EdgeCloudSim – Enhanced Orchestration Strategy Based on Task Criticality and Deadline Awareness

Overview

This project extends **EdgeCloudSim**, a well-known open-source simulation framework for modeling mobile edge computing (MEC) environments.
The modification introduces a **new orchestration strategy** that improves task allocation decisions by considering **task criticality**, **deadline constraints**, and **energy optimization (EO)**.

The goal is to ensure that high-criticality and time-sensitive tasks are prioritized and executed efficiently within a two-tier edge–cloud architecture, leading to higher task success rates, lower latency, and balanced VM loads.

Background

What EdgeCloudSim Is

EdgeCloudSim is a discrete event simulator built on **CloudSim**, designed to model complex MEC systems. It supports:

- **Multiple tiers** (Mobile, Edge, and Cloud),
- **Mobility modeling**,
- **Task offloading**,
- **VM allocation**, and
- **Various application types** (health, AR, infotainment, heavy compute).

By default, EdgeCloudSim uses **simple heuristic-based orchestrators** (like NextFit or FirstFit) that don't consider deeper metrics such as **task criticality**, **deadline**, or **QoS constraints**.

This limitation motivated the design of a more **intelligent and adaptive orchestration strategy**.

Problem Statement

In the existing simulator:

- Tasks are allocated **uniformly** without considering **criticality** (importance) or **deadline urgency**.
- High-priority tasks may fail or experience long latency if the orchestrator assigns them poorly.

- Edge VMs are often **underutilized** or **overloaded**, leading to performance degradation.
- There is no mechanism to dynamically optimize **energy efficiency** or balance **QoS vs resource utilization**.

Therefore, the goal is to:

Design and integrate an **Enhanced Orchestration (EO) Strategy** that intelligently assigns tasks based on their **criticality, deadline**, and **VM availability**, while maintaining optimal **latency**, **load**, and **energy efficiency**.

The Proposed Solution

Core Idea

The **Enhanced Orchestrator (EO)** extends EdgeCloudSim's NextFit strategy to include **task-level intelligence**:

- Tasks are tagged with:
    - **Criticality (0 or 1)** → indicates importance of the task.
    - **Deadline (continuous value)** → determines urgency.
- The orchestrator dynamically evaluates:
    - Task criticality,
    - Deadline slack,
    - Current VM load,
    - Network latency,
    - Energy consumption.
- Based on these, it selects the **most suitable VM** (edge or cloud tier).

Architecture

The EO strategy operates within the **two-tier system**:

1. **Tier 1 (Edge devices)** → Handles high-criticality or low-latency tasks.
2. **Tier 2 (Cloud layer)** → Handles low-criticality or high-latency-tolerant tasks.

The orchestrator continuously monitors available resources and updates task–VM mappings in real time.

Implementation Details

Files Modified / Created

File	Description
LimitedRandomTaskGenerator.java	Generates tasks with random **criticality** and **deadline** attributes.
Task.java	Extended to include new properties: criticality, deadline, and corresponding getters/setters.
EnhancedOrchestrator.java	Implements the EO logic that considers criticality and deadlines in VM selection.
SimulationManager.java	Registers the new orchestrator and ensures it runs as part of the simulation flow.
SimLogger.java	Modified to log additional parameters (success/fail counts, delays, loads).
applications.xml, edge_devices.xml	Updated to reflect new app types and device tiers for experiments.
runner script	Added execution entry for EO variant (WITH_EO).

Strategy Logic (Simplified Pseudocode)

```
for each incoming task {
   evaluate all VMs:
      compute score = w1 * (1 / latency)
                    + w2 * (1 / VM_load)
                    + w3 * (criticality_weight)
                    - w4 * (deadline_violation_risk);
   assign task to VM with maximum score;
}
```

Where:

- w1...w4 are weight parameters tuned experimentally.
- Critical tasks (high criticality) are prioritized when edge VMs have available capacity.
- Tasks nearing deadlines are dispatched faster or escalated to cloud if edge is busy.

Integration Process

1. **Extended Task Model**
   Added attributes for criticality and deadline in Task.java.
2. **Task Generation Logic**
   Modified LimitedRandomTaskGenerator.java to assign random criticality and deadline values during simulation.
3. **New Orchestrator Class**
   Created EnhancedOrchestrator.java under edge_orchestrator package implementing the EO scheduling algorithm.
4. **Simulation Configuration**
   Updated SimulationManager.java to include EnhancedOrchestrator in the orchestrator registry.
5. **Result Logging**
   Modified SimLogger.java to export data such as:
   - Total tasks generated
   - Successful vs failed tasks
   - VM load
   - Service time, processing time, network delay
   - Energy consumption

Evaluation and Results

Experimental Setup

- **Devices simulated:** 100
- **Applications:** Health, AR, Infotainment, Heavy Compute
- **Tiers:** Edge + Cloud

- **Strategies compared:**
    1. Baseline → NextFit (single-tier)
    2. Proposed → Two-Tier with EO

Key Performance Metrics

Metric	Old Strategy	New EO Strategy	Improvement
Total Tasks	2343	5704	↑ 143%
Task Success Ratio	97.9%	**98.7%**	Improved
Avg Service Time (s)	1.88	**1.27**	Reduced
Avg Processing Time (s)	1.87	**1.02**	Reduced
Avg Network Delay (s)	0.019	0.082	Slight increase
VM Load	0.0	**1.57**	Better utilization
Energy Consumption	0.0	0.0	—

Visualization Summary

Generated plots clearly show that:

- Task completion rates increased significantly.
- Average latency dropped due to intelligent task routing.
- VM load became balanced, eliminating idle edge nodes.
- Network delay saw a slight, expected increase due to added communication overhead — but remains negligible.

Running the Simulation

Step 1 – Locate the Simulator Folder

After cloning the repository, open your terminal and navigate to the simulator's directory.
For example:

```
cd ~/Desktop/EdgeCloudSim
```

Step 2 – Move to the Sample Application Folder

```
cd scripts/sample_app1
```

Step 3 – Grant Execution Permission

Before running the simulator, make the shell scripts executable:

```
chmod +x compile.sh
chmod +x runner.sh
```

Step 4 – Compile the Simulator

Run the following command to compile all necessary Java files:

```
./compile.sh
```

Step 5 – Run the Simulation

Once compilation is complete, execute the simulation using:

```
./runner.sh result default_config edge_devices.xml applications.xml
1
```

This will start the experiment using the defined configuration files.

Step 6 – View Simulation Results

After the simulation completes, results are stored automatically under:

```
scripts/sample_app1/results/default_config/
```

You'll find output files such as:

```
SIMRESULT_TWO_TIER_WITH_EO_NEXT_FIT_100DEVICES_ALL_APPS_GENERIC.log
```

These log files contain the data used to analyze metrics like task success ratio, latency, VM load, and energy performance.

Conclusion

The **Enhanced Orchestration (EO)** strategy successfully integrates **criticality** and **deadline awareness** into EdgeCloudSim, enabling smarter and more adaptive task allocation.

Key outcomes:

- Higher success rate and throughput.
- Reduced latency and processing time.
- Better VM utilization across edge tiers.
- Proven stability under varying workloads.

This demonstrates that **context-aware orchestration** can significantly improve MEC performance in both research and real-world applications.

Credits

- Base Simulator: EdgeCloudSim by BOUN