Experiment 9– Load Balancing Algorithm

Learning Objective: Student should be able to develop a program for Load Balancing Algorithm.

Tools : Java

Theory:

Load Balancing Algorithm:

The scheduling algorithms using this approach are known as load-balancing algorithms or load-leveling algorithms. These algorithms arc based on the intuition that, for better resource utilization, it is desirable for the load in a distributed system to be balanced evenly. Thus, a load-balancing algorithm tries to balance the total system load by transparently transferring the workload from heavily loaded nodes to lightly loaded nodes in an attempt to ensure good overall performance relative to some specific metric of system performance. When considering performance from the user point of view, the metric involved is often the response time of the processes. However, when performance is considered from the resource point of view, the metric involved is the total system throughput. In contrast to response time, throughput is concerned with seeing that all users are treated fairly and that all are making progress. Notice that the resource view of maximizing resource utilization is compatible with the desire to maximize system throughput. Thus the basic goal of almost all the load-balancing algorithms is to maximize the total system throughput.

Static versus Dynamic

At the highest level, we may distinguish between static and dynamic load-balancing algorithms. Static algorithms use only information about the average behavior of the system, ignoring the current state of the system. On the other hand, dynamic algorithms react to the system state that changes dynamically.

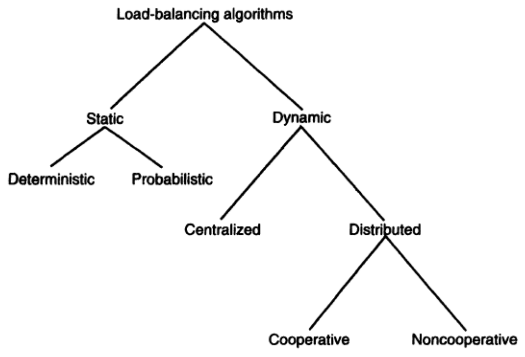


Fig. 7.3 A taxonomy of load-balancing algorithms.

Deterministic versus Probabilistic

Static load-balancing algorithms may be either deterministic or probabilistic. Deterministic algorithms use the information about the properties of the nodes and the characteristics of the processes to be scheduled to deterministically allocate processes to nodes. Notice that the task assignment algorithms basically belong to the category of deterministic static load-balancing algorithms.

Centralized versus Distributed

Dynamic scheduling algorithms may be centralized or distributed. In a centralized dynamic scheduling algorithm, the responsibility of scheduling physically resides on a single node. On the other hand, in a distributed dynamic scheduling algorithm, the work involved in making process assignment decisions is physically distributed among the various nodes of the system. In the centralized approach, the system state information is collected at a single node at which all scheduling decisions are made. This node is called the centralized server node. All requests for process scheduling are handled by the centralized server, which decides about the placement of a new process using the state information stored in it. The centralized approach can efficiently make process assignment decisions because the centralized server knows both the load at each node and the number of processes needing service. In the basic method, the other nodes periodically send status update messages to the central server node. These messages are used to keep the system state information up to date at the centralized server node. One might consider having the centralized server query the other nodes for state information. This would reduce message traffic if state information was used to answer several process assignment requests, but since nodes can change their load any time due to local activities, this would introduce problems of stale state information.

Cooperative versus Noncooperative

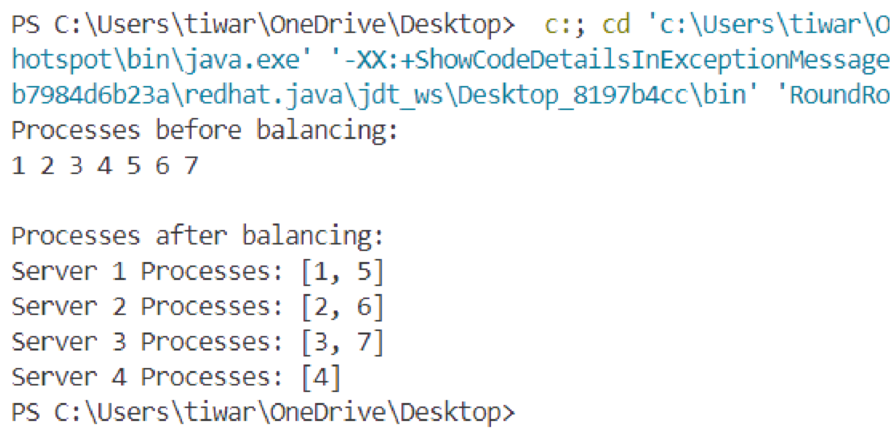
Distributed dynamic scheduling algorithms may be categorized as cooperative and noncooperative. In noncooperative algorithms, individual entities act as autonomous entities and make scheduling decisions independently of the actions of other entities. On the other hand, in cooperative algorithms, the distributed entities cooperate with each other to make scheduling decisions. Hence, cooperative algorithms are more complex and involve larger overhead than noncooperative ones. However, the stability of a cooperative algorithm is better than that of a noncooperative algorithm.

Result and Discussion:

Code:

```
import java.util.ArrayList;

public class RoundRobinLoadBalancer {
    // Simulating servers with ArrayLists
    private ArrayList<Integer>[] servers;
    private int numServers;

    public RoundRobinLoadBalancer(int numServers) {
        this.numServers = numServers;
        servers = new ArrayList[numServers];
        for (int i = 0; i < numServers; i++) {
            servers[i] = new ArrayList<>();
        }
    }

    // Add processes to the servers
    public void addProcesses(int[] processes) {
        int currentIndex = 0;
        for (int process : processes) {
            servers[currentIndex].add(process);
            currentIndex = (currentIndex + 1) % numServers; // Round robin distribution
        }
    }

    // Print processes present in each server
    public void printProcesses() {
        for (int i = 0; i < numServers; i++) {
            System.out.println("Server " + (i + 1) + " Processes: " + servers[i]);
        }
    }

    public static void main(String[] args) {
        // Initial processes in the servers
        int[] initialProcesses = {1, 2, 3, 4, 5, 6, 7};

        // Number of servers
        int numServers = 4;

        RoundRobinLoadBalancer loadBalancer = new
RoundRobinLoadBalancer(numServers);

        System.out.println("Processes before balancing:");
        for (int process : initialProcesses) {
            System.out.print(process + " ");
        }
        System.out.println();

        loadBalancer.addProcesses(initialProcesses);

        System.out.println("\nProcesses after balancing:");
        loadBalancer.printProcesses();
    }
}
```

Output:



```
PS C:\Users\tiwar\OneDrive\Desktop>  c:; cd 'c:\Users\tiwar\O
hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessage
b7984d6b23a\redhat.java\jdt_ws\Desktop_8197b4cc\bin' 'RoundRo
Processes before balancing:
1 2 3 4 5 6 7

Processes after balancing:
Server 1 Processes: [1, 5]
Server 2 Processes: [2, 6]
Server 3 Processes: [3, 7]
Server 4 Processes: [4]
PS C:\Users\tiwar\OneDrive\Desktop>
```

Learning Outcomes: The student should have the ability to

LO1: Comprehend the Load balancing concept

LO2: Analyze different that load balancing methods

Course Outcomes: Upon completion of the course students will be able to understand Load Balancing Algorithm.

Conclusion:

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				