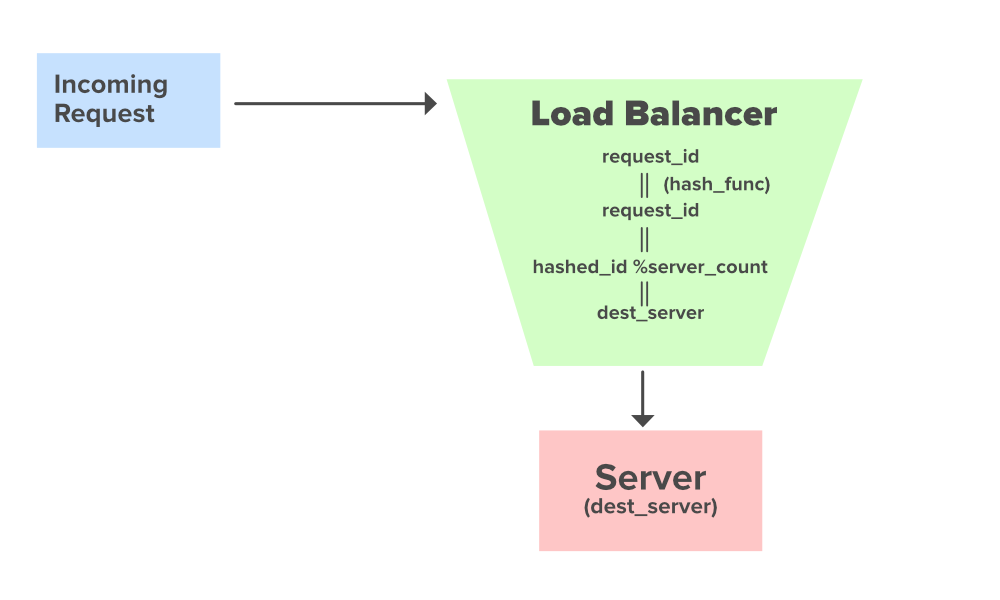
Title : **Write code to simulate requests coming from clients and distribute them among the servers using the load balancing algorithms.**

Theory:

**What are Load Balancers?**

In case multiple servers are present the incoming request coming to the system needs to be directed to one of the multiple servers. We should ensure that every server gets an equal number of requests. The requests must be distributed in a uniform manner across all the servers. The component which is responsible for distributing these incoming requests uniformly across the servers is known as Load Balancer. A Load Balancer acts as a layer between the incoming requests coming from the user and multiple servers present in the system.

We should **avoid the scenarios where a single server is getting most of the requests while the rest of them are sitting idle.** There are various Load Balancing Algorithms that ensure even distribution of requests across the servers.



Hashing Approach to direct requests from the Load Balancer

We will be discussing the Hashing Approach to direct the requests to multiple servers uniformly. Suppose we have server\_count as the Total number of servers present in the System and a load\_balancer to distribute the requests among those servers. A request with an id request\_id enters the system. Before reaching the destination server it is directed to the load\_balancer from where it is further directed to its destination server. When the request reaches the load balancer the hashing approach will provide us with the destination server where the request is to be directed.

* **request\_id** : Request ID coming to get served
* **hash\_func** : Evenly distributed Hash Function
* **hashed\_id** : Hashed Request ID
* **server\_count** : Number of Servers

**Java Code:**

**class** GFG {

**public** **static** **int** hash\_func(**int** request\_id)

    {

        // Computing the hash request id

**int** hashed\_id = 112;

**return** hashed\_id;

    }

**public** **static** **void** route\_request\_to\_server(**int** dest\_server)

    {

        System.out.println("Routing request to the Server ID : " + dest\_server);

    }

**public** **static** **int** request\_id = 23; // Incoming Request ID

**public** **static** **int** server\_count = 10; // Total Number of Servers

**public** **static** **void** main(String args[])

    {

**int** hashed\_id = hash\_func(request\_id); // Hashing the incoming request id

**int** dest\_server = hashed\_id % server\_count; // Computing the destination server id

        route\_request\_to\_server(dest\_server);

    }

}

**Python:**

The **Load Balancer** class has two methods: **round robin** for round-robin load balancing and **random selection** for random load balancing. The **simulate\_client\_requests** function simulates client requests and prints the server selected by each algorithm for each request.

import random

class **LoadBalancer**:

def \_\_init\_\_(self, servers):

self.servers = servers

self.server\_index\_rr = 0

def round\_robin(self):

server = self.servers[self.server\_index\_rr]

self.server\_index\_rr = (self.server\_index\_rr + 1) % len(self.servers)

return server

def random\_selection(self):

return random.choice(self.servers)

def simulate\_client\_requests(load\_balancer, num\_requests):

for i in range(num\_requests):

# Simulating client request

print(f"Request {i+1}: ", end="")

# Using Round Robin algorithm for load balancing

server\_rr = load\_balancer.round\_robin()

print(f"Round Robin - Server {server\_rr}")

# Using Random algorithm for load balancing

server\_random = load\_balancer.random\_selection()

print(f"Random - Server {server\_random}")

print()

if \_\_name\_\_ == "\_\_main\_\_":

# List of servers

servers = ["Server A", "Server B", "Server C"]

# Create a LoadBalancer instance

load\_balancer = LoadBalancer(servers)

# Simulate 10 client requests

simulate\_client\_requests(load\_balancer, 10)