Name : Ajay Shitkar
Roll No : 58
Experiment No.8
Implement Task Assignment Approach
Date of Performance: 01/04/24
Date of Submission: 05/04/24

Aim: To implement Task Assignment Approach.

Objective: Implement Task Assignment Approach.

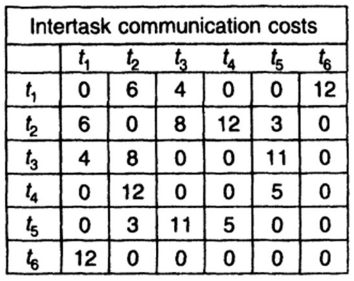
Theory:

In task assignment approach, process is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance.

A process is split into pieces called tasks. This split occurs along natural boundaries, so that data transfers among the tasks is minimized. The amount of computation required by each task and the speed of each processor are known. The cost of processing each task on every node of the system is also known. The inter-process communication (IPC) costs between every pair of tasks are known.

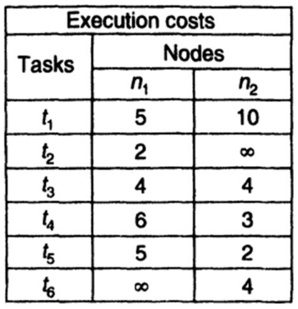
Other constraints, such as resource requirements of the tasks and the available resources at each node, precedence relationships among the tasks, and so on, are also known.

Consider the following example:



Intertask communication costs						
	t_1	t_2	t_3	t_4	t_5	t_6
t_1	0	6	4	0	0	12
t_2	6	0	8	12	3	0
t_3	4	8	0	0	11	0
t_4	0	12	0	0	5	0
t_5	0	3	11	5	0	0
t_6	12	0	0	0	0	0

(*a*)



Execution costs		
Tasks	Nodes	
	n_1	n_2
t_1	5	10
t_2	2	∞
t_3	4	4
t_4	6	3
t_5	5	2
t_6	∞	4

(*b*)

The serial assignment cost can be calculated as follows:

Serial assignment execution cost $(x) = x_{11} + x_{21} + x_{31} + x_{42} + x_{52} + x_{62}$
$= 5 + 2 + 4 + 3 + 2 + 4 = 20$

Serial assignment communication cost $(c) = c_{14} + c_{15} + c_{16} + c_{24} + c_{25} + c_{26} + c_{34} + c_{35} + c_{36}$
$= 0 + 0 + 12 + 12 + 3 + 0 + 0 + 11 + 0 = 38$

Serial assignment total cost $= x + c = 20 + 38 = 58$

And the optimal assignment cost can be calculated as follows:

Optimal assignment execution cost $(x) = x_{11} + x_{21} + x_{31} + x_{41} + x_{51} + x_{62}$
$= 5 + 2 + 4 + 6 + 5 + 4 = 26$

Optimal assignment communication cost $(c) = c_{16} + c_{26} + c_{36} + c_{46} + c_{56}$
$= 12 + 0 + 0 + 0 + 0 = 12$

Optimal assignment total cost $= x + c = 26 + 12 = 38$

For implementing task assignment approach, take from the user all the parameters and then calculate the cost.

Code:

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;


class Server {

    private final String name;

    private final Map<String, Integer> executionCosts; // Execution costs for each task

    private int load;


    public Server(String name) {

        this.name = name;

        this.executionCosts = new HashMap<>();

        this.load = 0;

    }


    public String getName() {

        return name;

    }


    public int getLoad() {

        return load;
```

```
}
    public void increaseLoad() {
        load++;
    }

    public void decreaseLoad() {
        load--;
    }

    public void setExecutionCost(String task, int cost) {
        executionCosts.put(task, cost);
    }

    public int getExecutionCost(String task) {
        return executionCosts.getOrDefault(task, 0); // Default to 0 if task not found
    }
}

class LoadBalancer {
    private final List<Server> servers;
    private int currentIndex;

    public LoadBalancer(List<Server> servers) {
```

```
this.servers = servers;
        this.currentIndex = 0;
    }

    public Server getNextServer() {
        Server nextServer = servers.get(currentIndex);
        currentIndex = (currentIndex + 1) % servers.size();
        return nextServer;
    }
}



class Main {
    private static final int COMMUNICATION_COST_PER_REQUEST = 2;

    public static void main(String[] args) {
        // Create servers
        List<Server> servers = new ArrayList<>();
        Server server1 = new Server("Server 1");
        Server server2 = new Server("Server 2");
        Server server3 = new Server("Server 3");

        // Define execution costs for tasks on each server
```

```
server1.setExecutionCost("Task A", 5);
        server1.setExecutionCost("Task B", 7);


        server2.setExecutionCost("Task A", 6);
        server2.setExecutionCost("Task B", 8);


        server3.setExecutionCost("Task A", 4);
        server3.setExecutionCost("Task B", 6);


        servers.add(server1);
        servers.add(server2);
        servers.add(server3);


        LoadBalancer loadBalancer = new LoadBalancer(servers);


        int totalExecutionCost = 0;
        int totalCommunicationCost = 0;


        // Simulate requests
        String[] tasks = {"Task A", "Task B"};
        for (int i = 0; i < 10; i++) {
            String task = tasks[i % tasks.length]; // Rotate through tasks
            Server server = loadBalancer.getNextServer();
```

```
 server.increaseLoad();


            int executionCost = server.getExecutionCost(task);

            totalExecutionCost += executionCost;


            totalCommunicationCost += COMMUNICATION_COST_PER_REQUEST;

            System.out.println("Request " + i + " (Task: " + task + ") handled by " + server.getName()
+

                    " (Execution Cost: $" + executionCost + ")");

        }


        // Print server loads

        for (Server server : servers) {

            System.out.println(server.getName() + " load: " + server.getLoad());

        }


        // Calculate total cost

        int totalCost = totalExecutionCost + totalCommunicationCost;

        System.out.println("Total Execution Cost: $" + totalExecutionCost);

        System.out.println("Total Communication Cost: $" + totalCommunicationCost);

        System.out.println("Total Cost: $" + totalCost);

    }

}
```

Output :

```
java -cp /tmp/KQLxc26Ezq Main
Request 0 (Task: Task A) handled by Server 1 (Execution Cost: $5)
Request 1 (Task: Task B) handled by Server 2 (Execution Cost: $8)
Request 2 (Task: Task A) handled by Server 3 (Execution Cost: $4)
Request 3 (Task: Task B) handled by Server 1 (Execution Cost: $7)
Request 4 (Task: Task A) handled by Server 2 (Execution Cost: $6)
Request 5 (Task: Task B) handled by Server 3 (Execution Cost: $6)
Request 6 (Task: Task A) handled by Server 1 (Execution Cost: $5)
Request 7 (Task: Task B) handled by Server 2 (Execution Cost: $8)
Request 8 (Task: Task A) handled by Server 3 (Execution Cost: $4)
Request 9 (Task: Task B) handled by Server 1 (Execution Cost: $7)
Server 1 load: 4
Server 2 load: 3
Server 3 load: 3
Total Execution Cost: $60
Total Communication Cost: $20
Total Cost: $80
```

Conclusion: The implemented Task Assignment approach demonstrated the capability to allocate tasks to suitable servers, aiming to optimize system performance. The system effectively divided a process into related tasks and assigned them to nodes based on predetermined execution costs. The simulation showcased how different servers handled tasks based on their execution costs, leading to varying server loads. Moreover, the total execution cost and communication cost were calculated, providing insights into the overall cost associated with task processing. The experiment highlights the significance of efficient task assignment in distributed systems, emphasizing the importance of considering execution costs and communication overheads for achieving optimal resource utilization. Such approaches are crucial for enhancing system efficiency, scalability, and performance in real-world scenarios.