ANISH ADNANI
D17B/101

EXPERIMENT 5

Aim: Write a program to demonstrate Load Balancing in distributed systems

Objective: To understand the importance of load balancing and its approaches to redistribute tasks from heavily loaded nodes to lightly loaded nodes.

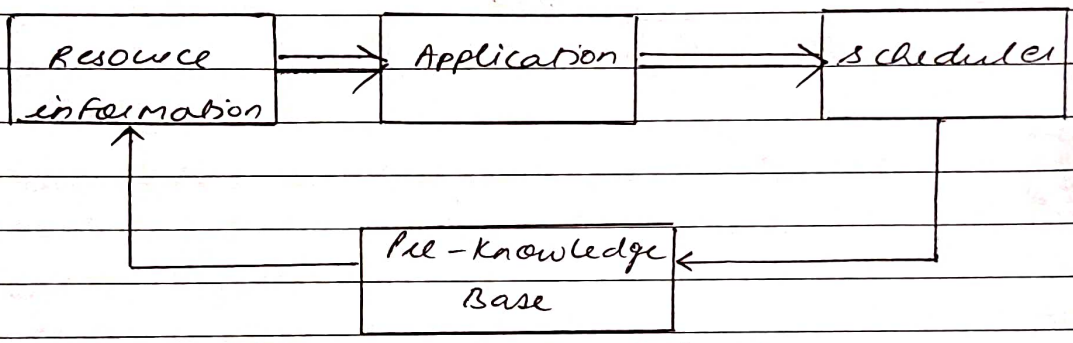
Theory:

In load balancing processes are distributed among nodes to localize the load among all nodes. There a number of nodes in a distributed system. The task that arrive in these systems are typically not uniformly distributed. Some of the nodes may be heavily loaded while the others may be lightly loaded. It may be possible to increase the overall throughput of the system if we allow heavily loaded nodes to redistribute tasks to lightly loaded nodes. This is known as load balancing. The aim of load balancing is to try to improve the performance of a distributed system, mainly in terms of resource availability or response time by distributing workload amongst a set of cooperating hosts.

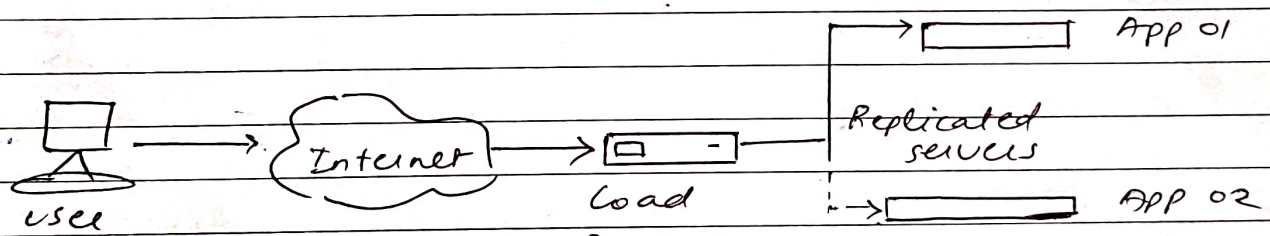
The algorithm for load balancing can be classified into two categories: static or dynamic

a) Static Load Balancing: Ignore the current state of system static load balancing algorithm allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based

on an average load for our workstation cluster. The decision related to load balance are made at compile time when resource requirements are estimated. The advantage of this sort of algorithm is the simplicity in terms of both implementation as well as overhead since there is no need to constantly monitor the workstations for performance statistics

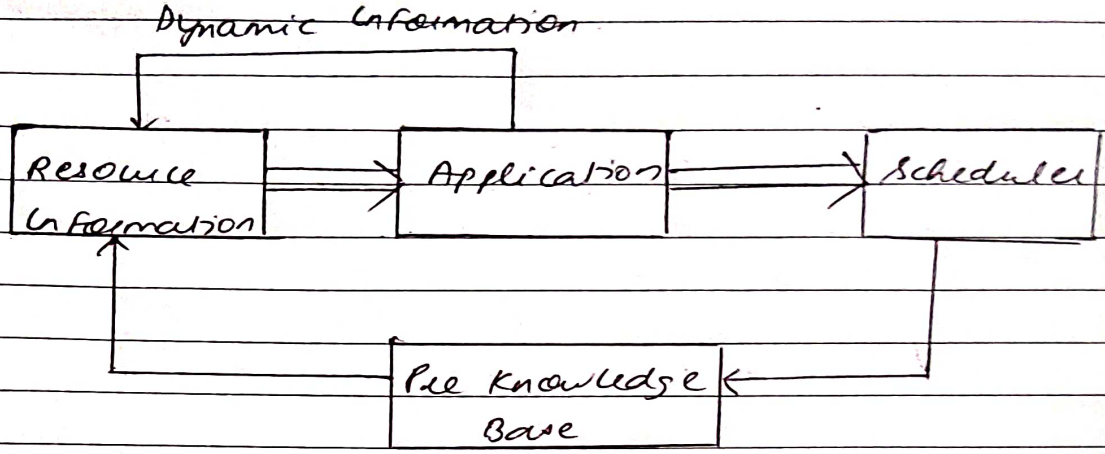


STATIC LOAD BALANCING



LOAD BALANCING

Dynamic Load Balancing:
Use the current state information for load balancing. Dynamic load balancing algorithm makes changes to the distribution of work among workstations at run time; they use current or recent load information when making distribution decisions



DYNAMIC LOAD BALANCING

Multicomputers with dynamic load balancing allocate reallocate resources at runtime based on a priori task information, which may determine when and where tasks can be migrated. As a result, dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. However, this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits

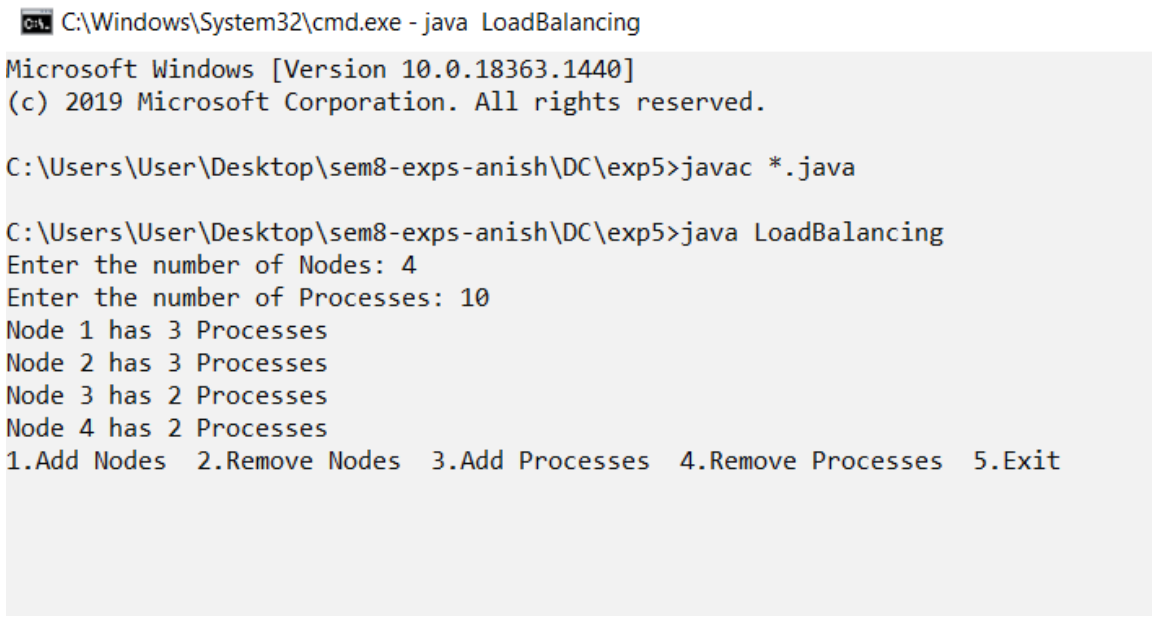
Conclusion: Load Balancing in distributed systems was successfully implemented.

Java Program for load balancing in distributed systems

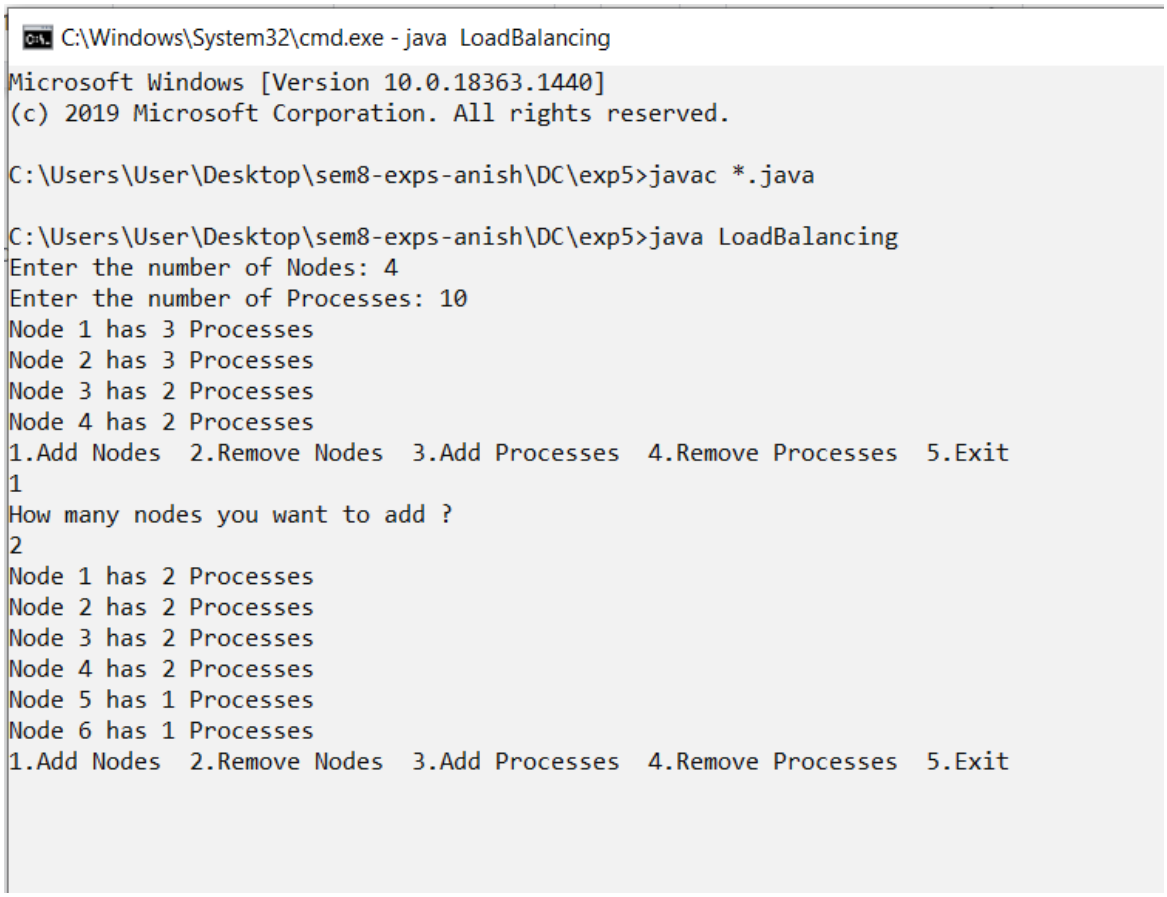
```
import java.util.*;
public class LoadBalancing{
   static void printLoad(int nodes, int processes)    {
      int each=processes/nodes;
      int extra=processes%nodes;
      int total=0;
      int i=0;
      for(i=0; i<extra; i++)       {
         System.out.println("Node "+(i+1)+" has "+(each+1)+" Processes");
      }
      for(;i<nodes;i++)       {
         System.out.println("Node "+(i+1)+" has "+each+" Processes");
      }
   }
   public static void main(String[] args)    {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the number of Nodes: ");
      int nodes=sc.nextInt();
      System.out.print("Enter the number of Processes: ");
      int processes=sc.nextInt();
      while(true) {
         printLoad(nodes, processes);
         System.out.println("1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove
Processes  5.Exit ");
         switch(sc.nextInt())   {
            case 1:
               System.out.println("How many nodes you want to add ? ");
               nodes+=sc.nextInt();
               break;
            case 2:
               System.out.println("How many nodes you want to remove ? ");
               nodes-=sc.nextInt();
               break;
            case 3:
               System.out.println("How many Processes you want to add ? ");
               processes+=sc.nextInt();
               break;
            case 4:
               System.out.println("How many Processes you want to remove ? ");
               processes-=sc.nextInt();
               break;
            case 5:
               return;
```

```
        }
      }
    }
}
```

Output


C:\Windows\System32\cmd.exe - java LoadBalancing
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\User\Desktop\sem8-exps-anish\DC\exp5>javac *.java
C:\Users\User\Desktop\sem8-exps-anish\DC\exp5>java LoadBalancing
Enter the number of Nodes: 4
Enter the number of Processes: 10
Node 1 has 3 Processes
Node 2 has 3 Processes
Node 3 has 2 Processes
Node 4 has 2 Processes
1.Add Nodes 2.Remove Nodes 3.Add Processes 4.Remove Processes 5.Exit

Adding Nodes



Removing Nodes

```
C:\Windows\System32\cmd.exe - java  LoadBalancing
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\User\Desktop\sem8-exps-anish\DC\exp5>javac *.java

C:\Users\User\Desktop\sem8-exps-anish\DC\exp5>java LoadBalancing
Enter the number of Nodes: 4
Enter the number of Processes: 10
Node 1 has 3 Processes
Node 2 has 3 Processes
Node 3 has 2 Processes
Node 4 has 2 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
1
How many nodes you want to add ?
2
Node 1 has 2 Processes
Node 2 has 2 Processes
Node 3 has 2 Processes
Node 4 has 2 Processes
Node 5 has 1 Processes
Node 6 has 1 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
2
How many nodes you want to remove ?
3
Node 1 has 4 Processes
Node 2 has 3 Processes
Node 3 has 3 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
```

Adding Processes

C:\Windows\System32\cmd.exe - java LoadBalancing

```
Node 1 has 3 Processes
Node 2 has 3 Processes
Node 3 has 2 Processes
Node 4 has 2 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
1
How many nodes you want to add ?
2
Node 1 has 2 Processes
Node 2 has 2 Processes
Node 3 has 2 Processes
Node 4 has 2 Processes
Node 5 has 1 Processes
Node 6 has 1 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
2
How many nodes you want to remove ?
3
Node 1 has 4 Processes
Node 2 has 3 Processes
Node 3 has 3 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
3
How many Processes you want to add ?
5
Node 1 has 5 Processes
Node 2 has 5 Processes
Node 3 has 5 Processes
1.Add Nodes  2.Remove Nodes  3.Add Processes  4.Remove Processes  5.Exit
```

Removing Processes