AIM:
Write a program to evaluate Load Balancing in distributed systems for **Dynamic loads** using **various load balancing strategies. (at least 3)**

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

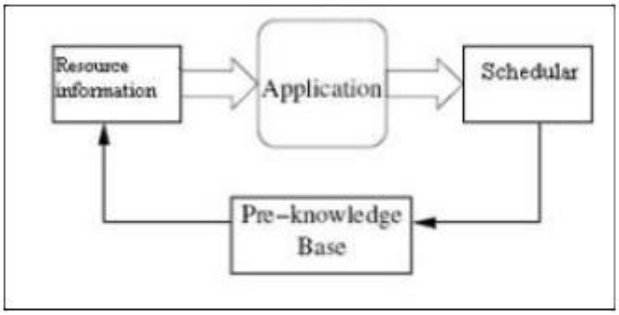
THEORY:
In load balancing processes are distributed among nodes to equalize the load among all nodes. There a number of nodes in a distributed system. The tasks that arrive in these systems are typically not uniformly distributed. Some of the nodes may be heavily loaded while the others may be lightly 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. This load balancing method tries to ensure that the workload on each host is within some small range (or balance criterion) of the workload present on all the other nodes in the system.

Load Balancing Algorithm:

The algorithms for load balancing can be classified into two categories: **static or dynamic.**

(a) Static Load Balancing: Ignore the current state of the system.

Static load balancing algorithms 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 of our workstation cluster. The decisions 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.



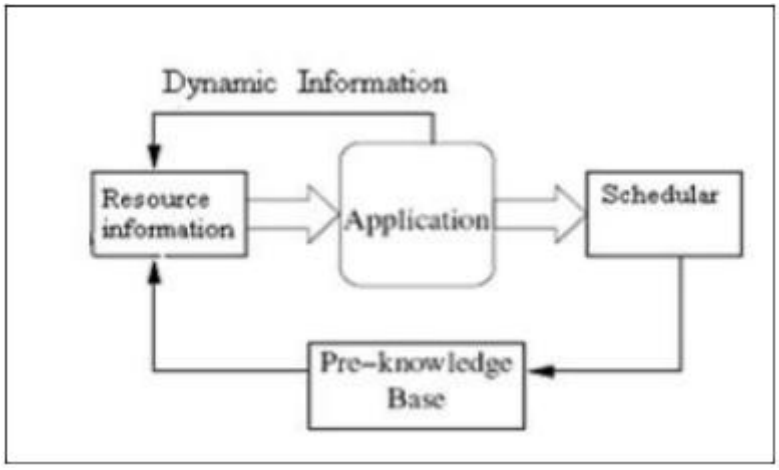
Static Load Balancing

However, static algorithms only work well when there is not much variation in the load on the workstations.

Basic Example of static load balancer:

Dynamic Load Balancing:

Use the current state information for load balancing. Dynamic load balancing algorithms make 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 no a priori task information, which may determine when and whose 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.

1. **Load estimation policy** determines how to estimate the workload of a node. Some measurable parameters (with time and node dependent factor) can be the following:

- Total number of processes on the node
- Resource demands of these processes
- Instruction mixes of these processes
- Architecture and speed of the node's processor

In some cases the true load could vary widely depending on the remaining service time, which can be measured in several way
An acceptable method for use as the load estimation policy in these systems would be to measure the CPU utilization of the nodes
Central Processing Unit utilization is defined as the number of CPU cycles actually executed per unit of real time. It can be measured by setting up a timer to periodically check the CPU state (idle/busy)

2. **Process transfer policy** determines whether to execute a process locally or remote
3. **State information exchange policy** determines how to exchange load information among nodes
4. **Location policy** determines to which node the transferable process should be sent
5. **Priority assignment policy**
determines the priority of execution of local and remote processes
6. **Migration limiting policy** determines the total number of times a process can migrate

Types implemented:

Least Connection load balancing method

Least connection load balancing is a dynamic load balancing algorithm where client requests are distributed to the application server with the least number of active connections at the time the client request is received. In cases where application servers have similar specifications, one server may be overloaded due to longer lived connections; this algorithm takes the active connection load into consideration. This technique is most appropriate for incoming requests that have varying connection times and a set of servers that are relatively similar in terms of processing power and available resources.

Weighted Least Connection load balancing method

Weighted least connection builds on the least connection load balancing algorithm to account for differing application server characteristics. The administrator assigns a weight to each application server based on the relative processing power and available resources of each server in the farm. The LoadMaster makes load balancing decisions based on active connections and the assigned server weights (e.g., if there are two servers with the lowest number of connections, the server with the highest weight is chosen).

Weighted Response Time load balancing method

The weighted response time load balancing algorithm that uses the application server's response time to calculate a server weight. The application server that is responding the fastest receives the next request. This algorithm is appropriate for scenarios where the application response time is the paramount concern.

OUTPUT:

1. Least Connection

Initial Load Distribution:
Put number of servers and processes to be 0.

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\DC\7-LoadBalancing\DynamicAlgos> python new.py
Dynamic Load Balancing Algorithms:
1. Least Connection
2. Weighted Least Connection
3. Least Response Time
Enter your choice of load balancing algorithm: 1
Enter the number of servers: 0
Enter the number of processes: 0

Initial Load distribution:

Current Load distribution:

Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option:
```

Add/Remove thus update them through the menu driven program:

1.1. Add servers 1,2,3:

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 1
Enter name for the new server: server1
Server 'server1' added.

Current Load distribution:
server1 has 0 processes.

Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 1
Enter name for the new server: server2
Server 'server2' added.

Current Load distribution:
server1 has 0 processes.
server2 has 0 processes.
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 1
Enter name for the new server: server3
Server 'server3' added.

Current Load distribution:
server1 has 0 processes.
server2 has 0 processes.
server3 has 0 processes.
```

1.2. Add processes (Destribution of Load): p1, p2, p3

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p1
Process 'p1' added. Assigned to server1.
Load distribution after adding process:
server1: 1 connection(s): p1
server2: 0 connection(s)
server3: 0 connection(s)

Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p2
Process 'p2' added. Assigned to server2.
Load distribution after adding process:
server1: 1 connection(s): p1
server2: 1 connection(s): p2
server3: 0 connection(s)
```

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p3
Process 'p3' added. Assigned to server3.
Load distribution after adding process:
server1: 1 connection(s): p1
server2: 1 connection(s): p2
server3: 1 connection(s): p3
```

1.3. Remove process

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 4
Enter the name of the process to remove: p2
Process 'p2' removed from server2.
Load distribution after removing process:
server1: 1 connection(s): p1
server2: 0 connection(s):
server3: 1 connection(s): p3
```

1.4. Remove Server (Redistribution of Load)

Consider the following actions for better intutive understanding:

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p4
Process 'p4' added. Assigned to server2.
Load distribution after adding process:
server1: 1 connection(s): p1
server2: 1 connection(s): p4
server3: 1 connection(s): p3
```

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p5
Process 'p5' added. Assigned to server1.
Load distribution after adding process:
server1: 2 connection(s): p1, p5
server2: 1 connection(s): p4
server3: 1 connection(s): p3
```

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p6
Process 'p6' added. Assigned to server2.
Load distribution after adding process:
server1: 2 connection(s): p1, p5
server2: 2 connection(s): p4, p6
server3: 1 connection(s): p3
```

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 2
Enter name of the server to remove: server1
Server 'server1' removed.
Process 'p1' reassigned to server3.
Process 'p5' reassigned to server2.

Current Load distribution:
server2 has 3 process(es): p4, p6, p5
server3 has 2 process(es): p3, p1
```

2. Weighted Least Connection

```
Dynamic Load Balancing Algorithms:
1. Least Connection
2. Weighted Least Connection
3. Least Response Time
Enter your choice of load balancing algorithm: 2
```

2.1. Add servers

```
Enter the number of servers: 3
Enter name for server 1: s1
Enter name for server 2: s2
Enter name for server 3: s3
```

2.2. Add Weights for the servers:

```
Enter weight for server s1: 1
Enter weight for server s2: 2
Enter weight for server s3: 3
```

2.3. Add processes (Destribution of Load)

```
Enter the number of processes: 3
Enter name for process 1: p1
Enter name for process 2: p2
Enter name for process 3: p3

Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p4
Process 'p4' added. Assigned to s3.
Load distribution after adding process:
s1: 1 connection(s): p1
s2: 1 connection(s): p2
s3: 2 connection(s): p3, p4
```

New process is allocated to server with max weight.

2.4. Remove Server (Redistribution of Load)

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 2
Enter name of the server to remove: s1
Server 's1' removed.
Process 'p1' reassigned to s2.

Current Load distribution:
s2 has 2 process(es): p2, p1
s3 has 2 process(es): p3, p4
```

P1 got distributed to s2 which had the least weight.

3. Least Response Time

```
Dynamic Load Balancing Algorithms:
1. Least Connection
2. Weighted Least Connection
3. Least Response Time
Enter your choice of load balancing algorithm: 3
```

3.1. Add servers

```
Enter the number of servers: 3
Enter name for server 1: s1
Enter name for server 2: s2
Enter name for server 3: s3
```

3.2. Add response times for servers

```
Enter response time for server s1: 10
Enter response time for server s2: 20
Enter response time for server s3: 30
```

3.3. Add processes

```
Enter name for process 1: p1
Enter name for process 2: p2
Enter name for process 3: p3
```

Load distributions as of now:

```
Initial Load distribution:
s1: 1 connection(s)
s2: 1 connection(s)
s3: 1 connection(s)

Current Load distribution:
s1 has 1 process(es): p1
s2 has 1 process(es): p2
s3 has 1 process(es): p3
```

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 3
Enter name for the new process: p4
Process 'p4' added. Assigned to s1.
Load distribution after adding process:
s1: 2 connection(s): p1, p4
s2: 1 connection(s): p2
s3: 1 connection(s): p3
```

new process is allocated to server with least response time

3.4. Remove Server (Redistribution of Load)

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 2
Enter name of the server to remove: s3
Server 's3' removed.
Process 'p3' reassigned to s1.

Current Load distribution:
s1 has 3 process(es): p1, p4, p3
s2 has 1 process(es): p2
```

redistribution is done to server with least response time
And then exit the program:

```
Menu:
1. Add Server
2. Remove Server
3. Add Process
4. Remove Process
5. Exit
Select an option: 5
Exiting...
Process balancing completed.
```

CONCLUSION:

The load balancing algorithms, including Least Connection, Weighted Least Connection, and Least Response Time, have been implemented with a menu-driven program allowing updates to the number of servers and processes. Users can dynamically manage server attributes such as connections, weight, and response time for efficient resource allocation.