# Software Requirements Specification

For

**Meta-Heuristic Load Balancing Algorithms In Cloud Computing**

25th April, 2023

Prepared by

| Specialization | SAP ID | Name |
|---|---|---|
| CCVT | 500083967 | Annie Jain |
| CCVT | 500082786 | Ayush Juyal |
| CCVT | 500087519 | Rishabh Anand |


Department of Systematics
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

**Guided by:**

Dr. Rajeev Tiwari
Professor (PL)
Cluster- Systemics
School of Computer Science

**Cluster Head:**

Dr. Neelu J. Ahuja

# Table of Contents

# Revision History

| Date | Change | Reason for Changes | Mentor Signature |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

# 1. INTRODUCTION

One of the most widely used technologies now-a-days in the field of information technology and its enabled services is cloud computing. With the help of online computing resources, cloud computing, an internet-based network technology, has contributed to the rapid advancement of communication technology by serving clients with a range of needs. Its resources include platforms for software creation, testing tools, and both hardware and software applications. Services are used to carry out this resource delivery. Due to its remarkable qualities and advantages, including great flexibility, scalability, and dependability, a number of researchers and service providers are moving towards it. Low-cost virtual organizations and resources are provided through the cloud. The main reason why cloud computing is so popular is because it provides virtualization. Customers can access resources using cloud computing based on their needs. Although there are numerous advantages to cloud computing, there are also significant challenges, including those related to load balancing, work scheduling, VM migration, security, and many more.

When resources are used as effectively as feasible, a cloud computing model is effective, and this effective use can be done by implementing and maintaining adequate management of cloud resources. By using effective resource scheduling, allocation, and scalability approaches, resource management can be accomplished. Customers receive these resources in the form of Virtual Machines (VM) thanks to the virtualization process, which makes use of the hypervisor, a component that can be either software or hardware or both. The biggest benefit of cloud computing is the conversion of a single user physical machine into a multiuser virtual machine. With the limited virtual resources that are now accessible, the Cloud Service Provider (CSP) has a significant impact on how services are delivered to consumers.

Some VMs will experience a high volume of user tasks while serving user requests, while others will experience a lower volume. Because of this, the Cloud Service Provider (CSP) is left with computers that are out of balance and have a wide gradient of user tasks and resource consumption.

The issue of load unbalancing is an unwanted occurrence on the CSP side that impairs both the assured Quality of Service (QoS) on the agreed Service Level Agreement (SLA) between customer and provider and the performance and efficacy of the computing resources. In these situations, load balancing (LB), which is a peculiar research issue of interest among researchers, becomes necessary.

In cloud computing, load balancing can take place at the VM or physical machine level. A task uses resources from a virtual machine, and when many tasks arrive at the same machine at once, the resources become spent and are no longer accessible to fulfil the new task demands. The VM is considered to have become overwhelmed when such a situation occurs. At this point, chores will either starve to death or come to a standstill with no chance of being completed. As a result, jobs must be moved to another resource on another VM.

The workload migration process consists of three fundamental steps: resource discovery, which seeks for a different resource that is acceptable, and workload migration, which transfers additional jobs to accessible resources. Three distinct components, often referred to as load balancer, resource discovery, and task migration units, respectively, carry out these functions.

In a distributed system like cloud computing, load balancing is the process of redistributing workload to make sure that no computing machine is overloaded, underloaded, or idle. By attempting to speed up several limited metrics including reaction time, execution time, system stability, etc., load balancing aims to increase cloud performance. It is a method of optimization where the problem of task scheduling is NP-hard. Numerous load balancing strategies have been put forth by researchers, with the majority of attention being paid to task scheduling, task allocation, resource scheduling, resource allocation, and resource management. To the best of our knowledge, we were unable to locate an extensive and detailed literature addressing the causes that lead to load unbalancing situations. The load balancing survey projects were unable to offer an accurate, systematic

classification of methods and procedures. The project's major goal is to review the prior research and point out its benefits and drawbacks. The difficulties encountered in cloud load balancing are also compared with other existing load balancing solutions. The survey also identifies the causes of the load unbalancing issue and makes recommendations for approaches that could be applied in subsequent research.

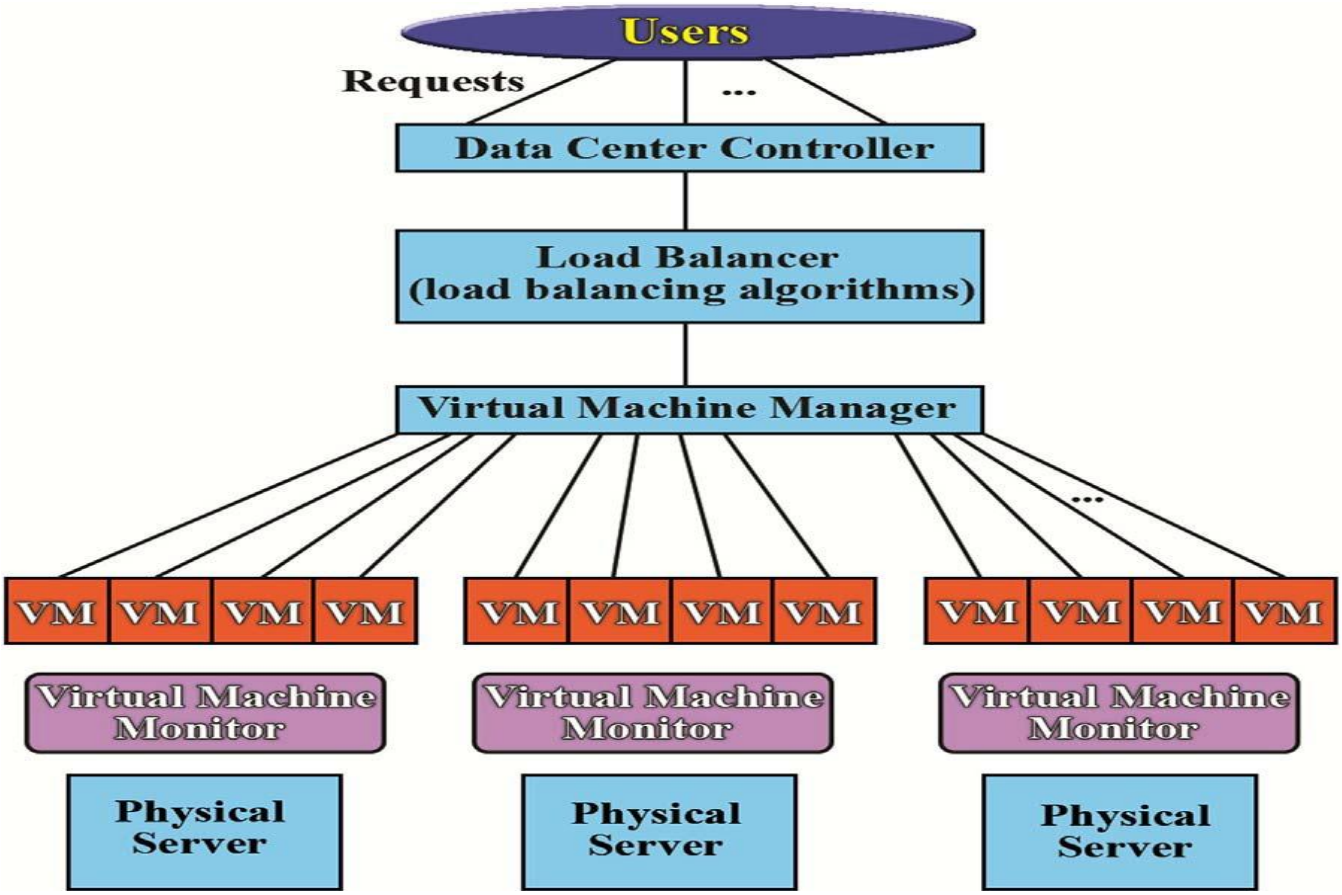



**Figure 1.1**

This project focuses in particular on the effective load balancing process. It is becoming increasingly difficult to assign cloud tasks fairly so that the nodes within the cloud computing environment will have a balanced load as the field of cloud computing becomes more popular and at the same time, there are more intensive tasks waiting to be processed. This task allocation strategy is known as load balancing. Load balancing has a significant impact on cloud computing performance since it seeks to improve resource efficiency, increases throughput, speed up reaction times, and prevent overloading of any single resource. Load balancing makes cloud computing more affordable and improves customer satisfaction. In order for the task that is currently operating to be finished without interruption, "it is that the method of ensuring the equal distribution of work on the pool of system node or processor." One type of load balancing used in cloud computing is called cloud load balancing, and it can be done both manually and on a categorised basis. There are many algorithms made for distributing the load among various tasks. Because of the special requirements of workshop site management, standard scheduling techniques must be combined with new advanced manufacturing techniques. This work's main motivation is to use metaheuristic techniques to investigate the clustering

problem. As previously noted, optimization techniques can be very helpful in finding a good solution because the development of balanced clusters leading to the operation of an energy-efficient network involves the adequate consideration of many criteria such as nodes' proximity and cluster size. The main objective of network lifetime improvement can be successfully accomplished with the acquisition of balanced clusters and rotation of the cluster head's duty among the nodes across the network rounds. The idea of differential evolution, a metaheuristic technique, is used in this research to present the Metaheuristic Load-Balancing-Based Clustering Technique, a revolutionary energy-efficient clustering protocol for wireless sensor networks. A suitable fitness function is defined in the suggested strategy to create a balanced network partitioning. The approach freezes the clusters once they are complete and permits the CH-role rotation among the cluster members. A complete collection of simulations exhibits the demonstration of the enhanced network lifetime and network energy consumption to demonstrate the effectiveness of the plan.

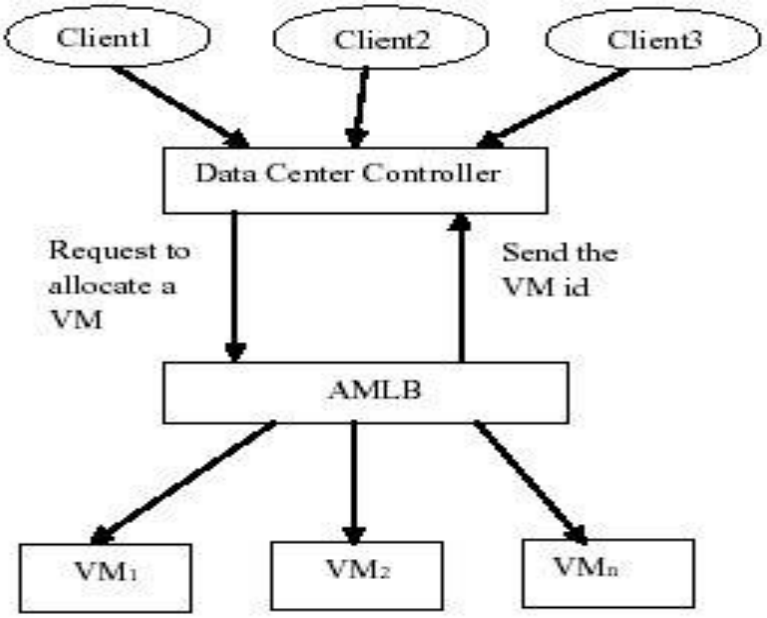



**Figure 1.2**

We utilised a tool called Cloud Analyst to investigate the load balancing methods. Cloud analyst is a cloud sim - based GUI tool that is primarily used for modelling and analysing large-scale cloud computing environments. Additionally, it makes it quick and simple for the modeler to run the simulation frequently while changing the settings. The GUI interface of the cloud analyst tool is shown in the diagram below. The configuration of simulation, definition of internet properties, and running of simulation are its three key menus. These menus are used to configure the simulation process as a whole.


**Figure 1.3**

## 1.1 Purpose of the Project

An emerging computer paradigm called load balancing allows for the vast storage of data and services in the cloud, making them accessible online from any connected device. It is renowned for being a supplier of dynamic services over the internet employing extremely vast, scalable, and virtualized resources. The goal of load balancing is to achieve optimal resource usage, increase throughput, reduce reaction time, and prevent overload by distributing workload over numerous computer clusters, network lines, or other resources. It is a method that uniformly distributes the dynamic local work load across all of the cloud's nodes in order to prevent a situation in which some nodes are overloaded while others are not. Its objective is to increase the resource efficiency and general performance of the system. With the use of load balancers, load balancing is carried out in a transparent manner to the client making the request, with each incoming request being routed. The load balancer uses a variety of scheduling algorithms to decide which server should handle a request and then transmits it to the chosen server based on specified characteristics, such as availability or the current workload.

The following is a discussion of various metrics used in current cloud computing load balancing techniques:

- The ability of an algorithm to carry out load balancing for a system with any finite number of nodes is known as scalability. This metric has to be raised.
- Resource Utilization is used to assess how well resources are being utilised. It ought to be tuned for effective load balancing.
- Performance is used to assess the system's effectiveness. This must be enhanced at a fair cost, for instance, by cutting down on task response time while maintaining tolerable delays.
- Response Time measures how long it takes a specific load balancing algorithm in a distributed system to respond. Reduce this parameter as much as possible.
- The amount of overhead incurred when putting a load-balancing algorithm into practice is determined by Overhead Associated. It consists of communication between processors and processes as well as overhead caused by task relocation. This should be kept to a minimum to ensure effective load balancing.

### 1.2 Project Scope

The scope of our project is making metaheuristic load balancing algorithms which is improved version of their original version. This is achieved by combining it with other meta heuristic algorithm which will add on additional features to it and increase the response ability. Here, cloud analyst which is a cloudsim based GUI tool used for giving user-friendly environment. The user will get full privilege to configure the data center and the region in which they are located, number of requests, load balancing policies, cost of service, etc. By configuring all the above configurations then we will run the simulation which tells us about the data center average response time, average processing time, cost of the service usage for that load balancing algorithm. These data will be represented graphically after the simulation.

### 1.3 Target Beneficiary

The target beneficiary of this project is each and every website provider who need to balance the complex workload efficiently, improve the response time of the website, maximize utilization of the resources to get better performance.

## 2. PROJECT DESCRIPTION

### 2.1 Load Balancing Algorithms Used

- Ant Colony Optimization- Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the behavior of real ant colonies. It is commonly used to solve optimization problems, such as the traveling salesman problem.

  The main idea behind ACO is that ants are capable of finding the shortest path between their colony and a food source by leaving a trail of pheromones. The more ants that follow a particular path, the stronger the pheromone trail becomes, making it more attractive to other ants. ACO mimics this behavior by simulating a colony of artificial ants that deposit pheromones on edges of a graph representing the problem to be solved. As the ants move through the graph, they preferentially follow edges with higher pheromone concentrations, allowing them to explore the graph and converge on the optimal solution.

  Advantages of ACO:

- ACO is a highly parallelizable algorithm, which means it can be efficiently implemented on parallel architectures such as multi-core processors and distributed computing systems.
- ACO can quickly converge to a near-optimal solution, even for large and complex problems.
- ACO is a metaheuristic algorithm, which means it does not rely on any assumptions about the problem being solved. This makes it applicable to a wide range of optimization problems.
- ACO is capable of handling dynamic environments, where the problem parameters or constraints may change over time.

  Disadvantages of ACO:

- ACO can be sensitive to the initial pheromone values and parameters, which can make it difficult to find the optimal solution for some problems.
- ACO may converge to a local optimum rather than the global optimum, especially for problems with multiple local optima.
- ACO requires a large number of iterations to converge, which can be computationally expensive for very large problems.
- ACO may not be suitable for problems with complex constraints or highly nonlinear objective functions.

- Honey Bee Optimization- Honey Bee Optimization (HBO) is a swarm intelligence algorithm inspired by the behavior of honey bees. HBO is commonly used to solve optimization problems, such as the traveling salesman problem.

  The main idea behind HBO is that bees use a combination of local and global search strategies to efficiently collect nectar from flowers. The bees communicate with each other by performing a waggle dance that indicates the direction and distance to the location of the best nectar source. HBO mimics this behavior by simulating a colony of artificial bees that perform local and global search operations to explore the solution space.

  Advantages of HBO:

- HBO is a highly parallelizable algorithm, which means it can be efficiently implemented on parallel architectures such as multi-core processors and distributed computing systems.
- HBO can quickly converge to a near-optimal solution, even for large and complex problems.
- HBO is a metaheuristic algorithm, which means it does not rely on any assumptions about the problem being solved. This makes it applicable to a wide range of optimization problems.
- HBO can handle constraints and multiple objectives efficiently.

  Disadvantages of HBO:

- HBO can be sensitive to the initial parameters, which can make it difficult to find the optimal solution for some problems.
- HBO may converge to a local optimum rather than the global optimum, especially for problems with multiple local optima.
- HBO requires a large number of iterations to converge, which can be computationally expensive for very large problems.
- HBO is not suitable for problems with complex constraints or highly nonlinear objective functions.
- HBO may require a significant amount of memory, especially for problems with large solution spaces.

- Tabu search- Tabu search is a metaheuristic algorithm used to solve combinatorial optimization problems, such as the traveling salesman problem. It was first introduced by Fred Glover in 1986.

  The main idea behind tabu search is to avoid cycling through previously visited solutions by maintaining a tabu list of forbidden moves. This encourages the algorithm to explore new regions of the solution space and prevent it from getting stuck in local optima.

  Advantages of Tabu Search:

- Tabu search is a flexible algorithm that can be adapted to various optimization problems.
- Tabu search can handle a wide range of problem types, including discrete, continuous, and mixed-integer optimization problems.
- Tabu search is often able to find high-quality solutions in a relatively short amount of time.
- Tabu search is a deterministic algorithm, which means it can be easily replicated to ensure reproducibility of results.
- Tabu search can be easily combined with other optimization algorithms, such as simulated annealing, to improve performance.

Disadvantages of Tabu Search:

- Tabu search requires careful tuning of its parameters, such as the size and duration of the tabu list, to ensure good performance.
- Tabu search can be sensitive to the choice of initial solution, which can impact its ability to find the global optimum.
- Tabu search may not be suitable for very large or complex optimization problems, as it can require a large number of iterations to converge.
- Tabu search may not perform well when faced with problems that have a high degree of symmetry or redundancy in the search space.
- Tabu search is a local search algorithm, which means it may get stuck in local optima and fail to find the global optimum.

- Simulation Annealing- Simulated annealing is a metaheuristic algorithm used to solve combinatorial optimization problems, such as the traveling salesman problem. It was first introduced by Kirkpatrick, Gelatt, and Vecchi in 1983.

  The main idea behind simulated annealing is to mimic the process of annealing in metallurgy, where a material is heated and slowly cooled to reach a low-energy state. In the context of optimization, the algorithm starts with a high-temperature solution space, and gradually cools the system to explore the solution space and converge to a near-optimal solution.

  Advantages of Simulated Annealing:

- Simulated annealing is a flexible algorithm that can be applied to a wide range of optimization problems.
- Simulated annealing is able to handle problems with nonlinear objective functions and constraints.
- Simulated annealing can be easily parallelized, which can improve its performance for large-scale problems.
- Simulated annealing is able to escape local optima by allowing for occasional uphill moves during the search process.
- Simulated annealing is a stochastic algorithm, which means it can be used to explore the solution space and identify multiple potential solutions.

Disadvantages of Simulated Annealing:

- Simulated annealing requires careful tuning of its parameters, such as the cooling schedule and acceptance criteria, to ensure good performance.
- Simulated annealing can be sensitive to the choice of initial solution, which can impact its ability to find the global optimum.
- Simulated annealing may require a large number of iterations to converge, which can make it computationally expensive for large-scale problems.
- Simulated annealing is a probabilistic algorithm, which means it may not always produce the same results for the same problem instance.
- Simulated annealing may converge to a suboptimal solution if the cooling schedule is not chosen properly.

- Particle Swarm Optimization- Particle Swarm Optimization (PSO) is a metaheuristic algorithm used to solve optimization problems, such as the traveling salesman problem. It was first introduced by Kennedy and Eberhart in 1995.

  The main idea behind PSO is to mimic the behavior of a flock of birds or a school of fish. The algorithm maintains a swarm of particles, each representing a potential solution, and updates the particles' velocities and positions based on the position of the best-performing particle and the swarm's overall best position.

  Advantages of Particle Swarm Optimization:

- PSO is a simple and easy-to-implement algorithm that can be applied to a wide range of optimization problems.
- PSO is able to escape local optima by exploring the solution space through its swarm-based search strategy.
- PSO is a population-based algorithm, which means it can be used to identify multiple potential solutions.
- PSO can be easily parallelized, which can improve its performance for large-scale problems.
- PSO is a stochastic algorithm, which means it can be used to explore the solution space and identify multiple potential solutions.

  Disadvantages of Particle Swarm Optimization:

- PSO requires careful tuning of its parameters, such as the inertia weight and the acceleration coefficients, to ensure good performance.
- PSO can be sensitive to the choice of initial swarm and its parameters, which can impact its ability to find the global optimum.
- PSO may require a large number of particles to converge, which can make it computationally expensive for large-scale problems.
- PSO may converge to a suboptimal solution if the swarm becomes trapped in a local optimum or oscillates around the global optimum.
- PSO may have difficulty handling constraints and non-convex objective functions.

We are working on the disadvantages of the above algorithms and improving these algorithms using other metaheuristic algorithms as a tool.

## 2.2 SWOT Analysis

Strength: The utmost strength of our project is that it is a real-time load balancing system which could help us to find the better load balancing algorithm for user defined configurations. It helps to compare load balancing algorithms on various parameters like data center response time, data center process time, request servicing time. It could even tell us the cost of VM, memory, data transfer, and storage usage. Hence, we could predict which algorithm will help in better load balancing.

Weakness: The weakness of our project is that we need to run the program again for the new configuration once run the simulation. Also, the log report can not be saved. For the project need JDK 8 along with cloud-sim 2.1.

Opportunities: It can have wide range of valid possibility that this project can evolve into some bigger testing environment so to use the appropriate load balancing algorithm so that maximum utilization of resources can be done. This project can also be used for educational purposes, so that students can compare various algorithms and observes their behavior in different circumstances.

Threats: There is no threat for our project since it is an offline real-time analysis tool. It gives output according to the configuration setup by the user. But if someone alter the cloud-sim library they results may change and we could not get the efficient load balancing algorithm

## 2.3 Project Features

The main function of our project is making meta heuristic load balancing algorithms which is improved by the characteristic of other load balancing algorithm. These will run in real-time using the Cloud analyst which is cloudsim based GUI tool used for modeling and analysis of large-scale cloud computing environment. In our project we could check for the given scenario which load balancing algorithm could perform best on the basis of data center response time, service time, service processing time. Through this simulation we can easily compare each meta heuristic algorithm can choose best suitable one. Also, we could even generate the cost of VM, data transfer, memory, and storage cost. Here user just need to configure the data center configuration, user base, application deployment configuration, choose the load balancing policies and then run the simulation. The user will get information of each parameter graphically as well.

## 2.4 Design and Implementation

Algorithm 1: Honeybee + Simulated Annealing

- What is Honeybee?
  The Honeybee algorithm is a swarm intelligence optimization algorithm that is based on the behavior of honeybees. The algorithm is inspired by the foraging behavior of honeybees, where they communicate with each other to find the best food sources. In the algorithm, the population of solutions is represented by a colony of honeybees. Each honeybee represents a solution and communicates with other honeybees to find the best solution. The algorithm uses different parameters, such as the number

of scout bees, the number of employed bees, and the number of onlooker bees, to simulate the behavior of a real honeybee colony. The algorithm has been successfully applied to a variety of optimization problems, such as function optimization, parameter estimation, and image processing.

- What is Simulated Annealing

  Simulated Annealing is a metaheuristic algorithm used for optimization problems, especially in cases where finding the global optimum is challenging. The algorithm is inspired by the physical annealing process of heating and cooling metals to achieve a minimum energy state.

  In the simulated annealing algorithm, a random initial solution is generated and iteratively modified in a probabilistic manner, allowing the algorithm to escape local optima. The algorithm gradually decreases the "temperature" parameter, which controls the amount of randomness in the solution, and thus, the probability of accepting worse solutions.

  By doing so, the algorithm is able to explore a wide range of solutions in the early stages, and later on, converge towards a better solution. The simulated annealing algorithm has been successfully applied in various domains, including scheduling, logistics, and machine learning.
- Implementation

  The algorithm is based on the behavior of bees in search of food. The algorithm works as follows:

- The honeyBee class extends the VmLoadBalancer abstract class and implements the CloudSimEventListener interface. It also has some private variables such as initialTemperature, coolingRate, currentTemperature, cutoff, scoutBee, vmStatesList, vmAllocationCounts, and fitness.

- The honeyBee constructor initializes the vmStatesList variable with the list of available virtual machine states and registers the honeyBee class as a CloudSimEventListener.

- The getNextAvailableVm() method returns the ID of the next available virtual machine that can be allocated to a cloudlet. It first calls the getScoutBee() method to get the next available VM, then calls the allocatedVm() method to mark the VM as allocated. Finally, it updates the currentTemperature based on the coolingRate parameter.

- The cloudSimEventFired() method is called whenever a cloudlet is allocated or finished in a virtual machine. It updates the vmAllocationCounts and vmStatesList based on the current state of virtual machines.

- The isSendScoutBees() method checks if the current scout bee has reached its cutoff limit of VM allocations. If not, it returns false, and if yes, it returns true.

- The getScoutBee() method returns the index of the next scout bee. If there is no scout bee assigned, it returns the first VM. Otherwise, it checks if the current scout bee has reached its cutoff limit, and if yes, it calls the SendEmployedBees() method to update the fitness values. Then, it calls the SendOnlookerBees() method to get the best VM.

- The MemorizeBestSource() method returns the index of the VM with the best fitness value based on the waggleDance() method.

- The SendOnlookerBees() method returns the index of the best VM based on the MemorizeBestSource() method.

- The SendEmployedBees() method calculates the fitness values of all available virtual machines and updates the fitness map.

- The calculation() method is called by the SendEmployedBees() method. It calculates the fitness value of each virtual machine based on its current allocation count. It then performs the simulated annealing algorithm to find the next best VM to allocate to a cloudlet.

- The getNeighborSolution() method returns the index of the neighbor VM based on the current scout bee index.

- The method SendEmployedBees() calculates the fitness of each VM using the calculateFitness() method. In the employed bee phase, each employed bee chooses a random neighbor solution using the getNeighborSolution() method and checks if it has better fitness. If it has better fitness, the employed bee moves to the new VM. Otherwise, it calculates the acceptance probability of the new VM and chooses to move to it with a probability determined by the acceptance probability.

- Finally, the waggleDance() method selects the VM with the best fitness among all VMs found by the employed bees.

- SA Implementation

  The calculation() function implements the employed bee phase of the artificial bee colony algorithm to calculate the fitness values of each virtual machine (VM). Then, it uses a simulated annealing (SA) algorithm to explore the search space by moving from one solution (i.e., VM) to another.
  In the SA algorithm, the current solution is represented by the scoutBee variable, and the neighbor solution is represented by the getNeighborSolution() function. The fitness value of the current solution is represented by the currentSolutionFitness variable, and the fitness value of the neighbor solution is represented by the neighborSolutionFitness variable.
  If the neighbor solution is better than the current solution (i.e., deltaE < 0), then the neighbor solution is accepted by setting the scoutBee variable to the neighbor solution. Otherwise, the neighbor solution is accepted with a certain probability calculated using the SA acceptance probability formula. If the neighbor solution is accepted, the scoutBee variable is set to the neighbor solution.

- Advantages:

- Optimization: The SA algorithm can be used to optimize the placement of cloudlets onto virtual machines (VMs) by finding the best solution that minimizes the fitness function. The fitness function used in the code calculates the task length of each cloudlet and the capacity of the VM, and then assigns a fitness value to each VM based on this calculation.
- Efficiency: The SA algorithm can improve the efficiency of the load balancing process by ensuring that cloudlets are allocated to VMs in a way that minimizes the overall load on the datacenter. This is achieved by finding the best VM for each cloudlet based on the fitness function, and then allocating the cloudlet to that VM.
- Flexibility: The SA algorithm is a flexible algorithm that can be used in a variety of different scenarios. It can be adapted to different fitness functions and different optimization problems, making it a versatile tool for load balancing and optimization.

- Robustness: The SA algorithm is a robust algorithm that is not easily trapped in local optima. This means that it is able to explore a large search space and find global optima, even in complex optimization problems.

Algorithm 2: Ant colony + Tabu Search

- Ant colony

  Ant Colony Optimization (ACO) is a metaheuristic optimization algorithm inspired by the behavior of ant colonies. The idea behind ACO is to simulate the foraging behavior of ants in their search for food. Ants communicate with each other through chemical signals, called pheromones, which they deposit on the ground as they move. These pheromones attract other ants to follow the same path and reinforce it, making it more attractive for future ants.

  In the ACO algorithm, a set of artificial ants is used to explore the solution space of a given optimization problem. Each ant represents a possible solution and moves through the problem space by selecting a path based on the amount of pheromone present on each edge. The pheromone levels on each edge are updated at each iteration of the algorithm, with the aim of biasing future ants towards the best solutions found so far.

- Tabu Search

  Tabu Search is a metaheuristic optimization algorithm that is used to solve optimization problems. It is a type of local search method that uses a memory-based approach to avoid getting stuck in local optima. The basic idea behind Tabu Search is to maintain a "tabu list" of previously explored solutions or moves that are not allowed to be revisited in the near future. This helps the algorithm to avoid getting trapped in local optima by forcing it to explore a wider range of solutions.

  The Tabu Search algorithm starts by initializing a candidate solution and then iteratively searching the neighborhood of that solution by making small changes to it. Each time a new solution is generated, it is evaluated based on an objective function that quantifies the quality of the solution. The tabu list is used to store the recently explored solutions and the moves that were made to reach them. The algorithm then uses this list to avoid revisiting those solutions or making the same moves again.

- Implementation:

  - The AntColonyVmLoadBalancer class : is a subclass of the VmLoadBalancer class and implements the getNextAvailableVm method to allocate virtual machines (VMs) to physical hosts in a datacenter.

  - The algorithm starts by initializing: the pheromone matrix and creating a set of ants, which are used to explore the solution space. The ants move between VMs in the datacenter, updating the pheromone matrix as they go. The algorithm uses a tabu list to avoid assigning the same VM to a host multiple times in a row.

  - The Ant class represents an ant and contains methods for sending an ant to explore the solution space (SendAnt), fetching the final VM selected by the ant (FetchFinalVm), and processing the ant (ProcessAnt).

- The computeProbability method: computes the probability of selecting a VM as the next destination for the ant, based on the pheromone levels and a score function that takes into account the current VM and the potential next VM.

- The getNextVmNode: method selects the next VM for the ant based on the probabilities computed by the computeProbability method.

- The UpdatePheromone: method updates the pheromone matrix based on the ant's movement between VMs.

- The Evaporation method :is used to evaporate the pheromone levels, simulating the decay of pheromones over time.

- Tabu Implementation

  The tabu list is used in the getNextAvailableVm() method of the AntColonyVmLoadBalancer class. When this method is called, it first updates the tabu list by removing the oldest item if the list's size exceeds the tabu tenure limit. Then, it allows the ants to explore the solution space, during which the tabu list is passed to the ants. Once the ants have completed their exploration, the method finds the best available VM that is not in the tabu list and adds it to the list. Finally, the selected VM is allocated and returned.

  Thus, the tabu list is used to prevent revisiting the same VMs repeatedly, which can help to explore different parts of the solution space and potentially find a better solution.

- Advantage:

  - Prevents cycling: The tabu list prevents the algorithm from getting stuck in a cycle of repeating the same moves.
  - Promotes diversification: By forbidding previously explored solutions, the tabu list encourages the algorithm to explore different regions of the search space. This promotes diversification, which can lead to finding more optimal solutions
  - More robust: The Tabu Search Ant Colony Optimization algorithm is more robust to changes in the problem landscape and can adapt to different problem types more easily.

Algorithm 3: Ant colony + Simulated Annealing

- Implementation

  - getNextAvailableVm(): This method first initializes the pheromone matrix, the ants, and the counter if the counter is equal to 0. Then, it creates a new Ant object to find the next VM using the current pheromone matrix. After that, it runs the SendAnt() method for each ant to build the solutions. Then, it runs the saAlgorithm() method to optimize the solution found by the ants using the simulated annealing algorithm. Finally, it allocates the new VM to the next available VM using the allocatedVm() method and returns the ID of the new VM.
  - saAlgorithm(int currentVmId): This method optimizes the solution using the simulated annealing algorithm. The method initializes the nextVmId variable to currentVmId, sets the temperature and cooling rate, and initializes a random number generator. The while loop iterates until the temperature is less than 1e-8. In each iteration, it generates a random neighbor using the getNextVmNode() method, calculates the energy difference between the current and neighbor solutions using the scoreFunction() method, and decides whether to move to the

neighbor solution or not based on the temperature and the energy difference. After each iteration, it decreases the temperature. Finally, it returns the ID of the selected neighbor.

- Evaporation(): This method evaporates the pheromone trail by dividing each element in the pheromone matrix by the EVAPORATION_FACTOR.
- Ant(double[][] ph): This method initializes an Ant object with a given pheromone matrix and sets the fakeVmId to the last index of the pheromone matrix.
- SendAnt(): This method sends an ant to explore the pheromone trail and build a solution. It calls the ProcessAnt(true) method, which updates the pheromone trail.
- FetchFinalVm(): This method returns the last VM visited by the ant without updating the pheromone trail. It calls the ProcessAnt(false) method.
- ProcessAnt(boolean updatePheromones): This method sends an ant to explore the pheromone trail and build a solution. It starts with the fakeVmId and uses the getNextVmNode() method to find the next VM until it reaches the fakeVmId again. It updates the pheromone trail if updatePheromones is true. Finally, it returns the last VM visited by the ant.
- getNextVmNode(int vmId): This method returns the next VM to visit using the pheromone matrix and the probability distribution. It first computes the probability distribution using the computeProbability() method, generates a random number between 0 and 0.5, and uses the probability distribution to select the next VM. If all the VMs have been visited, the next available VM is selected based on the VM with the highest pheromone level. This is done to ensure that all the VMs are visited at least once and the algorithm doesn't get stuck in a loop.

- Advantage:
  - Escape Local Optima: SA can help the algorithm avoid getting stuck in a local optimum by allowing it to sometimes move to a worse solution with a certain probability. This way, the algorithm can explore the search space more thoroughly and potentially find a better solution.
  - Tolerance for Noise: SA is tolerant to noise in the fitness landscape. This means that the algorithm can still perform well even if the fitness values for different solutions are not exact or have some degree of randomness.
  - Convergence to Global Optimum: SA guarantee of converging to the global optimum if the temperature is cooled down sufficiently slowly. This means that if the algorithm is run for a sufficiently long time, it will eventually find the best solution.
  - Flexibility: SA can be easily adapted to different optimization problems by changing the cooling schedule, the score function, and the way neighbor solutions are generated. This makes it a versatile algorithm that can be used in a variety of scenarios.

Algorithm 4: Honeybee + Particle Swarm Optimization

- Particle Swarm Optimization

  Particle Swarm Optimization (PSO) is a metaheuristic optimization algorithm inspired by the collective behavior of social animals, such as birds flocking or fish schooling. In PSO, a population of candidate solutions, called particles, are randomly initialized in a search space, and are then updated

in each iteration based on their own previous best known position and the global best known position of the swarm.
Each particle has a position and a velocity, which are updated based on its previous position, its velocity, and the best-known positions of other particles in the swarm. These updates are guided by two key factors: the particle's own experience (i.e., its own previous best position) and the experience of the swarm (i.e., the best position found by any particle in the swarm). The algorithm seeks to find the optimal solution by iteratively adjusting the velocity and position of each particle, with the goal of converging towards the global optimum.
PSO has been widely applied to various optimization problems, including function optimization, parameter estimation, and feature selection, among others.

- Implementation

- honeyBee(): This is the constructor of the honeyBee class. It takes a DatacenterController object (dcb) as input and initializes several instance variables, including the vmStatesList, vmAllocationCounts, and fitness maps. It also adds the honeyBee object as a listener for cloudlet-related events fired by the data center.

- getNextAvailableVm(): This method overrides the getNextAvailableVm() method of the VmLoadBalancer abstract class. It selects a VM to allocate the next cloudlet to using the getScoutBee() method, which uses the PSO algorithm. It then updates the state of the selected VM in the vmStatesList map and returns the ID of the selected VM.

- cloudSimEventFired() :This method implements the CloudSimEventListener interface and handles two types of events: EVENT_CLOUDLET_ALLOCATED_TO_VM and EVENT_VM_FINISHED_CLOUDLET. When a cloudlet is allocated to a VM, the method updates the vmAllocationCounts map to keep track of how many cloudlets are assigned to each VM. If the number of assigned cloudlets for a VM exceeds the cutoff value, the method updates the state of the VM in the vmStatesList map to busy. When a cloudlet finishes executing on a VM, the method updates the vmAllocationCounts and vmStatesList maps accordingly.

- isSendScoutBees(int scoutBee): This method checks whether the selected scoutBee VM (identified by its ID) should be replaced by a new one in the PSO algorithm. If the number of cloudlets assigned to the scoutBee VM is less than the cutoff value, it returns false (meaning that the scoutBee should not be replaced). Otherwise, it returns true (meaning that the scoutBee should be replaced).

- pso(): This method implements the PSO algorithm to select the best VM to allocate a new cloudlet to. It first initializes a swarm of particles, each representing a candidate solution (i.e., a VM to allocate the cloudlet to). It then evaluates the fitness of each particle (i.e., the quality of the solution), updates the best positions found so far by each particle, and updates the global best position found so far. Finally, it updates the positions and velocities of each particle based on the best positions found so far and continues the loop until a maximum number of iterations is reached. The method returns the best position found, which corresponds to the ID of the best VM to allocate the cloudlet to.

- Particle class: This is a nested class that represents a particle in the PSO algorithm. Each particle maintains its current position and velocity in the search space, as well as the best position and fitness value found so far. The Particle class has two constructors, one for initializing a new particle with a random position and velocity, and one for initializing a particle with a given position and fitness value.

- getScoutBee(): This method returns the ID of the VM (Virtual Machine) that will be used as the food source for honeybees. The method first checks if the scoutBee variable is -1. If yes, it means that the food source needs to be determined using the Particle Swarm Optimization (PSO) algorithm. If scoutBee is not -1, it checks if the current food source (VM) can be used. If yes, it returns the scoutBee

VM. Otherwise, it invokes the SendEmployedBees() method to update the food source using the employed bees' information and returns the VM selected by SendOnlookerBees() method.

- SendEmployedBees(): This method is used to update the food source using the information gathered by the employed bees. It loops through all the VMs and updates their fitness value by invoking the calculateFitness() method. If the fitness value of a VM is higher than the current food source, it replaces the food source with the VM.

- SendOnlookerBees(): This method is used to select a new food source using the information gathered by the onlooker bees. It first calculates the total fitness value of all the VMs by looping through them and invoking the calculateFitness() method. Then, it loops through all the VMs and selects a VM with a probability proportional to its fitness value. It returns the selected VM as the new food source.

- calculateFitness(): This method calculates the fitness value of a VM based on the number of cloudlets allocated to it. The higher the number of cloudlets allocated to a VM, the lower its fitness value. The formula used to calculate the fitness value is: fitness = 1 / (1 + countCloudlets). The method takes the number of cloudlets allocated to a VM as input and returns the calculated fitness value.

- Advantage

- PSO provides a solution by exploring the solution space with a swarm of particles moving towards the global best solution. This helps in achieving a good balance between exploration and exploitation of the solution space.
- PSO does not require gradient information to optimize the problem. It can handle nonlinear and non-differentiable problems, which makes it suitable for solving complex optimization problems.
- PSO has the ability to escape local optima due to the random movement of particles. This means that it can find the global optimal solution, not just the local optimal solution.
- PSO is parallelizable, which makes it suitable for distributed computing environments. The algorithm can be easily scaled up to handle large-scale optimization problems.

Algorithm 5: Honeybee + Tabu

- Code implementation

  - import: The required packages are imported. These packages are necessary for the functioning of the program.
  - honeyBee: This is the constructor of the honeyBee class. It initializes the instance variables cutoff, scoutBee, vmStatesList, vmAllocationCounts, fitness, tabuList, and tabuTenure to their default values. It also sets the vmStatesList to the list of virtual machine states obtained from the DatacenterController object and registers the honeyBee object as a listener for CloudSim events.
  - getNextAvailableVm: This method is used to get the next available virtual machine (VM) for processing a cloudlet. It first calls the getScoutBee method to get a VM that is not on the tabu list. If the VM obtained is already on the tabu list, it calls the SendEmployedBees method to find a better VM. Finally, the method calls the allocatedVm method to mark the VM as allocated and returns the VM ID.

- cloudSimEventFired: This method is called whenever a CloudSim event is fired. It checks whether the event is EVENT_CLOUDLET_ALLOCATED_TO_VM or EVENT_VM_FINISHED_CLOUDLET and updates the vmAllocationCounts and vmStatesList accordingly.
- isSendScoutBees: This method is used to check whether a scout bee should be sent or not. It checks whether the vmAllocationCounts of the given VM is less than cutoff.
- getScoutBee: This method is used to get the next VM to be used by a scout bee. If scoutBee is -1, it returns 0 if the vmStatesList is not empty, else it returns -1. If scoutBee is not -1, it checks whether the VM is on the tabu list. If not, it checks whether the VM should be sent as a scout bee or not. If the VM should be sent as a scout bee, it calls SendEmployedBees method to find a better VM.
- MemorizeBestSource: This method calls the waggleDance method to get the best VM available.
- SendOnlookerBees: This method returns the result of calling the MemorizeBestSource method.
- calculation: This method is used to calculate the fitness value for each VM. It first clears the fitness map and then calculates the fitness value for each VM by calling the calculateFitness method.
- calculateFitness: This method calculates the fitness value of a VM using the formula solValue = 1/(1000-solValue).
- SendEmployedBees: This method is used to find the best VM using employed bees. It first calculates the fitness value of each VM by calling the calculation method. Then, it finds the VM with the best fitness value that is not on the tabu list. If a VM is found, it adds it to the tabu list and removes the oldest VM from the list if the list size exceeds tabuTenure.
- waggleDance: This method is used to find the best VM using the waggle dance method. It finds the VM with the lowest fitness value and returns its ID.

- Advantages

  - Diversification: Tabu search prevents the algorithm from getting stuck in local optima by keeping track of previously visited solutions and preventing revisits. This promotes diversification and exploration of the search space.

  - Intensification: The algorithm maintains a list of recently explored solutions, called the tabu list, and avoids revisiting them. This encourages the search to focus on promising solutions that have not yet been explored.

  - Flexibility: Tabu search allows for flexible and customizable implementation by incorporating different types of memory structures, such as the tabu list and the fitness values, which can be adjusted to suit the specific problem being solved.

  - Efficiency: Tabu search is known to be an efficient and effective algorithm for solving combinatorial optimization problems, such as load balancing in cloud computing. The honeyBee algorithm can benefit from the efficiency of tabu search by finding optimal solutions in a shorter amount of time compared to other optimization algorithms.

Algorithm 6: Ant Colony + Cumulative Distributive Function

- Code Implementation

- The AntColonyVmLoadBalancer class extends the VmLoadBalancer class and overrides its getNextAvailableVm() method. It initializes a pheromones matrix, which stores the pheromone value between each pair of VMs. The algorithm uses Ant objects that traverse this matrix, updating the pheromone value along their path.

- The main algorithm steps in getNextAvailableVm() method are:

- Create Ant objects and send them to traverse the pheromones matrix.
- Evaporate the pheromone from the pheromones matrix using the evaporatePheromones() method.
- Calculate the final score of each VM using the Ant objects and select the next available VM with the highest score.
- Allocate the selected VM and return its ID.
- The Ant class represents an ant that traverses the pheromones matrix. It has a fakeVmId attribute, which is an integer value greater than the maximum VM ID. The ant starts from this node and moves to the next node based on the pheromone value and the score calculated using the scoreFunction() method. The ant updates the pheromone values using the UpdatePheromone() method as it moves along its path.

- The computeProbability() method calculates the probability of selecting each VM as the next node to visit. It uses the scoreFunction() method to calculate the score of each VM and the CDF of these scores to calculate the probability distribution. The getNextVmNode() method selects the next VM node using the probability distribution.

- Advantages

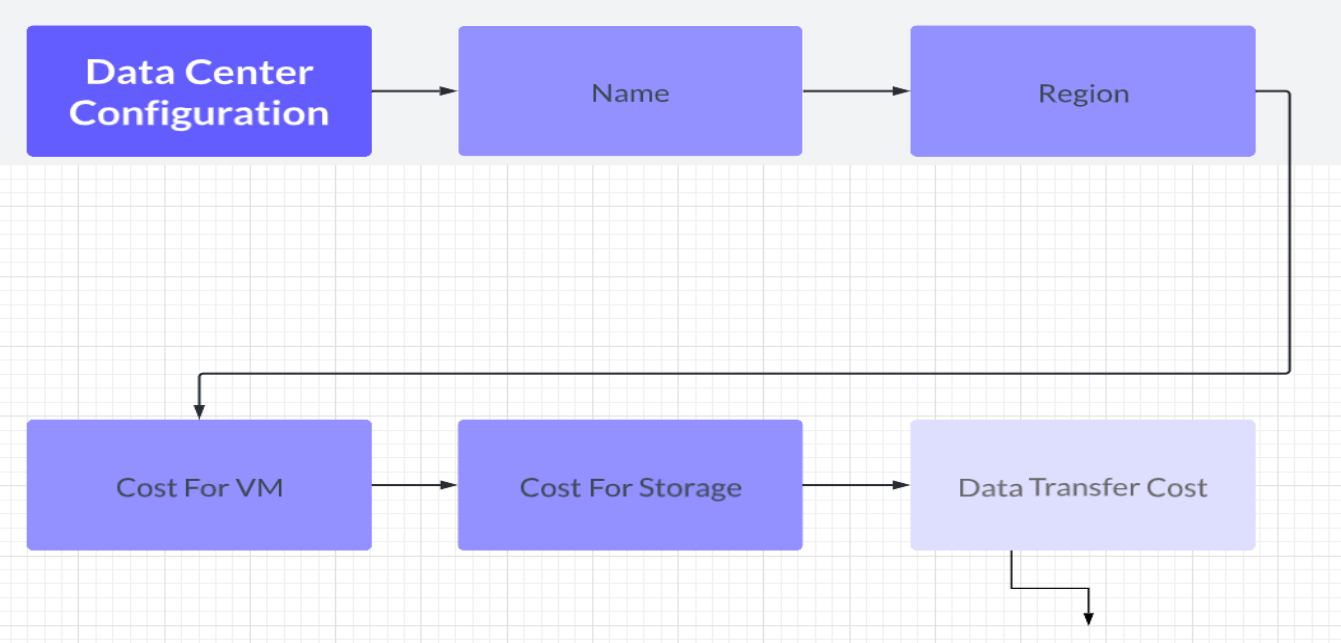
  - Better decision-making: The CDF provides a continuous probability distribution function that allows the algorithm to make a better decision when selecting the next VM for allocation. It calculates the probability of each VM's selection based on the fitness score function.

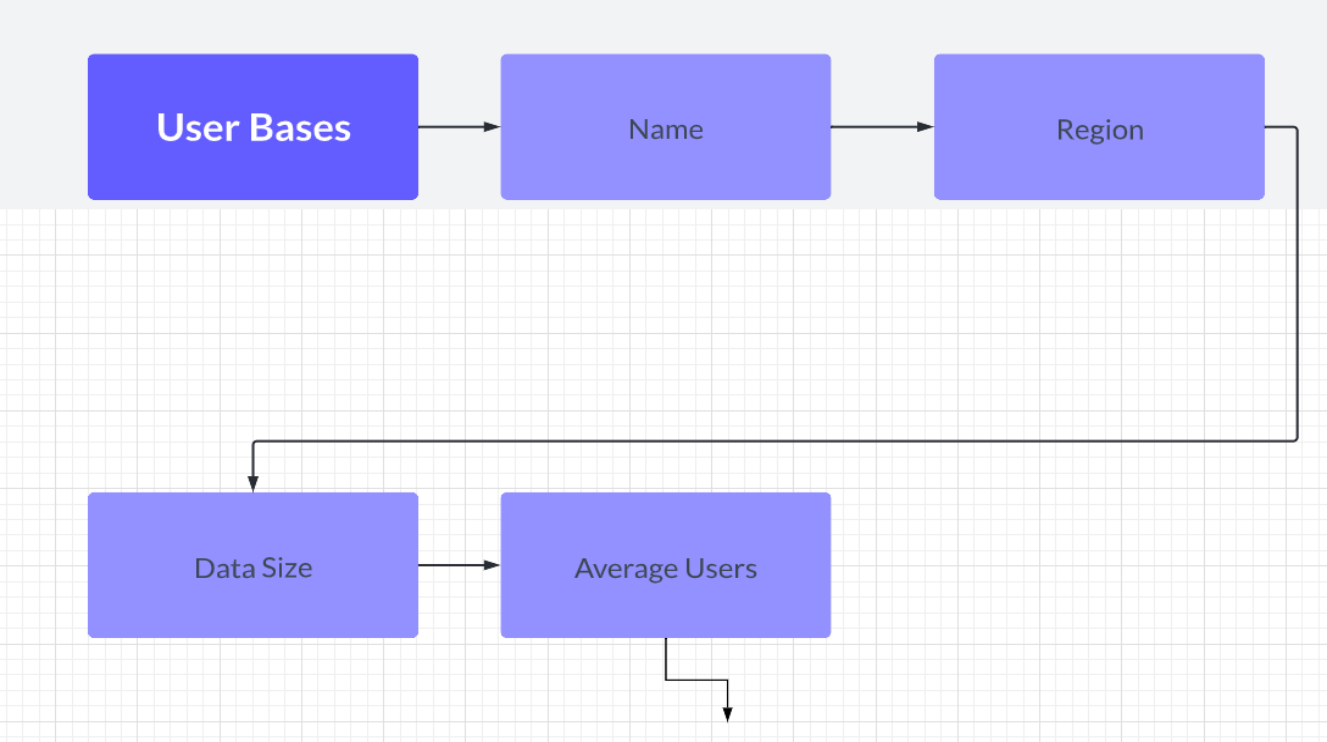
  - Efficient computation: The CDF uses the score function to calculate the probabilities of selecting each VM, which is computationally efficient as it avoids the need for a Monte Carlo simulation or other expensive techniques.

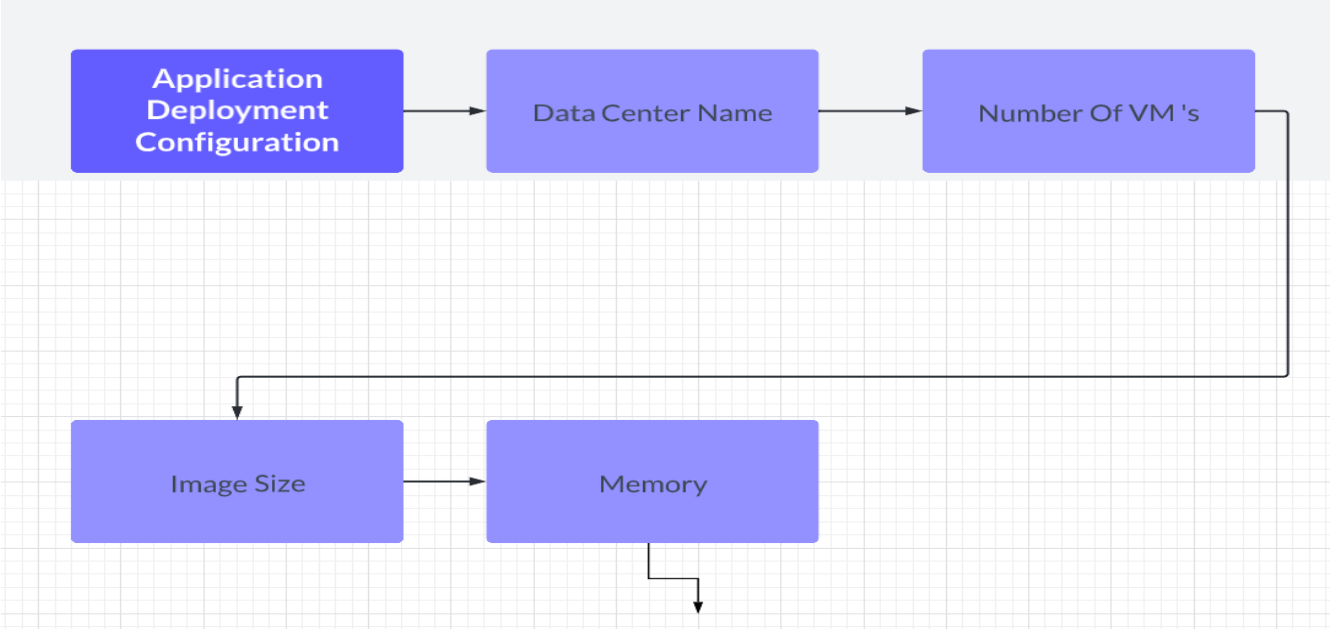
  - Improved convergence: The algorithm can converge to a better solution by using the CDF since it can capture the non-linearity of the search space more accurately. The CDF helps the algorithm to exploit good solutions and explore new ones more efficiently.

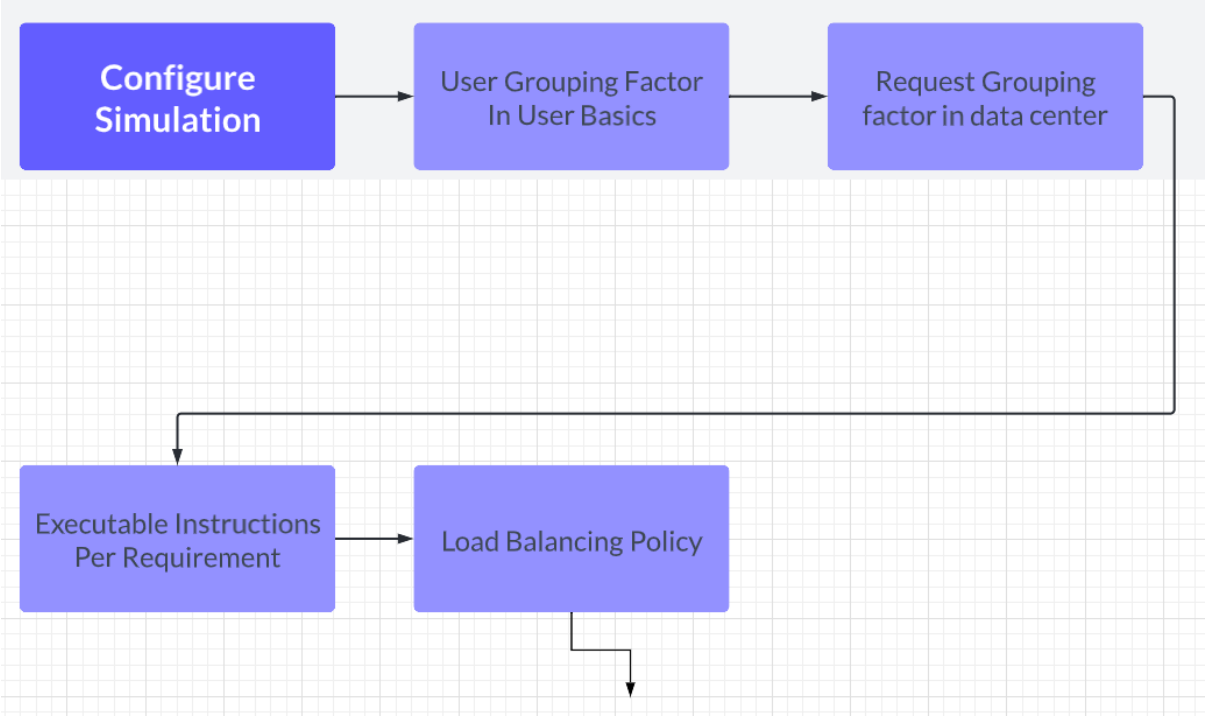
  By using the cumulative distribution function to calculate the probabilities, the algorithm becomes more efficient and more accurate in selecting the next virtual machine. The use of the cumulative distribution function reduces the number of iterations required to select the next virtual machine, which leads to faster load balancing.

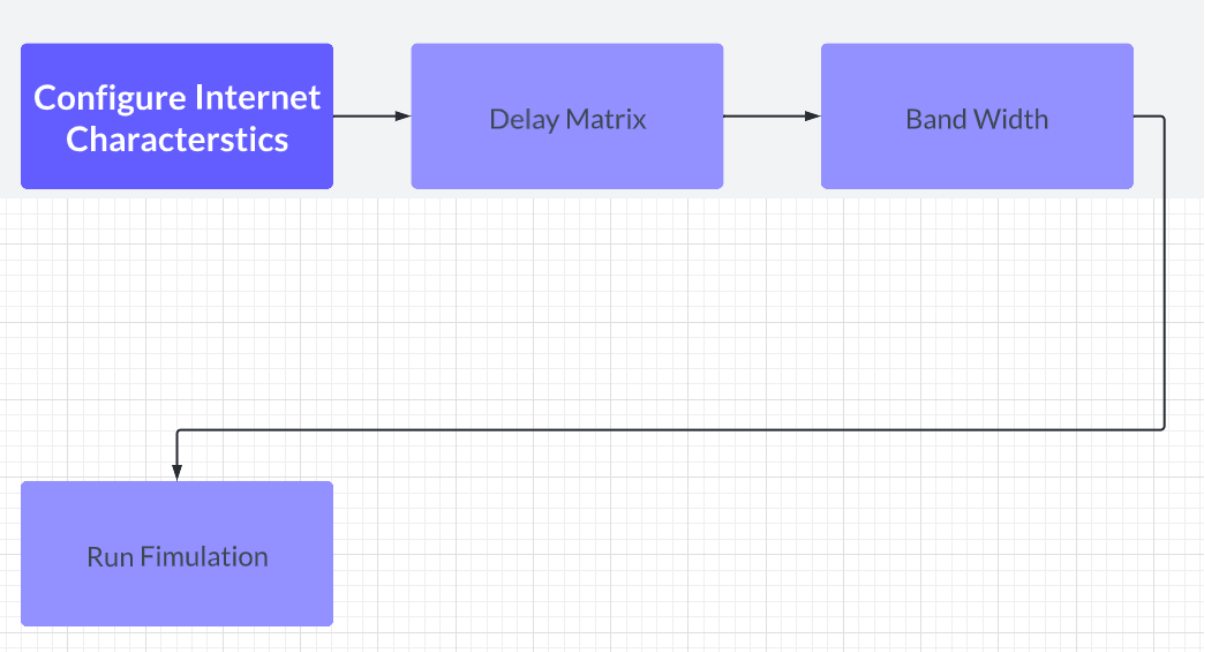
## 2.5 Design diagram


Data Center Configuration
Name
Region
Cost For VM
Cost For Storage
Data Transfer Cost


User Bases
Name
Region
Data Size
Average Users


Application Deployment Configuration
Data Center Name
Number Of VM 's
Image Size
Memory


Configure Simulation
User Grouping Factor In User Basics
Request Grouping factor in data center
Executable Instructions Per Requirement
Load Balancing Policy




## 3. SYSTEM REQUIREMENTS

### 3.1 User Interface

Cloud analyst, a cloudsim-based GUI tool used for modelling and analysis of a large-scale cloud computing system, is what we're employing for this project. Additionally, it makes it quick and simple for the modeller to run the simulation frequently while changing the settings. The GUI interface of the cloud analyst tool is shown in the diagram below. The configuration of simulation, definition of internet properties, and running of simulation are its three key menus. These menus are used to configure the simulation process as a whole. The tool gives us the option to switch algorithms based on our needs. Three data centres, DC1, DC2, and DC3, each with 75, 50, and 25 virtual machines, are used in the simulation setup, and the results are analyzed after a 60-minute run.

### 3.2 Protocols

Underlined Framework-:

Cloud Sim- A simulation toolkit called CloudSim enables the modelling and simulation of key cloud features like job/task queues, event processing, the development of cloud entities (datacenters, datacenter brokers, etc.), inter-entity communication, and broker policy execution. It is an open-source framework created by the CLOUDS Lab company and is totally written in Java.

## 4. NON-FUNCTIONAL REQUIREMENTS

### 4.1 Performance requirements

The following are the five basic requirements needed:

- Configuration of main tab (user region, requests, data size, service broker policy)
- Configuration and Control of Data center
- Configuration of load balancing policy across VM's
- Monitoring and Diagnostics
- Software Updates and Maintenance

### 4.3 Software Quality Attributes

Here are some of the security measures we intend to take for our project:

- Availability: Our project is highly available since depends upon the availability of Load balancing policy and cloud sim which are inbuilt in the program.

- Portability: Our project is light weight software which could work on any device (having JDK 8) hence portable.

- Usability: The usability of our project is very easy as proper GIU is used for each and every configuration. Also, Graphical representations are used for comparison of various algorithms which helps to achieve required goals effectively and efficiently. Our project provides a simple and friendly GUI keeping ease-of-use for users.

- Testability: Our project can be easily broken down into individual Load Balancing policies based on performance testing. Each policy used in the project can be individually tested and verified it's working.

## 5. References


[1] Ajay Jangra, Neeraj Mangla, "An efficient load balancing framework for deploying resource schedulingin cloud based communication in healthcare, Measurement: Sensors", Volume 25, 2023, 100584, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2022.100584 https://www.sciencedirect.com/science/article/pii/S2665917422002185

[2] Afzal, S., Kavitha, G. Load balancing in cloud computing – A hierarchical taxonomical classification. J Cloud Comp 8, 22 (2019). https://doi.org/10.1186/s13677-019-0146-7

[3] Sara Tabaghchi Milan, Lila Rajabion, Hamideh Ranjbar, Nima Jafari Navimipour, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments", Computers & Operations Research, Volume 110, 2019, Pages 159-187, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2019.05.022 https://www.sciencedirect.com/science/article/pii/S0305054819301352

[4] Suman Sansanwal, Nitin Jain, "An Improved Approach for Load Balancing among Virtual Machines in Cloud Environment", Procedia Computer Science, Volume 215,
2022, Pages 556-566, ISSN 1877-0509,
https://doi.org/10.1016/j.procs.2022.12.058
https://www.sciencedirect.com/science/article/pii/S1877050922021299

[5] Agrawal, Priyanka & Gupta, Subhash & Choudhury, Tanupriya. (2021). Load Balancing Issues in Cloud Computing. 10.1007/978-981-16-4149-7_10.

[6] Nuaimi, Klaithem & Mohamed, Nader & Nuaimi, M. & Al-Jaroodi, Jameela. (2012). A survey of load balancing in cloud computing: Challenges and algorithms. Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on. 137-142.

[7] Sidhu, Amandeep & Kinger, Supriya. (2005). Analysis of Load Balancing Techniques in Cloud Computing. INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY. 4. 737-741. 10.24297/ijct.v4i2C2.4194.

[8] Nuaimi, Klaithem & Mohamed, Nader & Nuaimi, M. & Al-Jaroodi, Jameela. (2012). A survey of load balancing in cloud computing: Challenges and algorithms. Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on. 137-142.

[9] Ren, Haozheng & Lan, Yihua & Yin, Chao. (2012). The load balancing algorithm in cloud computing environment. 925-928. 10.1109/ICCSNT.2012.6526078.

## APPENDIX A: GLOSSARY

- Load Balancing- Load balancing is the method of distributing network traffic equally across a pool of resources that support an application. Modern applications must process millions of users simultaneously and return the correct text, videos, images, and other data to each user in a fast and reliable manner. To handle such high volumes of traffic, most applications have many resource servers with duplicate data between them. A load balancer is a device that sits between the user and the server group and acts as an invisible facilitator, ensuring that all resource servers are used equally.

- Cloudsim- CloudSim is a simulation toolkit that supports the modeling and simulation of the core functionality of cloud, like job/task queue, processing of events, creation of cloud entities(datacenter, datacenter brokers, etc), communication between different entities, implementation of broker policies, etc. This toolkit allows to:

- Test application services in a repeatable and controllable environment.
- Tune the system bottlenecks before deploying apps in an actual cloud.
- Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques
- Cloud Analyst- Cloud Analyst is a tool that helps developers to simulate large-scale Cloud applications with the purpose of understanding performance of such applications under various deployment configurations.