ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2024.0485

SCHEDULING AND OPTIMIZATION OF RESOURCES IN EDGE COMPUTING USING RANDOM ALLOCATION AND MAX FIT ALLOCATION

P. Priya Ponnuswamy and C.P. Shabariram
Department of Computer Science and Engineering, PSG Institute of Technology and Applied Research, India

Abstract

The cloud computing is the new technique used most today for computation and storage. Edge computing is also a new computing domain that gives computation of tasks and storage of data nearer to the data sources. Various applications of edge computing are IOT, healthcare, retail, manufacturing etc. Virtualization technologies have enabled cloud platforms to provide various services such as virtual machines to the users. Task scheduling is the major issues faced in both cloud and edge domains. Since the usage of machines is high, it is difficult to assign tasks manually. So, an efficient and optimized algorithm is required in the cloud as well as the edge environment. Here, a task scheduling is implemented based on Heuristic Optimization Technique to examine the performance measures using cloudsim. Task scheduling and optimization techniques dynamically allocate resources in the cloud environment. Similarly, in edge computing, scheduling the task is performed with different optimization techniques such as random fit, and max fit algorithms. These task scheduling methods are allocated to the VM based on the resources that are available at the VM. These algorithms are implemented, and it increased the VM utilization by 4.82%. The processing time taken is reduced by 2.67% and the percentage of failed tasks is reduced by 3.68%. The experiment result shows the importance of the scheduling in the cloud and edge environment, in terms of processing time, failed task and increase in VM utilization in edge devices.

Keywords:

IoT, Healthcare, Random Fit, Max Fit, Heuristic optimization Technique, Task Scheduling

1. INTRODUCTION

Cloud computing and edge computing are categories of parallel and distributed systems which have interconnected computers. Cloud and edge platforms give consumers to access the virtual machines. Due to the more usage in cloud technologies, it is very challenging to assign jobs manually, and to compute resources in cloud and edge. As a result, an effective task scheduling algorithm is needed. An effective dynamic task scheduling algorithm in the cloud is Heuristic Optimization Technique(HOT), which is appropriate due to its adaptability. This algorithm shows how natural ant colonies interact with one another and search for food by leaving pheromones along travel paths. Certain modifications in the Heuristic Optimization are done to improve the probability of assigning tasks in the virtual machines. In cloudsim, there is no limitation for the number of tasks to be assigned, which ultimately takes more time. In edge cloudsim, a CPU utilization model is implemented where the number of tasks that run parallel in the virtual machine is limited. Task scheduling in edge computing refers to the allocation procedure of computational resources and scheduling tasks efficiently across edge devices to optimize performance. It involves determining which tasks should be executed on which edge devices based on factors such as device capabilities, network conditions, and task deadlines. Here to optimize scheduling tasks in edge computing, random allocation and max fit algorithms are being used.

The use of dynamic scheduling techniques in cloud and edge devices, scheduling the tasks that are inbound to the virtual machines. The VM's are allocated based on the optimization method and calculate the probability of the best suitable virtual machines. The probability for assigning the cloudlet to the virtual machine is found to find the best optimal solution. The existing algorithm changes with the requirements of the cloud environment but the impact of load balancing and task hierarchy needs to be considered in the future work. The tasks in the edge devices will be assigned to the nearest nodes for the computation using the optimization algorithm.

In order to improve efficiency and performance, optimization strategies that dynamically alter with respect to the changes in the scheduling environment are required to allocate the enormous number of tasks in an efficient and optimized way. The Heuristic Optimization Technique uses the foraging behavior of the ants to optimize the tasks. The optimization algorithm uses the probability calculation method to give the optimal result in order to assign a specific VM to a task. The optimization follows dynamic scheduling. As the computation in cloud gets difficult when there are few resources and huge number of tasks, the edge environment can be considered as a better way to assign the tasks to the optimal centers.

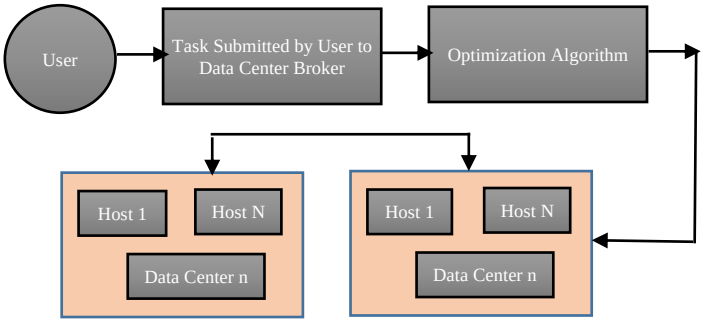

Fig.1. Cloud Task Scheduling Scenario

The Fig.1 explains the task scheduling scenario in the cloud, with number of host machine present in the data center. Here the specified optimization algorithm can be used in order to allocate the tasks to the appropriate edge devices. Fig.2 describes the edge task scheduling scenario where the tasks are given to the data center brokers. The given tasks are then assigned to edge nodes using the scheduling methods. For optimization in cloud environments, a Heuristic Optimization Technique algorithm is implemented.

1.1 OBJECTIVE

The goal of the work is to implement the scheduling and optimization algorithms in cloud and edge environments.

- Implementation of Probability based HOT in the cloud Systems
- The performance of the Heuristic Optimization Technique are analyzed with number of devices, number of VM's and average processing time
- Leveraging edge computing to schedule the resources to nearer edge devices using Random allocation, Min fit, Successor fit and Max fit.
- Assessing and analyzing the performance of recent development in task scheduling algorithm in the edge environment particularly concentrating on failed task, task load, resource utilization of VMs and processing time.

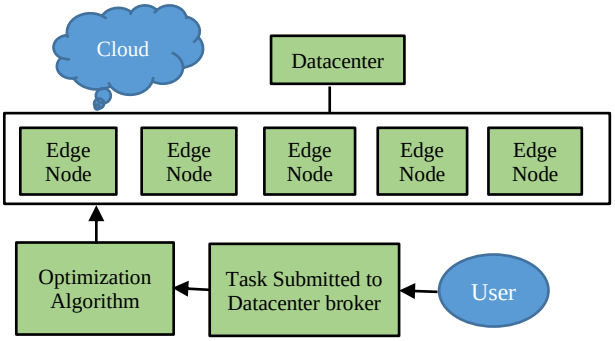

Fig.2. Edge Task Scheduling Scenario

For better scheduling in edge environments Ant-Colony-Optimization, Random allocation scheduling algorithm and Max fit scheduling algorithm are used in the edge environment. The scheduling of tasks is done using the optimization algorithm and performance analysis is measured for the same. The tasks are assigned in the edge environment based on the optimization algorithm and reduce the process time in the edge systems.

1.2 CONTRIBUTION

- Implementation of Probability based HOT in the cloud environment- Modified Heuristic Optimization Technique (HOT) is implemented in cloud environment. The algorithm follows the principle how natural ant colonies interact with one another and search for food by leaving pheromones along their travel pattern. Like pheromones in real life, here probability is calculated for assigning virtual machine to a cloudlet. The global best probability is stored in place of a pheromone value and the tasks are given to the best suitable VM.
- To analyze the performance of the Heuristic Optimization Technique, different parameters such as number of VM utilized in the cloud devices, number of tasks in the cloudlets, Processing time, scheduling time and number of cloudlets are considered. The size of the virtual ant colony is controlled by several parameters, one of which is the number of ants. Performance of the algorithm can be enhanced by boosting the ant population, but computational overhead rises as well.
- Implement the Random allocation and Max fit algorithm in edge environment - Performance of edge computing systems, which involve distributing computational tasks across a network of edge devices, is greatly improved by scheduling methods.
- The effectiveness of the scheduling techniques in the edge devices are analyzed based on number of failed tasks, the availability of devices, task load, and resource utilization in random allocation and max fit algorithms.

2. LITERATURE REVIEW

Gupta et al. [1] have proposed a Scheduling based on Cloud using HOT and Load Balancing. The proposed approach aims to balance the load that is spread among the virtual machines. The paper highlights the challenges faced for load balancing in cloud computing and offers a comparative evaluation of various load balancing algorithms. The outcomes indicate that the HOT-based approach performs far better than other algorithms with respect to task completion time and resource utilization. Thus, the paper presents load balancing about cloud computing that could improve the overall performance of project scheduling in cloud environments. Shi and Shiet [2] have proposed that multi-objective optimization is used for multi-node scheduling in the edge devices. The nodes are dynamic in higher orders and the resources are not balanced in edge computing. Because of this factor, when a task is scheduled, the resources availability needs to be calculated. The impact time with respect to completion, load balancing on task scheduling is considered while developing this optimization technique. Two different scenarios are considered where the edge node shows sensitivity towards real time performance and expects to reduce the overhead. The multi objective optimization, makes sure that the tasks are scheduled in an evenly manner. Reddy et al. [3] have proposed a Scheduling in Cloud Environment. A local search based on the pheromone trail and heuristic data is then used to improve the first solution that the algorithm generated using a constructive heuristic. This paper assumes that the resources are equal and that the resources have no effect on how quickly tasks are performed. For some cloud computing environments, this may not be a valid assumption. Sun et al. [4] Based on the concept of load balancing, a scheduling strategy is proposed. Business wait queue model and server resource scheduling model are used. The scheduling of tasks is done based on the standard deviation. Standard deviation of the tasks to be computed is used for grouping up the tasks and to determine the proportion of services. A secondary allocation ideology is used where a heavy load task is assigned to a light load resource for the scheduling to take place efficiently. The algorithm has achieved the effect of decreasing the load balance in the server and improving the resource optimization rate. Parvesh Humane et al. [5] proposed the practical method for the simulation of cloud systems via CloudSim simulator. The paper provides a summary of the CloudSim simulator's key features, including its capacity to simulate all components used in cloud. The paper presents a case study where they use CloudSim to simulate a cloud-based e-commerce application. The performance of the application is analyzed under different scenarios, such as varying workload, resource allocation, and scheduling policies. The experimental results show how effective CloudSim is for modeling and assessing cloud systems. The paper also depicts some restrictions and difficulties associated with using CloudSim, such as the need for accurate modeling of the underlying hardware and software elements and the difficulty of accurately capturing the dynamic nature of cloud environments. Zhang et al. [6] proposed the paper about energy efficiency aspects of resource

allocation in edge computing. The techniques such as dynamic voltage scaling, sleep mode operation, and workload consolidation to minimize energy consumption. The paper reviews various resource allocation algorithms, including dynamic programming, game theory, and machine learning approaches. It investigates the trade-offs between latency, energy consumption, and resource utilization in resource allocation decisions. It presents energy-aware resource allocation algorithms and optimization frameworks that consider both performance and energy efficiency objectives, contributing to sustainable and environmentally friendly mobile edge computing deployments. Razaque et al. [7] proposed an algorithm in consideration with the availability of the network bandwidth in addition to the resources needed, execution time and cost, CPU memory, etc. This paper discussed the algorithm used for the divisible task scheduling model in order to maintain the accuracy in assigning the tasks to every virtual machine. Lakra et al. [8] proposed a multi objective algorithm in the cloud environment. The paper is based on the QoS parameters where the tasks and VMs were assigned based on the same. The algorithm demonstrates that the suggested method performs FCFS and priority scheduling techniques. This paper also proposes a method to improve this algorithm by taking in consideration some other QoS parameters. Medhat A. Tawfeek et al. [9] presented a system based on Ant-Colony-Optimisation for the cloud environment. This paper explains the need for change in adaptation of planning approach to the environment's shifting needs. The algorithm has achieved the effect of decreasing the load balance in the server and improving the resource optimization rate. In the future, the impact of task hierarchy and load balancing will be viewed as an improvisation. Hu et al. [10] The proposed methodology enhances the conventional HOT algorithm by considering the dynamic characteristics of cloud computing. The algorithm is founded on the notion that parameters can self-adjust to the dynamic changes in the cloud environment. By cutting down on communication costs and time delays between nodes, the proposed methodology improves work scheduling. Hence enhancing the overall efficacy of job scheduling in cloud computing. Satyanarayanan et al. [11] proposed the paper and gives in- depth overview of mobile edge computing architectures and computation offloading techniques. The paper discusses the benefits, challenges, and practical implications of offloading work from mobile devices to edge servers. It covers various architectural models, including hybrid, cloudlet-based, and fog-based architectures, and evaluates their benefits and drawbacks. It examines several Task divisions, workload prediction, and decision-making algorithms. The survey also examines the impact of network latency, bandwidth, and mobility on computation offloading decisions. The paper provides case studies and actual mobile edge computing applications, illustrating the potential for enhanced performance and decreased energy use. Optimisation is suggested to address the DWRR algorithm's latency issue with task execution. With regard to both these algorithms a fuzzy logic was developed in the cloud environment. It further aims at optimizing the make span and the time taken to schedule the task. Gao et al. [13] Cloud computing makes use of a dynamic work Ant-Colony-Optimisation-based scheduling method. The suggested method aims to improve system throughput while reducing response time and energy usage. The proposed methodology makes use of a dynamic pheromone update mechanism that takes into consideration how the cloud environment changes. Zheng Shi et al. [14] have suggested a technique for edge devices. The proposed methodology considers more objectives, like accuracy, latency etc. to determine the optimal configuration of an edge device. Experimental methods shows that the given methodology outperforms the baseline approach, particularly in terms of energy efficiency. The authors also analyze the sensitivity of the optimization results to different target weights and demonstrate the applicability of the approach to different device types and peak loads. Mao, Y., You, C al. [15] proposed the paper and examines the difficulties and advantages of implementing edge computing in mobile networks, including the reduction of latency, increased energy efficiency, and improved user experience. To provide secure and private edge computing operations, it thoroughly investigates data encryption, access control, and authentication methods. The paper provides insights into various edge architectures, cloudlet, and mobile edge computing. It also evaluates the strategies for managing resources like work dumping, assigning resources, and load distribution on network performance. The paper emphasizes numerous edge computing in mobile networks applications and use cases, such as real-time analytics, augmented reality, and Internet of Things (IoT) deployments.

3. SYSTEM MODEL

A sizable user base**,** a wide range of application task types, and a big scale of different resources are all features of the cloud computing and edge computing systems. It becomes difficult to manage these circumstances when there are many users and minimal resources. It can be difficult to schedule non-dynamic algorithms when there are lots of users and minimal resources available. In order to distribute the vast number of jobs in an effective and optimum manner, optimization algorithms that dynamically adapt with respect to changes in the scheduling environment are necessary. The Heuristic Optimization Technique optimizes task scheduling by mimicking the foraging behavior of ants. To provide the ideal solution in the optimization techniques, the probability of allocating task is important. Dynamic scheduling is followed by optimization. The edge environment can be taken into consideration as a better method for resource allocation or task scheduling because cloud computation becomes challenging when there are few resources and many jobs.

3.1 SYSTEM BLOCK DIAGRAM

The system flow comprises two blocks namely the optimization algorithm used in the cloud scenario and the scheduling methods in the edge environment. The user submits the tasks where the different scheduling methods such as Random allocation, Min fit, Successor fit and Max fit are present.

The Fig.3 explains the flow where a task assigned by the user goes to the host via a datacenter broker. In the cloud environment, the tasks are optimized by the optimization algorithm. The best optimal VM is chosen for the cloudlet assignment based on the global updated probability. Five different scheduling methods are used in the edge to allocate the task to the edge devices. The task is then processed based by the scheduling algorithm according to the type.

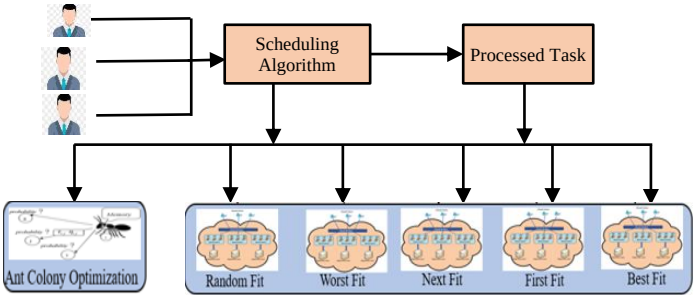

Fig 3 System Block Diagram

The HOT is the use of positive feedback mechanisms, represented by pheromone trails, to guide the search process. Ants probabilistically select edges to traverse based on the combination of pheromone levels and heuristic information, which encodes problem-specific knowledge. The pheromone trails left by ants on the edges serve as a form of indirect communication, allowing them to exploit the experience of previous ants and bias their decisions. By iteratively constructing solutions and adjusting the pheromone levels based on the quality of solutions found, HOT converges toward better solutions over time. This exploration-exploitation balance allows HOT to effectively navigate complex solution spaces and find near-optimal solutions for combinatorial optimization problems. Max fit can be defined as assigning the largest available resource to perform a simple task. This way, the task can be processed within minimum time. The time taken to schedule the smallest task in the maximum resource is less compared to other algorithms. In Random allocation algorithm, an edge device is selected at random, and it is checked based on the resources it contains. If the device can process the incoming task, the task is allocated to the VM to perform.

4. IMPLEMENTATION

4.1 OPTIMIZATION USING TASK PROBABILITY

All data Centers available in the cloud need to register themselves with the cloud registry. The cloud datacenters are created whenever the VM's are required. Cloudlets have computational capabilities, including CPUs (Central Processing Units), memory, storage, and networking infrastructure. These resources allow the cloudlet to process and execute applications or services. In the proposed system, the mapping of VM's to the cloudlet is done by the probability calculation. An array called "task probability "is initialized to hold the probability of each cloudlet being assigned to a VM, and an array task to hold the VM index that each cloudlet is assigned to. Then a boolean array is initialized to track which VMs have already been considered for each cloudlet assignment. The best optimal VM is chosen for the cloudlet assignment based on the global updated probability.

4.2 HEURISTIC OPTIMIZATION TECHNIQUE

In HOT, a set of virtual ants iteratively constructs solutions to the problem at hand. Each ant probabilistically selects the next component or state based on pheromone trails deposited on the edges of the problem graph. The amount of pheromone deposited is influenced by the quality of the solution found by each ant. Over time, the pheromone levels on the edges evolve through local pheromone updating and global pheromone evaporation. Local updating enhances the pheromone strength of the edges used in constructing a solution, while evaporation reduces pheromone levels on all edges. This process allows the algorithm to converge towards better solutions by exploiting paths with higher pheromone levels. HOT can be used on the cloud system to overcome the issues of scheduling the user task. The objective is to allocate resources in a way that maximizes system performance while lowering costs and energy usage. In the Initialization phase, the cloud resources as a group of ants are dispersed at random. The ant movement is done when each ant travels from its present node to a neighboring node in accordance with a probability function dependent on the node's desirability and pheromone level. In the Resource distribution phase, when an ant arrives at a node, it distributes to its application with the resource connected to that node. In the Pheromone update phase, depending on how well the ant finds a decent allocation and the pheromone level of each edge is updated. The final phase is termination, the algorithm ends when a predetermined condition is satisfied, such as when the allotted number is touched, or a workable solution is discovered the resource allocation to the edge is terminated. Here the probability of allocating resources in Heuristic Optimization Techniques found by

$$tempProb = T[vm][task][t]^a \quad (1)$$

It then calculates the value of 'd_{ij}' as

$$d_{ij} = (Size\ of\ cloudlet) / (Processing\ capacity)\ (Cloudlet * VM) \quad (2)$$

Then the tempProb is calculated because of raising the value of a specific element T matrix which is the pheromone matrix to the power of a constant a which is the level at which the pheromones have an effect on the probability value. It then divides the result by d_{ij} and stores the result in 'tempProb' In order to enhance the probability calculation, the following changes have to be made.

$$x = x + T[vm][task][t]^a \quad (3)$$

$$x = x + Math.exp(a * T[vm][task][t]) \quad (4)$$

$$x = x + log\ (1 + a * T[vm][task][t]) \quad (5)$$

$$x = x + (a * T[vm][task][t]+1) \quad (6)$$

$$tempProb = x/(4*dij) \quad (7)$$

The probability to find the nearest VM is enhanced using the modification. By raising the probability, ants are more likely to choose less frequently visited or less favorable edges during solution construction. This enables further exploration, leading to the discovery of new paths and potential improvements. The increased probability promotes the diversification of ant behavior, ensuring that different paths and solutions are explored. This diversification helps prevent getting stuck in the local optima infinitely or choosing an optimal solution before even exploring other paths. By allowing for increased exploration and diversification, higher probability values in HOT contribute to a more thorough search process, potentially leading to the discovery of better and more optimal solutions.

4.3 MAX FIT AND RANDOM ALLOCATION IN EDGE DEVICES

Edge computing provides a greater number of data request or application request from different devices need to be scheduled appropriately to the edge devices. Three scenarios ae used in the edge computing which are 1-tier, 2-tier and 2-tier with Edge Orchestrator. In a 1-tier edge computing model, the edge devices

themselves perform both data processing and application execution. This means that the edge devices directly handle the computation tasks without offloading them to separate edge servers or cloud resources. The devices act as units that process, store, and respond to data locally. The 2-tier edge computing model introduces a separation of responsibilities between edge devices and edge servers. In this model, the edge devices at the network edge capture and processes the data locally. Allocation of task to a edge device is *EDi*, Workload of edge devices is *WDi* ,Storage capacity of edge devices is *SCEDi*, Computational capacity of edge devices is *CDEDi*.

Allocation of task based on the availability is considered as

$$XEDi = WDi \in (SCEDi+CDEDi)) \quad (8)$$

Task is allocated based on priority to the edge device is

$$PEDi = WDi \in (SCEDi + CDEDi) \& XEDi) \quad (9)$$

The number of task received for a particular edge device at a particular time is mentioned as

$$\delta T = n(EDi) \quad (10)$$

The task waiting to schedule in the queue in cloud and is given wait

$$(EDi) = XEDi>0 \text{ then } WDi \in (SCEDi+CDEDi) \rightarrow EDi \quad (11)$$

$$XEDi>0 \text{ then } WDi \in (SCEDi+CDEDi) \rightarrow \text{EDi} = \delta T\text{-}1 \quad (12)$$

The edge servers, which are typically more powerful than the edge devices, handle the heavy computation and can also provide additional services, such as caching, storage, and analytics. The 2-tier edge computing model can be enhanced by incorporating an Edge Orchestrator (EO). The Edge Orchestrator is responsible for managing and orchestrating the distribution of workloads between the edge devices and the edge servers. It acts as an intelligent intermediary that determines where each task or workload should be executed based on factors such as device capabilities, network conditions, application requirements, and resource availability. The application performance and allocation of task in Edge Orchestrator is better and it achieves the efficient resource utilization and application performance. It also provides on demand workload migration and balancing the work load across all the edge devices and edge servers. The inclusion of an EO adds an additional layer of intelligence and coordination to the 2-tier edge computing architecture. The Max fit algorithm plays a vital role in allocating tasks to edge devices based on the maximum available resources in the cloud system. The max fit algorithm calculated the resource need of the submitted tasks. The algorithm then checks the resources availability on the edge devices and cloud devices. The Random allocation algorithm are used to allocate tasks randomly to edge devices based on a random selection process. The edge device is selected randomly from the pool of available edge devices by the random allocation algorithm. This approach guarantees an equal and impartial distribution of tasks among the edge devices.

4.4 RANDOM ALLOCATION ALGORITHM IN EDGE ENVIRONMENT

A heuristic approach in allocating resources to edge devices is done with random allocation technique. It operates by choosing a resource at random from the pool of resources available and assigning it to a job that needs the resource. Until all jobs have been assigned to resources, this process is repeated. The Random Fit algorithm begins by collecting information about available edge devices and their capacities. It considers factors such as CPU capacity, memory, and other relevant resources.

- **Task Arrival**: When a cloudlet arrives in the system, the scheduler needs to find a suitable edge device for its execution. The random Fit algorithm selects an edge device randomly based on the steps

Algorithm for Random Fit

Input: Capacity of Task

Output: Time taken for a task to run on a VM

If Random Fit **then**

Int randIndex=simUtils.getRandomNumber(0,vmArray.size()-1)

Doublerequiredcapacity=((CpuUtilizationModecustom)task.getUtilization ModelCpu().predictUtilization(vmArray.get(randomIndex).getVMType());

doubletargetVMCapacity=(double)100- vmArray.get(randomIndex).

getCloudetScheduler().

getTotalUtlizationofCPU(CloudSim.clock());

if requiredCapacity<=targetVMCapacity **then**

electVM=vmArray.get(randomIndex);

end if

end if

- **Step 1:** Resource Comparison: The algorithm compares the capacity requirements of the cloudlet with the available capacity of the selected edge device. It checks whether the edge device has sufficient resources to accommodate the cloudlet.
- **Step 2:** Allocation Decision: If the selected edge device has enough capacity to handle the cloudlet, the scheduler assigns the task to that edge device for execution. Otherwise, it repeats the random selection process until a suitable edge device is found. Task
- **Step 3:** Execution: Once the task is allocated to an edge device, it starts executing on that device using the available resources. It aims to distribute tasks randomly across available resources without following a specific optimization criterion.

4.5 MAX FIT ALGORITHM IN EDGE ENVIRONMENT

The Max fit algorithm seeks to reduce resource fragmentation in an edge computing environment. It operates by choosing the resource with the biggest capacity that is available and assigning the task to that resource.

Algorithm for Max Fit

Input: Capacity of a Task

Output : Time taken for a task to run a VM

doubleselectedVMCapacity=0;

for VM index=0 to vmArray.size() **do**

doublerequiredCapacity=(CpuUtilizationModelCustom) task.getUtilizationModeCpu().PredictUtilization (vmArray.get(vmIndex).getVmType());

```
doubletargetVMcapacity=(double)100-vmArray.
get(vmIndex).getCloudletScheduler().getTotalUtlizationof
CPU(CloudSim.Clock());
if required Capacity <=target VMCapacity
                    > slelectedVMCapacity then
selectedVM=vmArray.get(vmIndex);
selectedVMCapacity=targetVMCapapcity;
end if
end for
```

Task arrival in max fir algorithm happens like Task Arrival: When a cloudlet arrives in the system, the scheduler needs to find a suitable edge device for its execution. The Max Fit algorithm aims to utilize the edge devices with the highest available capacity for task allocation. By doing so, it helps to balance the workload distribution and leaves smaller available capacities on edge devices for future tasks. The Max fit method has the advantage of reducing resource fragmentation, which can increase resource utilization and lower resource management costs. However, it might not always lead to the best resource distribution, particularly when resources have a wide range in size or when various jobs have varying resource needs.

4.6 LINEAR FIT, AND MIN FIT ALGORITHM IN EDGE ENVIRONMENT

Task scheduling in edge computing is about assigning different tasks or jobs to the right computers at the edge in an efficient way. When there are multiple devices and servers at the edge, the scheduler decides which tasks should be done by which devices. It considers things like how complex the task is, how much computing power each device has, how good the network connection is, and how quickly the task needs to be done. The scheduler tries to use the available resources wisely, make sure tasks are done quickly, and balance the workload across the edge devices. It uses smart algorithms and techniques to make the best decisions on where to send tasks so that the edge computing system works well, and applications perform smoothly. These algorithms are used to schedule the tasks to the resources accordingly. The Linear fit algorithm schedules the first highest resource to the incoming task, where there are chances for the other tasks to be resource deficit. The Min fit algorithm finds the optimal resource to be allocated to the task whereas the Max fit allocates the highest units of resource to the lowest needed task. The above-mentioned algorithms were already present in the edge simulator for assigning the tasks.

5. RESULTS AND ANALYSIS

The parameters used for analyzing the results are the number of devices, the number of cloud tasks, the processing time required in order for those devices to complete the task. The number of processors also have an effect in decreasing the processing time. The average processing time and average VM utilization are used in the edge environment to calculate the resource utilization with respect to the number of tasks. Percentage of failed tasks give us the number of tasks that cannot be successfully allocated to the resources.

The Table.1 shows the how the number of processors in a cloudlet interacts with the cloud devices and the time taken for scheduling tasks in a cloud environment. In a cloud computing context, this graph shows the time needed to complete a task in a virtual machine (VM) as a function of the task's processing requirements. Processing times grow when more processors are needed to complete a task because as the required number of processes increases for a task to run on a VM, then it means that it sometimes must wait for other tasks to free up the processors that other tasks have been using to run on a VM when required number of processes is not available. Thus more the number of processors required by a task, the time taken is directly proportional. This behavior is predictable given that the work calls for more processors, which in turn calls for more computing resources and longer processing times.

Table.1. Number of tasks(cloudlet) vs the time taken to schedule these tasks in HOT algorithm

Time Taken by VMs to Run a Task				
Number of Task/ Time Taken	**Time taken for 20 vm**	**Time taken for 30 vm**	**Time taken for 40 vm**	**Time taken for 50 vm**
10	270	180	177	173
15	390	200	185	178
20	460	205	185	178
25	-	220	190	180

Table.2. Number of devices vs Percentage of failed task in Max fit Algorithm

Percentage of Failed task in Max Fit			
Number of Devices/ Percentage of Failed Task (%)	**1-Tier**	**2-Tier**	**2-Tier with EO**
50	0	0	0
75	0.5	0.3	0
100	1	0.5	0.1
125	3.8	2.2	0.3
150	7.5	4.5	0.3
175	10.2	7.2	0.3
200	13	9.5	0.5

The Table.2 shows the relationship between number of devices and percentage of failed tasks in Max fit Algorithm. 1-tier, 2-tier, and 2-tier with EO entities are depicted in the table. The percentage of unsuccessful tasks rises for all three units as the number of units rises. The resources required by the tasks to run are tracked using “required capacity” and “target capacity” mentions the amount of resources that the VM contains. The Max fit algorithm is nothing but allocating the largest resource to the smallest task, which means that the required capacity must be less than target capacity and the largest resources should be chosen and that is tracked using “selectedVmCapacity”. It is important to remember that, while the 1-tier architecture has the highest task error percentage, the 2-tier architecture with EO has the lowest task error percent. The consequences of these findings for edge

computing include the possibility that a 2-tier architecture combined with EO may be more efficient in lowering the percentage of failed tasks. This may be since edge orchestration can aid in a better distribution of responsibilities among units, lowering the likelihood of error. On the other hand, a 1-tier architecture may result in more task failures, suggesting that a more distributed architecture may be more efficient in edge computing scenarios. Thus, the table highlights the importance of considering architecture and orchestration when designing an edge computing system to reduce the percentage of failed tasks and optimize performance.

The Table.3 shows the how the number of devices allocated to the edge devices and their processing time. Processing time is one of the major parameters that is used for checking the efficiency of a system or an architecture. The scenario here is that tasks are assigned as per the Max fit algorithm which means that the smallest task gets the largest available resource and then the processing time are monitored.

Table.3. Number of devices vs Average processing time taken in Max fit algorithm.

Average Processing Time in Max Fit			
Average Processing Time (s)/ Number of Devices	**1-Tier**	**2-Tier**	**2-Tier with EO**
50	0.3	0.2	0.1
75	0.5	0.3	0.1
100	1	0.5	0.1
125	1.7	1.2	0.1
150	2.1	1.6	0.1
175	2.5	2	0.1
200	2.8	2.3	0.1

The three entities plotted on the graph are 1-tier, 2-tier, and 2-tier with EO. As the number of edge devices increases, the average processing time of the tasks increases for all three entities because more tasks are required to be processed so the average time taken increases. The graph shows that using a 2-tier architecture with EO can help reduce the average processing time even when using a larger number of devices. This may be because edge orchestration can help better distribute tasks across devices, reducing processing time by balancing workloads. On the other hand, using a 1-tier architecture can increase processing time, suggesting that a more distributed architecture may be more efficient in edge computing scenarios.

The Table.4 plots the average virtual machine (VM) utilization percentage as a function of the number of devices used in an edge computing scenario. Average VM utilization is used here for the graph because it depicts how efficiently the resources are being used. This gives an insight on how to optimize the usage of the resources. The VM utilization rate must be improved in order to obtain an optimal utilization of the available resources. Here the resources required by the tasks to run are tracked using "required capacity" and "target capacity" mentions the amount of resources that the VM contains.

The three entities plotted on the graph are 1-tier, 2-tier, and 2-tier with EO. As the number of devices used increases, the average VM utilization percentage increases for all three entities. These results have important implications for the design of edge computing. The diagram shows that using a 2-tier architecture with EO can help reduce average VM utilization even when using a larger number of devices. This may be because edge orchestration can help better distribute VMs across devices, reducing VM usage by balancing workloads. On the other hand, using a 1-tier architecture can increase VM utilization, suggesting that a more distributed architecture can be more efficient in edge computing scenarios.

Table.4. Number of devices vs Average VM utilization in Max fit algorithm

Average VM Utilization in Max Fit			
Average VM Utilization(%)/ Number of Devices	**1-Tier**	**2-Tier**	**2-Tier with EO**
50	2.4	2.2	1.5
75	2.8	2.1	1.6
100	5.1	3.9	2.1
125	10.5	8.4	2.9
150	15.7	11.8	4.1
175	19	15	5.5
200	24.5	19	6.5

Table.5. Number of devices vs Percentage of failed task in Random allocation Algorithm

Percentage of Failed task in Random Allocation			
Percentage of Failed Tasks/ Number of Devices	**1-Tier**	**2-Tier**	**2-Tier with EO**
50	0.779065	0.763075	0.759986
75	1.289363	0.899609	0.656743
100	1.990336	1.717815	0.613497
125	4.28616	1.559764	0.501609
150	3.769856	1.210722	0.739169
175	4.528569	1.872832	0.502963
200	10.099068	4.986032	1.042699

The graph plots the percentage of failed tasks as a function of the number of devices used in an edge computing scenario. The resources required by the tasks to run is tracked using "required capacity" and "target capacity" mentions the amount of resources that the VM contains. Random allocation algorithm is allocating resource random to a task, which means that the required capacity must be less than target capacity and the largest resources should be chosen and then resources are randomly allocated. As the number of devices used and the Percentage of failed tasks is directly proportional for all three entities. The Results of these findings for the field of edge computing include the possibility that a 2-tier architecture combined with EO may be more efficient in lowering the percentage of failed tasks. This may be since office orchestration can aid in a better distribution of responsibilities among units, lowering the likelihood of error. On the other hand, a 1-tier architecture may result in more task failures, suggesting that a more distributed architecture may be more efficient in edge computing scenarios. Thus, the table highlights the importance of considering architecture and

orchestration when designing an edge computing system to reduce the percentage of failed tasks and optimize performance.

Fig.6. Number of devices vs Average Processing Time in Random allocation Algorithm

Average VM Utilization in Random Allocation							
Average VM Utilization/ No. of Devices	**50**	**75**	**100**	**125**	**150**	**175**	**200**
1-Tier	3.99	6.99	6.90	11.21	11.58	14.69	18.71
2- Tier	3.31	5.16	6.70	8.56	9.03	11.72	14.86
2- Tier with EO	3.57	3.81	6.25	5.91	8.91	8.361	11.93

The Table.6 plots the processing time as a function of the number of devices used in an edge computing scenario. The resources required by the tasks to run is tracked using "required capacity" and "target capacity" mentions the amount of resources that the VM contains. Random allocation algorithm is allocating resource random to a task, which means that the required capacity must be less than target capacity and the largest resources should be chosen and then resources are randomly allocated. The three entities plotted on the graph are 1-tier, 2-tier, and 2-tier with EO. When the number of devices increases randomly, the average processing time increases gradually for all three entities. The graph shows that using 2-tier architecture with EO can help reduce the average processing time even when using a larger number of devices. This may be because edge orchestration can help better distribute tasks across devices, reducing processing time by balancing workloads. On the other hand, using a 1-tier architecture can increase processing time, suggesting that a more distributed architecture may be more efficient in edge computing scenarios. These highlights the importance of considering architecture and orchestration in edge computing projects to optimize performance and processing time.

Table.7 Number of devices vs Average VM Utilization Time in Random allocation Algorithm.

Average Processing Time in Random Allocation							
Average Processing Time/ No. of Devices	**50**	**75**	**100**	**125**	**150**	**175**	**200**
1-Tier	1.80	1.94	1.91	2.11	2.05	2.07	2.34
2- Tier	1.53	1.70	1.61	1.61	1.53	1.64	1.76
2- Tier with EO	1.51	1.38	1.48	1.37	1.47	1.39	1.49

The Table.7 plots the average virtual machine (VM) utilization percentage as a function of the number of devices used in an edge computing scenario. Average VM utilization is used here for the graph because it depicts how efficiently the resources are being used. This gives an insight on how to optimize the usage of the resources. The VM utilization rate must be improved in order to obtain an optimal utilization of the available resources. As the number of devices used increases, the average VM utilization percentage increases for all three entities.

These results have important implications for the design of edge computing. These results shows that 2-tier architecture combined with EO is more effective in lowering the average VM Utilization rate. This may be because branch orchestration can help better distribute VMs across devices, reducing VM usage by balancing workloads. On the other hand, using 1-tier architecture can increase VM utilization, suggesting that a more distributed architecture can be more efficient in edge computing scenarios. These findings highlight the importance of considering architecture and orchestration in edge computing projects to optimize resource utilization.

6. CONCLUSION AND FUTURE ENHANCEMENT

Scheduling is very important in optimizing utilization of resources in edge computing for efficient and effective use of resources. It determines the allocation of computational tasks among computing resources, which plays as factor in system performance and energy consumption. The quality of service, in edge devices can be employed by optimizing task allocation, reduce latency, and optimize system response time, therefore, optimizing task scheduling should be prioritized.

Significant potential lies in optimizing task scheduling for edge computing in various applications, including healthcare, industrial automation, and autonomous vehicles. An example where task scheduling is significant in healthcare is real-time medical monitoring systems' data processing tasks. In autonomous vehicles, improving task scheduling can augment decision-making algorithms, which could yield improved vehicle safety. Also, optimizing task scheduling in industrial automation can enhance production process efficiency, resulting in increased productivity and lowered energy consumption.

Thus, optimizing task scheduling in edge computing gives a beneficial approach for increasing the performance and efficiency of an edge computing system. Edge computing can obtain an increase in performance in a wide range of industries by continuously modifying and adapting task scheduling methods.

6.1 FUTURE ENHANCEMENTS

There is a huge potential for improving scheduling methods and optimization in edge devices in future. The integration of edge computing with 5G networks is one potential topic of future research since it might enable extremely low latency and high-bandwidth connectivity and open up new applications that need real-time processing. The development of edge-native algorithms, specifically created for local processing, distributed computing, and to increase the usage of mobile devices. It could be another area of emphasis. Furthermore, there is a chance to research the application of blockchain technology for safe and open task scheduling in edge computing. Efficiency, dependability, and performance of edge computing systems might be significantly increased with continuous investment in task scheduling optimization, enabling new applications and unlocking new opportunities for both industries and clients.

REFERENCES

[1] Ashish Gupta and Ritu Garg, "Load Balancing based Task Scheduling with HOT in Cloud Computing", *Proceedings of International Conference on Computer and Application*, pp. 1-5, 2017.

[2] Zheng Shi and Zhiguo Shi, "Multi-Node Task Scheduling Algorithm for Edge Computing based on Multi-Objective Optimization", *Proceedings of International Symposium on Electronic Information Technology and Communication Engineering*, pp. 19-21, 2017.

[3] G. Narendra Babu Reddy, "Modified Heuristic Optimization Algorithm for Task Scheduling in Cloud Computing Systems", *Proceedings of International Conference on Computer Communications*, pp. 357-365, 2018.

[4] Lintan Sun and Zigan Li, "Edge Computing Task Scheduling Strategy based on Load Balancing", *Proceedings of International Conference on Computer Science Communication and Network Security*, pp. 309-316, 2019.

[5] Parvesh Humane, "Simulation of Cloud Infrastructure using CloudSim Simulator: A Practical Approach for Researchers", *Proceedings of International Conference on Information Technology and Mechatronics Engineering*, pp. 1615-1619, 2020.

[6] M. Peng, Zhang and Jal, "The Resource Allocation for Edge Computing in Cellular Networks", *Proceedings of International Conference on Computer and Applications*, pp. 29-41, 2019.

[7] Abdul Razzaque, "Task Scheduling in Cloud Computing", Proceedings of International Conference on Long Island Systems, Applications and Technology, pp. 1-7, 2016.

[8] Atul Vikas Lakra, "Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization", *Proceedings of International Conference on Intelligent Computing, Communication and Convergence*, pp. 1-6, 2019.

[9] Medhat A. Tawfeek, "Cloud Task Scheduling based on Heuristic Optimization", *Proceedings of International Conference on Computer Engineering Systems*, pp. 1-5, 2013.

[10] Y. Hu, "An Efficient Improved Heuristic Optimization Algorithm for Dynamic Software Rejuvenation in Web Services", *Proceedings of International Conference on Fog Computing*, pp. 1-6, 2021.

[11] M. Satyanarayana and P. Bahl, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading", *Proceedings of International Conference on Systems and Informatics*, pp. 2431-2435, 2017.

[12] K. Rajakumari, "Fuzzy based Heuristic Optimization Scheduling in Cloud Computing," *Proceedings of International Conference on Computer Systems Science and engineering*, Vol. 40, No. 2, pp. 581-592, 2022.

[13] X. Gao and J. Wu, "Dynamic Load Balancing Strategy for Cloud Computing with Heuristic Optimization", *Future Internet*, Vol. 7, No. 4, pp. 465-483, 2015.

[14] Y. Zheng Shi, M. Chen, Alam and J. Guo, "Multi-Task Scheduling based on Classification in Mobile Edge Computing", *Electronics*, Vol. 8, No. 9, pp. 938-952, 2019.

[15] Y. Mao, You and Cal, "The Integration of Edge Computing with Mobile Cellular Networks", *Proceedings of International Conference on Computer and Applications*, pp. 1-5, 2017.

[16] X. Chen, L. Wu and M. Zhang, "An Improved Heuristic Optimization Technique Algorithm for Task Scheduling in Fog Computing", *Proceedings of International Conference on Control and Automation*, pp. 1584-1589, 2018.

[17] S. Li, H. Wang and Y. Zhang, "A Novel Heuristic Optimization Technique Approach for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Cloud Computing and Intelligence Systems*, pp.437-441, 2017.

[18] M. Shu, L. Wang and X. Chen, "A Heuristic Optimization Technique Algorithm for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Smart Internet of Things*, pp. 58-62, 2018.

[19] G. Garg and S.K. Garg, "Efficient Task Scheduling using Heuristic Optimization Technique in Cloud Computing", *Proceedings of International Conference on Computational Intelligence and Computing Research*, pp. 1-5, 2019.

[20] J. Singh and V. Chauhan, "Enhanced Heuristic Optimization Technique Algorithm for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Electrical Electronics and Computer Engineering*, pp. 1-6, 2020.

[21] M.B. Al-Kassab, H. Moharram and A. Fathy, "Enhanced Ant-Colony-Optimization Algorithm for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Computer Engineering*, pp. 19-24, 2017.

[22] K. Gupta and M. Singh, "Hybrid Ant-Colony-Optimization for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Trends in Electronics and Informatics*", pp. 416-420, 2018.

[23] S. Bansal and A. Chhabra, "Multi-Objective Ant-Colony-Optimization Algorithm for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Computing Communication and Security*, pp. 1-6, 2019.

[24] P.S. Rathore, S.K. Singh and D.P. Vidyarthi, "A New Ant-Colony-Optimization Algorithm for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Communication Systems, Networks and Digital Signal Processing*, pp. 1-5, 2020.

[25] S.R. Singh, S.K. Chaturvedi and K. Chandra, "Ant-Colony-Optimization with Greedy Load Balancing for Task Scheduling in Cloud Computing", *Proceedings of International Conference on Electronics, Information and Communication*, pp. 1-4, 2021.

[26] K.R. Al-Salihi and F.A. Hadi, "An Improved Ant-Colony-Optimization Algorithm for Task Scheduling in Fog Computing", *Proceedings of International Multi-Conference on Systems, Signals and Devices*, pp. 656-660, 2018.

[27] Kumar and P. Rani, "Hybrid Ant-Colony-Optimization for Task Scheduling in Fog Computing", *Proceedings of International Conference on Cloud Computing and Big Data Analysis*, pp. 380-385, 2018.

[28] Y. Li, G. Zhu and J. Liu, "Task Scheduling Algorithm based on Improved Ant-Colony-Optimization in Fog Computing", *Proceedings of International Conference on Machinery Materials and Information Technology Applications*, pp. 1-5, 2019.

[29] H. Zhang, X. Wang and J. Wang, "An Ant-Colony-Optimization-based Task Scheduling Algorithm for Fog Computing", *Proceedings of International Conference on Cloud Computing and Big Data Analysis*, pp.194-198, 2020.

[30] Y. Luo and Q. Liu, "Ant-Colony-Optimization Algorithm for Task Scheduling in Fog Computing Environment", *Proceedings of International Conference on Cloud Computing and Big Data Analytics*, pp. 380-384, 2020.

[31] X. Li, Y. Zhao and Y. Guo, "Task Scheduling Optimization based on Heuristic Algorithm in Fog Computing", *Proceedings of International Conference on Advanced Information Management, Communicates, Electronic and Automation Control*, pp. 1799-1803, 2018.

[32] H. Li, Q. Wang and H. Zhu, "Task Scheduling Optimization in Fog Computing based on Improved Heuristic Algorithm", *Proceedings of International Conference on Big Data and Internet of Things*, pp. 146-150, 2019.

[33] Y. Li, Y. Wu and Y. Qian, "An Efficient Task Scheduling Algorithm for Fog Computing based on Ant-Colony-Optimization", *Proceedings of International Conference on Communication and Information Systems*, pp. 1- 5, 2019.

[34] Z. Wang, Q. Zhang and Y. Chen, "A Task Scheduling Algorithm for Fog Computing based on Improved Ant-Colony-Optimization", *Proceedings of International Conference on Wireless Communications and Signal Processing*, pp. 1-5, 2020.

[35] L. Zhang, J. Li and X. Xu, "Task Scheduling Algorithm based on Improved Ant-Colony-Optimization in Fog Computing", *Proceedings of International Conference on Computing and Artificial Intelligence*, pp. 1-5, 2021.

[36] S. Sahu, S. Dehuri and R. Mall, "A Novel Ant-Colony-Optimization for Task Scheduling in Fog Computing Environment", *Proceedings of International Conference on Computing Methodologies and Communication*, pp. 200-204, 2018.

[37] Y. Wu, J. Chen and Y. Qian, "Task Scheduling Algorithm based on Improved Ant-Colony-Optimization in Fog Computing", *Proceedings of International Conference on Computer and Communications*, pp. 774-779, 2018.

[38] S. Yang, W. Tian and J. Li, "A Multi-Objective Ant-Colony-Optimization Algorithm for Task Scheduling in Fog Computing", *Proceedings of International Conference on Systems and Informatics*, pp. 1509-1514, 2019.

[39] L. Jiang, Q. Liu and Y. Luo, "Task Scheduling Algorithm based on Ant-Colony-Optimization in Fog Computing Environment", *Proceedings of International Conference on Computer Science Communication and Information Technology*, pp. 201-205, 2020.

[40] H. Zhang, Z. Tang and L. Luo, "An Improved Ant-colony Optimization Algorithm for Task Scheduling in Fog Computing", *Proceedings of International Conference on Computer Science and Information Technology*, pp. 103-108, 2021.