import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

// Edge computing node

public class EdgeComputeNode {

private String name;

private boolean available;

private Lock availabilityLock;

private int currentTaskCount;

public EdgeComputeNode(String name) {

this.name = name;

this.available = true;

this.availabilityLock = new ReentrantLock();

this.currentTaskCount = 0;

}

public String getName() {

return name;

}

// Check whether the node is available

public boolean isAvailable() {

availabilityLock.lock();

try {

return available;

} finally {

availabilityLock.unlock();

}

}

// Execute the task

public void executeTaskLocally(Task subtask) {

availabilityLock.lock();

try {

if (available) {

// Simulate the task execution process

System.out.println("Node '" + name + "' executing task: " + subtask.getName());

simulateTaskExecution();

// Update the task count

currentTaskCount++;

} else {

System.out.println("Node '" + name + "' is not available for task execution.");

}

} finally {

availabilityLock.unlock();

}

}

// Simulated task execution method

private void simulateTaskExecution() {

try {

// Simulated task execution time

Thread.sleep(2000);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

// Callback after completing the task

public void onTaskCompleted() {

availabilityLock.lock();

try {

// Update the task count

currentTaskCount--;

if (currentTaskCount == 0) {

// Mark the node as available after all tasks are completed

available = true;

}

} finally {

availabilityLock.unlock();

}

}

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

// Edge computing node

public class EdgeComputeNode {

private String name;

private boolean enabled;

private boolean available;

private Lock availabilityLock;

private int currentTaskCount;

public EdgeComputeNode(String name) {

this.name = name;

this.enabled = true;

this.available = true;

this.availabilityLock = new ReentrantLock();

this.currentTaskCount = 0;

}

public String getName() {

return name;

}

// Enable a node

public void enableNode() {

availabilityLock.lock();

try {

enabled = true;

System.out.println("Node '" + name + "' has been enabled.");

} finally {

availabilityLock.unlock();

}

}

// Disable a node

public void disableNode() {

availabilityLock.lock();

try {

enabled = false;

available = false;

System.out.println("Node '" + name + "' has been disabled.");

} finally {

availabilityLock.unlock();

}

}

// Check whether the node is enabled

public boolean isEnabled() {

availabilityLock.lock();

try {

return enabled;

} finally {

availabilityLock.unlock();

}

}

// Check whether the node is available

public boolean isAvailable() {

availabilityLock.lock();

try {

return enabled && available;

} finally {

availabilityLock.unlock();

}

}

// Execute the task

public void executeTaskLocally(Task subtask) {

availabilityLock.lock();

try {

if (isAvailable()) {

// Simulate the task execution process

System.out.println("Node '" + name + "' executing task: " + subtask.getName());

simulateTaskExecution();

// Update the task count

currentTaskCount++;

} else {

System.out.println("Node '" + name + "' is not available for task execution.");

}

} finally {

availabilityLock.unlock();

}

}

// Simulated task execution method

private void simulateTaskExecution() {

try {

// Simulated task execution time

Thread.sleep(2000);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

// Callback after completing the task

public void onTaskCompleted() {

availabilityLock.lock();

try {

// Update the task count

currentTaskCount--;

if (currentTaskCount == 0) {

// Mark the node as available after all tasks are completed

available = true;

}

} finally {

availabilityLock.unlock();

}

}

}

}