a distributed scheduling and execution environment. The system simulates a three-layered architecture consisting of edge, fog, and cloud layers. Lightweight scheduling strategies were processed on synthetic IoT workloads, generating performance logs for comparison. The setup is deployed with Python scripts, Raspberry PI, and AWS EC2 cloud instances. Constrained scheduling conditions were replicated to assess latency, energy consumption, and task deadlines based on the analysis and visualization of execution logs.

2 System Requirements

In this section, the hardware and software tools that were used to implement the hybrid execution environment are described.

2.1 Hardware Requirements

1. **Edge Layer:** Raspberry Pi 3B with 1GB RAM, to compute latency-intensive tasks.
2. **Fog Layer:** AWS EC2 instance (t3.small) with mid-level processing capacity. *Amazon EC2 - Cloud Compute Capacity - AWS* (n.d.)
3. **Cloud Layer:** AWS EC2 instance (t3.large) for processing high-level computational tasks. *Amazon EC2 - Cloud Compute Capacity - AWS* (n.d.)
4. **Local Dispatcher System:** Intel Core i7-13700HX CPU with 16GB RAM for dispatching tasks amongst the three layers.

2.2 Software Requirements

1. **Python:**
   (a) Local dispatcher - version 3.12.6 *Python 3.12 documentation* (n.d.)
   (b) Edge-Fog-Cloud nodes - version 3.10.12 *Python 3.10 documentation* (n.d.)
2. **Operating System:**

(a) Debian GNU/Linux version 12 - bookworm (on Raspberry Pi) Ltd (n.d.)

(b) Windows 11 PRO 24H2 64-bit (on local system)

(c) AWS Ec2 instances (t3.large,t3.small) - ubuntu-jammy-22.04-amd64-server

3. **Remote Access Tools:**

   (a) RealVNC Viewer 7.13.1 *RealVNC®* (n.d.) - to connect Raspberry PI

   (b) WSL (Windows Subsystem for Linux) mattwojo (n.d.) - accessing fog and cloud EC2 instances via SSH.

4. **Python Libraries:** pandas version 2.2.2[1] , matplotlib version 3.10.0[2], seaborn version 0.13.2[3]

5. **Visualization:** Google Colab environment used for data visualization and analysis *Google Colab* (n.d.)

2.3 Synthetic Workload Generation

Python scripts were executed on Google Colab to generate a synthetic workload. It contained a balanced mix of IoT tasks, saved in the form of a JSON object. Task files of 500 and 1000 entries were created and fetched to the dispatcher as input.

The following parameters were included in each task:

timestamp	priority	deadline_ms	input_size_kb	type	cpu_cycles_mi	task_id

Table 1: Synthetic Task Parameters

3 Environment Setup

This section describes the setup of each component: the dispatcher, edge, fog, and cloud layers. A separate Python virtual environment was maintained to ensure modularity and version isolation.

3.1 System Updation and Python Environment Setup

To prepare the environment for deployment, the following steps must be completed on all Linux-based nodes (edge, fog, and cloud).

Step 1: System Package Updation

1. Run the following command:

```
sudo apt update && sudo apt upgrade -y
```

[1] `https://pandas.pydata.org/`

[2] `https://matplotlib.org/`

[3] `https://seaborn.pydata.org/`

Step 2: Installation of Python and Tools

1. Install Python and virtual environment tools:

```
sudo apt install python3-pip python3-venv -y
```

Step 3: Create and Activate Virtual Environment

1. Create a virtual environment:

```
python3 -m venv venv
```

2. Activate the environment:

```
source venv/bin/activate
```

Step 4: Install Required Python Libraries

1. Install required libraries:

```
pip install flask requests
```

3.2 Dispatcher Setup (Local System)

The dispatcher executes one of the scheduling strategies after the environment configuration. The following commands were utilized to execute the algorithms.

1. `python dispatcher.py` – Dehury et al. (2024)-based heuristic
2. `python dispatcherpriority.py` – Fahad et al. (2022)-based priority
3. `python dispatchermeta.py` – GA-PSO metaheuristic

The algorithm in the dispatcher reads the synthetic JSON workload and distributes the tasks based on internal logic to the receivers via Flask API and logs the summary locally.

3.3 Edge Layer Setup (Raspberry Pi 3B)

After successful booting of Raspberry Pi with Raspbian OS:

1. Connect to Wi-Fi and enable SSH.
2. Update the system using `apt`.
3. Set up a Python virtual environment.
4. Installation of required libraries: `flask`, `requests`.
5. The receiver script is initialized to start listening tasks.

3.4 Fog Layer Setup (EC2 t3.small)

Launching of an EC2 instance with a Unix-based Linux system:

1. Connect to instance via SSH.
2. Update the system and install Python environment.
3. Virtual Environment setup and activation.
4. Installation of Python dependencies.
5. Start the receiver script on port 5000.

3.5 Cloud Layer Setup (EC2 t3.large)

The steps performed in the fog layer needs to be repeated for the cloud layer using t3.large EC2 instance, which represents a high level of computational capacity.

3.6 Flask Communication Testing

After successful initialization of all receiver nodes, verify the Flask connectivity using the CURL request state below:

```
curl -X POST http://<node_ip>:5000/task \
-H "Content-Type: application/json" \
-d '{
      "task_id": "cloud_test_001",
      "cpu_cycles_mi": 2500,
      "input_size_kb": 500,
      "priority": 3,
      "deadline_ms": 5000,
      "type": "C",
      "timestamp": "2025-08-01T12:00:00Z"
    }'
```

HTTP response - 200 confirms the successful communication between the dispatcher and the receiver

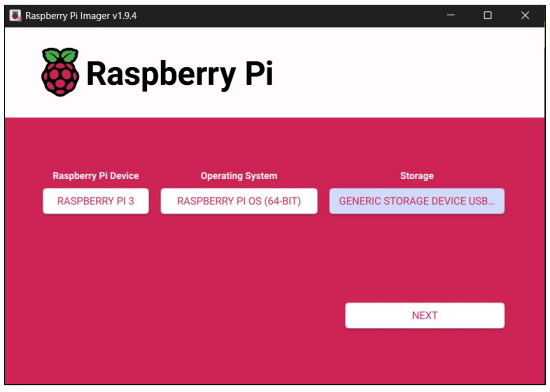

Figure 1: Flashing Raspberry Pi OS using Imager

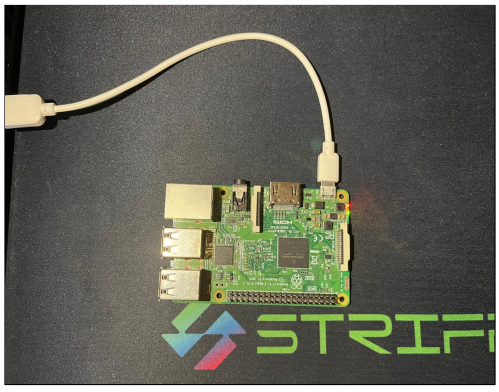

Figure 2: Raspberry Pi 3B setup as Edge Node

4 Logging and Visualization

The tasks distributed using the dispatcher scripts produce structured logs that contain task ID, execution time, transfer time, energy consumption, deadline, and status code. An example of terminal log is in figure 3.

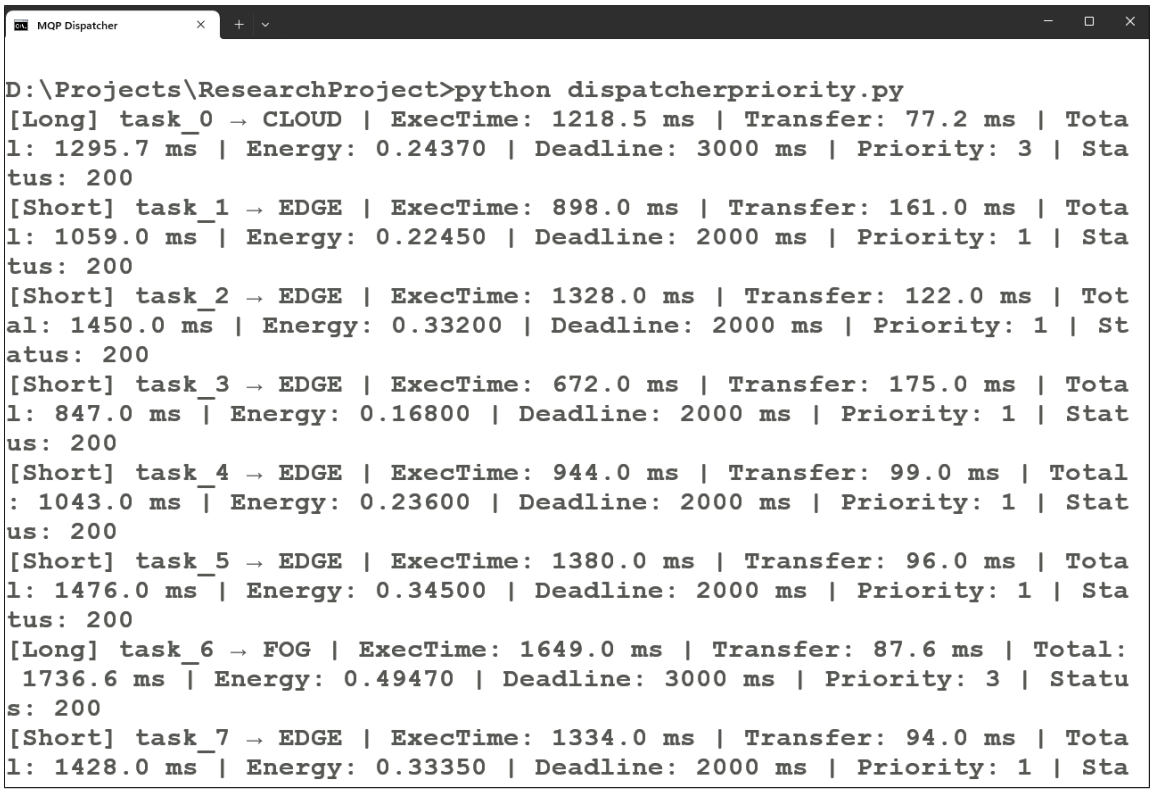

Figure 3: Terminal output showing dispatched tasks and execution summary

After the execution of each scheduling algorithm, the logs were exported as CSV files that contain a performance summary across execution layers. These CSVs were uploaded to the Google Colab platform for visualization and performance comparison. Figure 4 highlights the performance metrics in structured CSV format.

Layer	Count	AvgTime(ms)	AvgEnergy	MissedDL
edge	364	1130.8	0.24819	636
fog	503	1201.5	0.33872	133
cloud	133	1159.9	0.21982	0
	MissedDL%	AvgExec(ms)	AvgTransfer(ms)	StdDev(ms)
	63.60%	992.8	138.1	214.9
	20.91%	1129.1	72.5	340.7
	0%	1099.1	60.8	89.3

Figure 4: CSV summary with performance metrics for edge, fog, and cloud layers

References

Amazon EC2 - Cloud Compute Capacity - AWS (n.d.).
URL: *https://aws.amazon.com/ec2/*

Dehury, C. K., Veeravalli, B. and Srirama, S. N. (2024). HeRAFC: Heuristic resource allocation and optimization in MultiFog-Cloud environment, *Journal of Parallel and Distributed Computing* **183**: 104760.
URL: *https://www.sciencedirect.com/science/article/pii/S0743731523001302*

Fahad, M., Shojafar, M., Abbas, M., Ahmed, I. and Ijaz, H. (2022). A multi-queue priority-based task scheduling algorithm in fog computing environment, *Concurrency and Computation: Practice and Experience* **34**(28): e7376.
URL: *https://onlinelibrary.wiley.com/doi/10.1002/cpe.7376*

Google Colab (n.d.).
URL: *https://colab.research.google.com/*

Ltd, R. P. (n.d.). Raspberry Pi software.
URL: *https://www.raspberrypi.com/software/*

mattwojo (n.d.). Windows Subsystem for Linux Documentation.
URL: *https://learn.microsoft.com/en-us/windows/wsl/*

Python 3.10 documentation (n.d.).
URL: *https://docs.python.org/3.10/*

Python 3.12 documentation (n.d.).
URL: *https://docs.python.org/3.12/*

RealVNC® (n.d.).
URL: *https://www.realvnc.com/en/connect/download/viewer/*