POLITECNICO DI TORINO

Master of Science in
COMMUNICATIONS AND COMPUTER NETWORKS ENGINEERING

Master's Degree Thesis

Cloud Resource Allocation: Developing an efficient resource allocation model to optimize the performance of cloud computing systems, while minimizing costs and energy consumption

Supervisors
prof. MATEKOVITS Ladislau

Candidate
Mohsen KHEZLI

October_2023

Table of Contents:

Table of tables:

Table of figures:

1. Introduction

In recent years, cloud computing has gained popularity among users as it delivers customized and reliable computing environments through distributed, parallel, and grid computing. This paradigm enables companies to rent computing resources on demand without worrying about location and infrastructure requirements. The users' data is stored in data centers managed by the provider, who offers computing, storage, network, software, and hardware services.

Cloud computing offers various benefits, including lower resource reprovisioning, cost savings, improved security and reliability, and satisfactory performance. The most significant advantage of cloud computing is its cost-saving feature (Chien, Lai, and Chao 2019; Jula et al. 2021).

1.1. General Description of Problem

Efficient allocation of resources is a critical component of cloud computing. This involves assigning available resources to user requests or applications currently in use. To achieve this, cloud service providers utilize a variety of Dynamic Resource Allocation (DRA) strategies, which are depicted in Figure 1. The Resource Allocation Strategy (RAS) problem is significant because it impacts energy usage, service provider profits, and user costs. Addressing RAS issues can help to reduce resource usage, balance loads, and integrate resources effectively. In order to develop an optimal RAS, providers must avoid issues such as shortages, resource contention, fragmentation, overprovisioning, and under-provisioning. To accomplish this, a dynamic RAS requires knowledge of the type and number of resources required by each user or application.

1.2. Specific Problem consideration

Optimal RAS could solve the problem of starvation by provisioning resources to individual users by managing resources. It is crucial to have an open dialogue regarding the obstacles that both the service provider and the user encounter. It is not uncommon for overprovisioning of resources to transpire when users overestimate their needs. Conversely, under-provisioning may arise from the provider's allocation decisions.
It is important for the Resource Allocation Strategy of a system to meet certain parameters such as response time, latency, throughput, fairness, and economic service. In order to achieve optimal RAS, it is crucial to have knowledge about the application requirements and user needs.
The management of Dynamic RAS poses a significant challenge for cloud computing services as it impacts energy consumption, user costs, and service provider profits. Service providers strive to maximize their return on investment while users look for cloud services that offer quality service at a reasonable price. Therefore, an optical Dynamic RAS capability can prove to be highly effective for both cloud service providers and users, offering mutual benefits.

Dynamic RAS is a technique that evaluates the present state of each resource within the cloud ecosystem and offers an algorithm for the most efficient allocation of physical resources, resulting in reduced operational costs.

Given the scale and complexity of the cloud environment, service providers need a solution that can handle real-time processing and provide an efficient DRAS (Dynamic Resource Allocation System).

1.3. Resource Allocation Strategies

Resource allocation strategies (RAS) for cloud computing vary depending on the specific service, infrastructure, and applications that require resources. Figure 1 illustrates the different RAS proposed for the cloud paradigm. The next section explores the RAS used in cloud computing. Our paper aims to investigate some aspects of the Run-Time strategy.

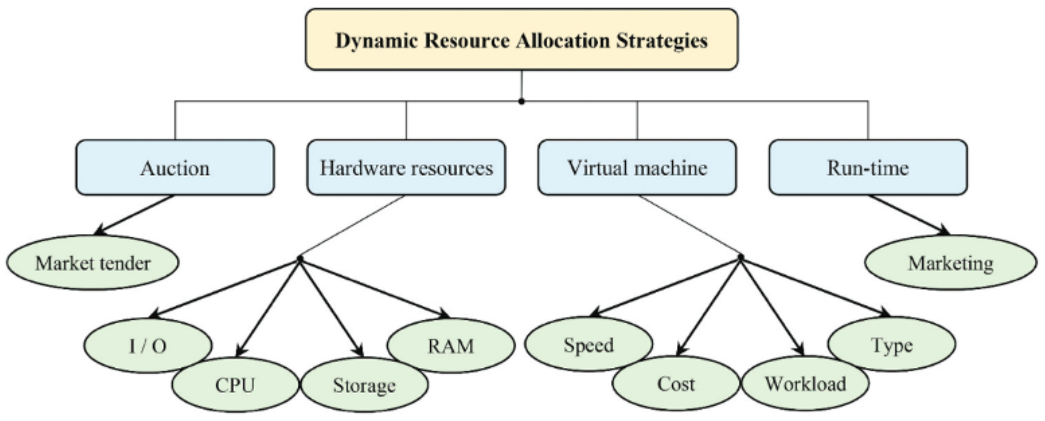


Figure.1, Dynamic Resource Allocation Strategies, [1]

1.4. State the problem

In a cloud environment, each physical server hosts multiple virtual machines with different resources and configurations. These virtual machines can serve as computers or servers for applications, or function as virtual monitoring devices. The goal of developing, customizing, and studying algorithms is to allocate appropriate virtual machines to users and improve resource utilization. This problem is known as the Dynamic Resource Allocation Strategy (DRAS).

In any cloud environment, the scheduling system plays a critical role in distributing tasks between physical servers and virtual machines (VMs). The system's main objective is to ensure that tasks are properly allocated among VMs to achieve maximum resource efficiency and minimize task duration (also known as makespan). Essentially, the scheduling system needs to address the DRAS problem, which is a multi-objective problem.

One of the biggest challenges for scheduling algorithms is migration. When a virtual machine (VM) is overloaded with too many tasks, the system needs to migrate some tasks to less-loaded VMs. This is important for load balancing, but it also comes with some overhead costs for the service provider. Cloud computing ensures fairness by giving equal access to resources for all users and applications, regardless of size or importance. This prevents any one user or application from monopolizing resources and maintains a level playing field. Fairness promotes efficient resource utilization in the cloud ecosystem.

1.5. Structure of Thesis

Our research is centered around the firefly algorithm, which takes cues from natural processes to tackle the DRAS problem with a highly effective solution. In Chapter 2.1, we offer notable contributions: firstly, we employ FA to address the DRA problem as a multi-objective optimization challenge; secondly, we utilize a pay-as-you-use model and a fuzzy approach to prioritize requests and improve load balancing; and lastly, we assess our proposed strategy and compare it with other established strategies.

In our work, we focus on developing an innovative strategy for optimal resource allocation in cloud computing services. Our goal is to make these services more economical and efficient by determining the best allocation of processor and bandwidth resources in different scenarios.

In Section 2.2, we evaluate the impact of joint multiple resource allocation and compare three different strategies (I, II, and III) in terms of loss probability and fairness across various scenarios. Our analysis aims to identify the most effective approach for achieving optimal resource allocation.

In Chapter 2.3, we discuss Femtocell Networks which are commonly used in small businesses for managing Resource Allocation. However, they are now also useful for resource allocation in cloud computing. The following section, 2.4, covers Linear Strategy. In the last two chapters, we briefly discuss these two strategies and provide details regarding their fairness, efficiency, advantages, and disadvantages.

2. Literature review

2.1. Firefly

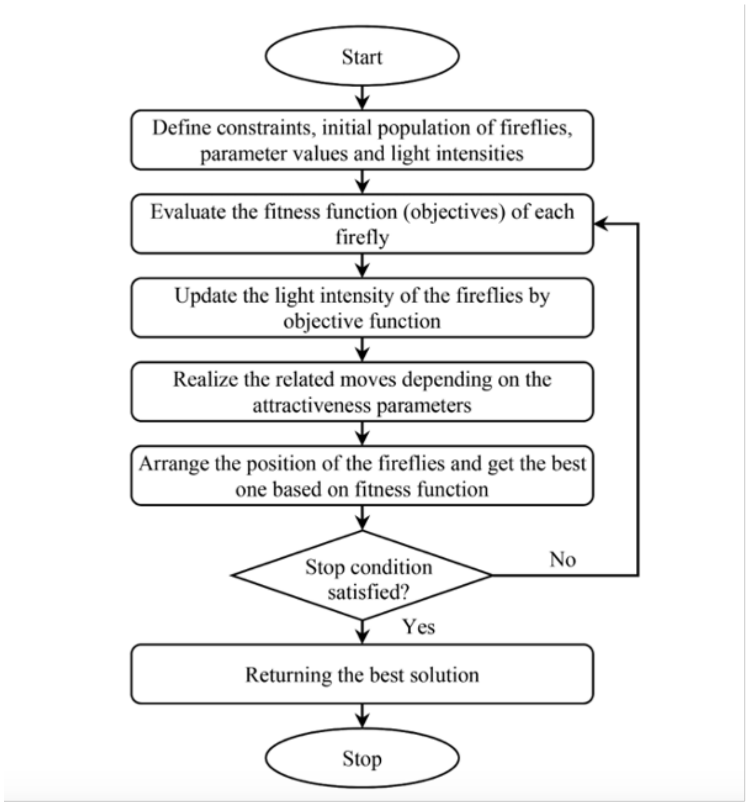
Numerous studies have concentrated on managing energy in cloud environments. This research was aimed at tackling the problem of energy consumption by examining two types of algorithms: energy-saving and energy-efficient. The ultimate goal of these algorithms is to decrease energy usage in data centers. However, certain parameters were overlooked in some of the algorithms, including the service-level agreement. To address the DRAS issue, most techniques utilize heuristic approaches that employ evolutionary algorithms.

2.1.1 Firefly Algorithm

Optimizing the algorithm requires utilizing a mathematical methodology to find a solution. The level of difficulty in optimization is dependent on the mathematical interplay between decision variables, constraints, and objectives. In this particular research, the focus is on multi-objective optimization, which involves optimizing more than one objective simultaneously. To address this challenge, the study employed nature-inspired algorithms and biological processes to develop a robust solution for the DRAS problem.

In 2009, Yang introduced the Firefly Algorithm, which is inspired by the communication among fireflies. It is a heuristic algorithm that relies on the cooperation of low-intelligence agents to create a high-level of swarm intelligence.

It is believed that fireflies are attracted to each other's brightness regardless of their gender. The level of attraction can vary depending on the distance between them. In cases where multiple fireflies have the same brightness, they tend to move randomly and operate independently in parallel. The firefly algorithm has found applications in different fields and various sciences and is often used in conjunction with other optimization algorithms and techniques.


Start
Define constraints, initial population of fireflies, parameter values and light intensities
Evaluate the fitness function (objectives) of each firefly
Update the light intensity of the fireflies by objective function
Realize the related moves depending on the attractiveness parameters
Arrange the position of the fireflies and get the best one based on fitness function
Stop condition satisfied?
No
Yes
Returning the best solution
Stop
Figure 2, Flowchart of Firefly algorithm steps, [1]

The Firefly algorithm is a cutting-edge method employed by researchers to tackle optimization problems in dynamic environments. It stands out as a standalone technique that can be utilized for parallel processing. As shown in Figure 2, the FA steps are outlined in a flowchart. While there are other algorithms that use random search to generate solutions within a search space, FA possesses a unique set of advantages that make it a fitting optimization approach for the DRAS problem. Consequently, in this study, we opted to utilize FA to solve the DRA problem.

2.1.2 Method I

This solution is called IFA-Dynamic RAS and it aims to solve the DRAS problem by combining a fuzzy approach with an optimization technique based on the Firefly algorithm, which is referred to as IFA-DRA. The main objective of this method is to determine the best sequence of tasks on VMs by including different heuristic operations and objectives. This sequence ensures that the appropriate VM resources are allocated to each task.

The fuzzy approach is a computational technique that deals with uncertainty and imprecision in data or information. It is based on fuzzy logic, which allows the representation and manipulation of vague or uncertain concepts. In the IFA-Dynamic RSA solution, the fuzzy approach is used to tackle the issue of dynamic resource allocation by considering uncertain factors and incorporating them into the decision-making process. By utilizing fuzzy logic, the solution can take into account varying levels of resource requirements and adjust the allocation strategy accordingly.

In the Firefly algorithm, each Firefly comes up with a solution to the problem and outlines a sequence of tasks to be executed on the VMs. These sequences include the allocation of the necessary resources of the VMs for each task. To achieve this, it is essential to prioritize each task based on the pay-as-use method. The initial population for the algorithm was generated based on this task priority, and we used a fuzzy method to distribute the tasks to the VMs.

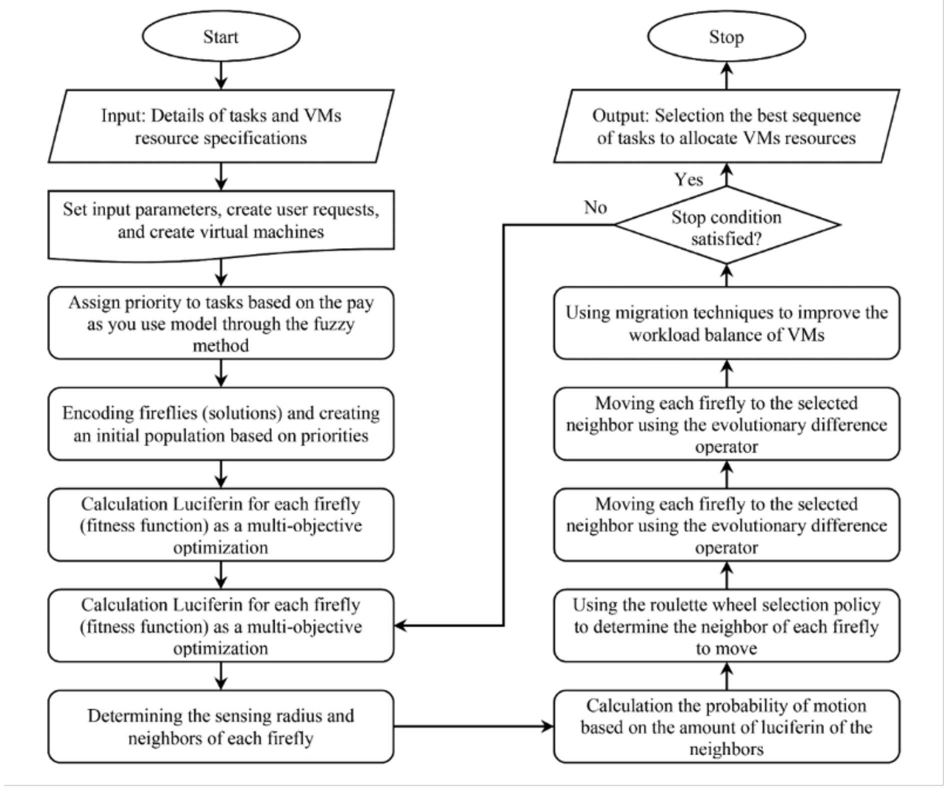

Start
Input: Details of tasks and VMs resource specifications
Set input parameters, create user requests, and create virtual machines
Assign priority to tasks based on the pay as you use model through the fuzzy method
Encoding fireflies (solutions) and creating an initial population based on priorities
Calculation Luciferin for each firefly (fitness function) as a multi-objective optimization
Calculation Luciferin for each firefly (fitness function) as a multi-objective optimization
Determining the sensing radius and neighbors of each firefly
Calculation the probability of motion based on the amount of luciferin of the neighbors
Using the roulette wheel selection policy to determine the neighbor of each firefly to move
Moving each firefly to the selected neighbor using the evolutionary difference operator
Moving each firefly to the selected neighbor using the evolutionary difference operator
Using migration techniques to improve the workload balance of VMs
Stop condition satisfied?
No
Yes
Output: Selection the best sequence of tasks to allocate VMs resources
Stop

Figure 3, IFA-DSA flowchart, [1]

The method is a multi-objective algorithm and includes four objectives: load-balancing, minimizing the task duration time of the last task, minimizing the migration rate, and finally minimizing the average runtime. The luciferin value (attractiveness) was evaluated for each firefly to optimize the objective function.

The radius of each neighborhood sensor is determined based on the luciferin value, and the probability of fireflies moving toward neighbors is calculated according to the luciferin value. To move each firefly to a neighbor, the evolutionary difference operator is used.

When dealing with an imbalanced workload, the migration technique can be utilized between the virtual machines to enhance the workload.

When the utilization of certain VMs exceeds a certain threshold, the performance of the entire system can be improved by utilizing the migration technique to transfer the workload from overloaded VMs to underutilized ones. To better understand this process, please refer to Figure 3 which illustrates a flowchart of IFA-DSA.

2.1.3 Prioritize Task

We've classified requests into Low, Medium, and High-priority levels using a fuzzy logic algorithm. This algorithm takes into account two crucial factors: the user's payment and the computational size of the tasks.

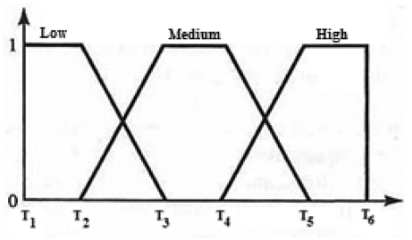
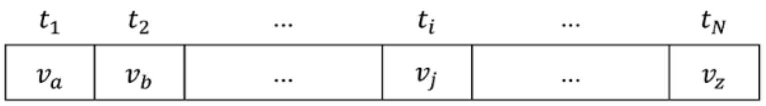


Figure 4, Trapezoidal membership function with three modes to prioritize tasks, [1]

Our algorithm utilizes a sophisticated system to prioritize tasks based on two key parameters: the priority decision parameter (ω) and the trapezoidal membership function (T1, T2, ..., T6). By taking into account the payment and size of each task, the decision parameter determines the optimal ratio between the two. Meanwhile, the trapezoidal membership function assigns a priority level to each task based on these factors, with higher payment and lower size leading to a higher priority. This cutting-edge approach, which employs a fuzzy logic system, results in greater flexibility and efficiency when allocating tasks while improving the overall user experience.

2.1.4 Encoding and Initialize Population in Fireflies

In the DRAS, each firefly can provide a solution within the search space, and each firefly represents a vector of length N, where N indicates the total number of tasks. In the vector, a VM is assigned to each task. Structure of the Firefly in the DRAS problem, which represents VM allocation to the task.



t_1	t_2	...	t_i	...	t_N
v_a	v_b	...	v_j	...	v_z

Figure 5, Encoding structure of fireflies, [1]

The FA algorithm generates an initial solution population randomly, which may result in an inadequate distribution of tasks among Virtual Machines (VMs). To address this issue, a heuristic-based approach is used to create an initial population based on the workload's task priorities. This ensures that tasks are assigned to the VM based on the workload.
To guarantee performance, the heuristic method assigns tasks to VMs in the following way: for low-priority tasks, select the VM with the least workload; for medium-priority tasks, assign the VM with the lowest number of high- and medium-priority tasks; and finally, for high-priority tasks, assign the VM with the lowest number of high-priority tasks.

2.1.5 Luciferin

Fireflies use Luciferin to emit their radiance and attract other Fireflies. The amount of Luciferin a Firefly has indicated its position in the search space. In the DRAS algorithm, Luciferin is a multi-objective function that includes factors such as Task Duration, Workload Balancing, Average runtime of tasks, and Migration Task. These factors are considered when determining the objective function of the DRAS.

2.1.6 Migration

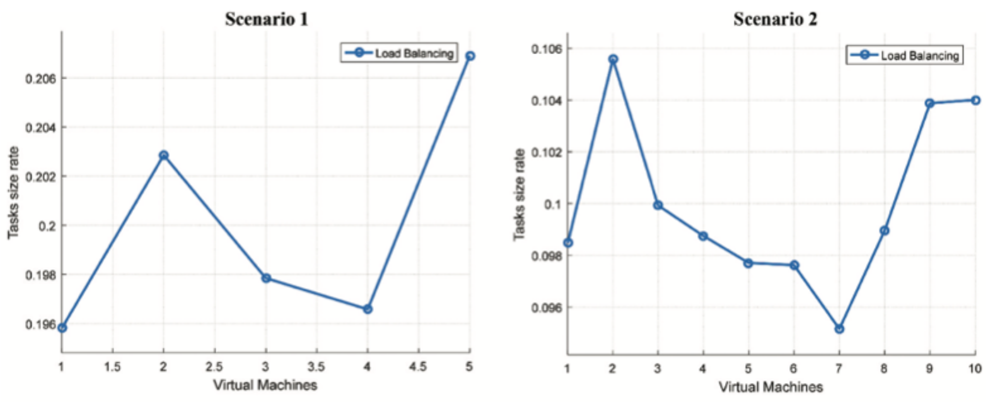
In system management, an increase in virtual machine (VM) load is commonly known as an "Overload VM." To ensure optimal workload distribution, a migration technique is applied. This involves selecting a task at random from the Overload VM and transferring it to the "Less Overload VM" (LVM) once the difference between the two surpasses a specific threshold.

Table 1, Scenario defined for the DSA problem, [1]

Scenario	Number of tasks(N)	Number of VMs
Scenario 1	100	5
Scenario 2	50	10
Scenario 3	500	60
Scenario 4	1000	100

2.1.7 Outcome

As you mentioned earlier in the paper, the initial population is created using a heuristic method based on a fuzzy approach, which ultimately sets the priority of the tasks. Additionally, the workload is distributed among the virtual machines (VMs) at the start of the scheduling process. Figure 6 illustrates the workload on each VM, which is evaluated based on the task size assigned to that VM in relation to the total workload.



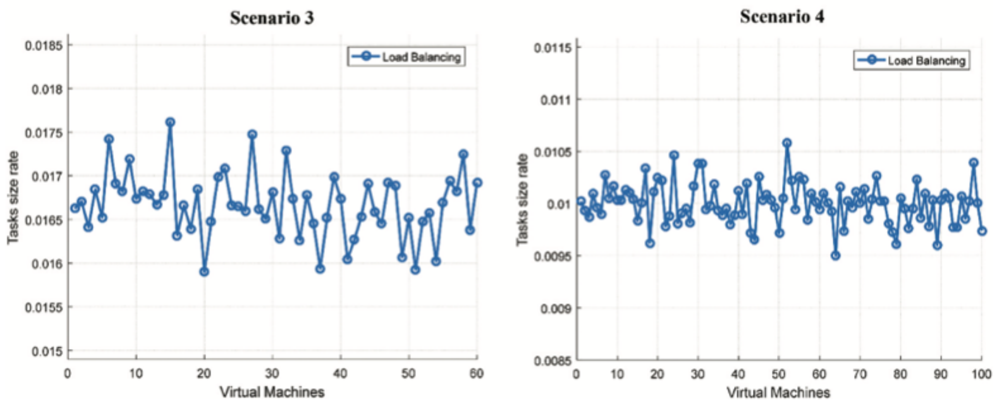
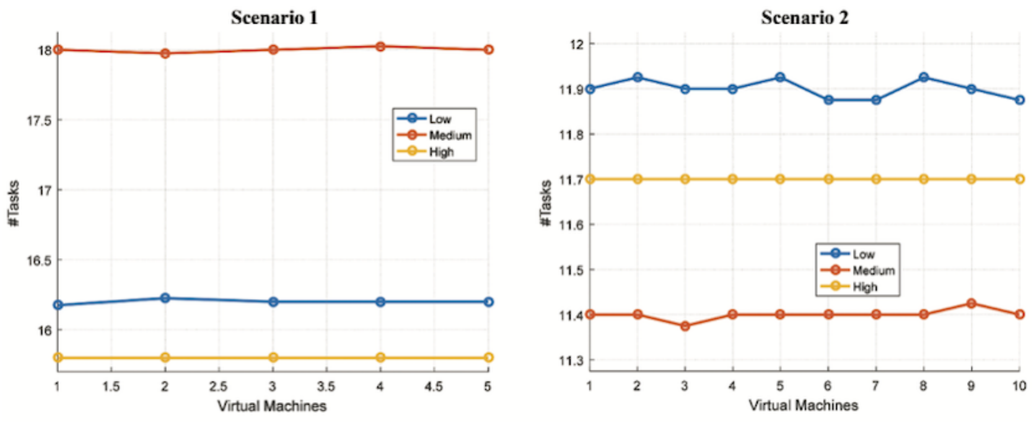


Figure 6, VMs load Balancing at the beginning of the scheduling process, [1]

According to the simulation, all VMs carry the same workload, which proves that the heuristic approach brings about the necessary load balancing at the start of the scheduling process. Figure 7 displays the simulation results for various scenarios, indicating that it is equitable to distribute tasks with different priorities among virtual machines. For instance, in scenario 3, the average load does not exceed 12.9, while it is around 12.6, indicating that the upper hand and lower hand show almost similar behavior, with the average load being almost the same.

This simulation suggests that the workload is evenly distributed across all VMs. As a result, the heuristic approach provides the required load balancing at the start of the scheduling process. The distribution of tasks on VMs is based on their respective priorities. For this purpose, a similar number of tasks should be assigned to each VM with different priorities. To ensure that the number of tasks with varying priorities on VMs is evenly distributed, the IFA-DRAS assigns sequence tasks to VMs while creating the initial population.



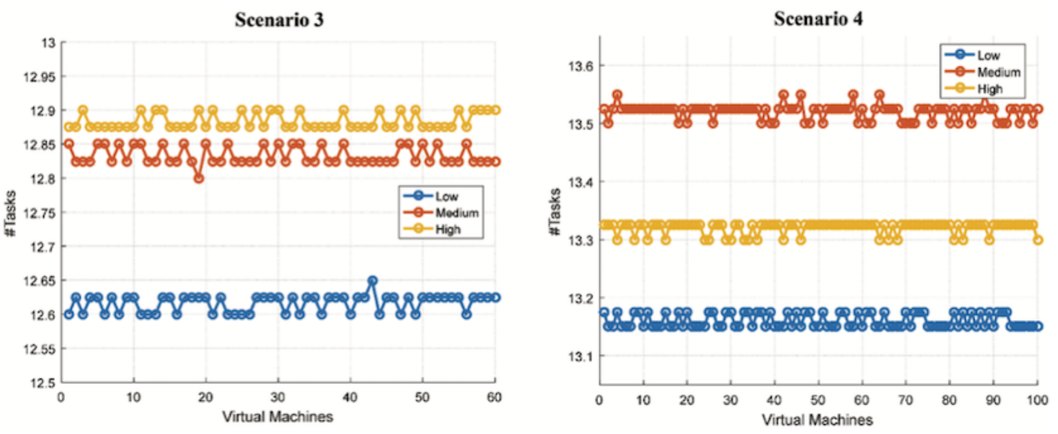


Figure 7, Distribution of Tasks with Different Priorities on VMs, [1]

2.1.8 Performance

This study compares the performance of three optimization algorithms – FA, GA, and PSO – using the makespan metric to evaluate scenarios with varying numbers of tasks (100, 200, 300, 400, and 500), as shown in Figure 8. According to the results, FA performed better than both GA and PSO due to its heuristic approach for generating the initial population and ensuring proper task distribution. Consequently, FA achieved better results in comparison to the proposed method, which started with a population of the best quality. Additionally, faster convergence during evaluation was facilitated by load balancing on VMs.

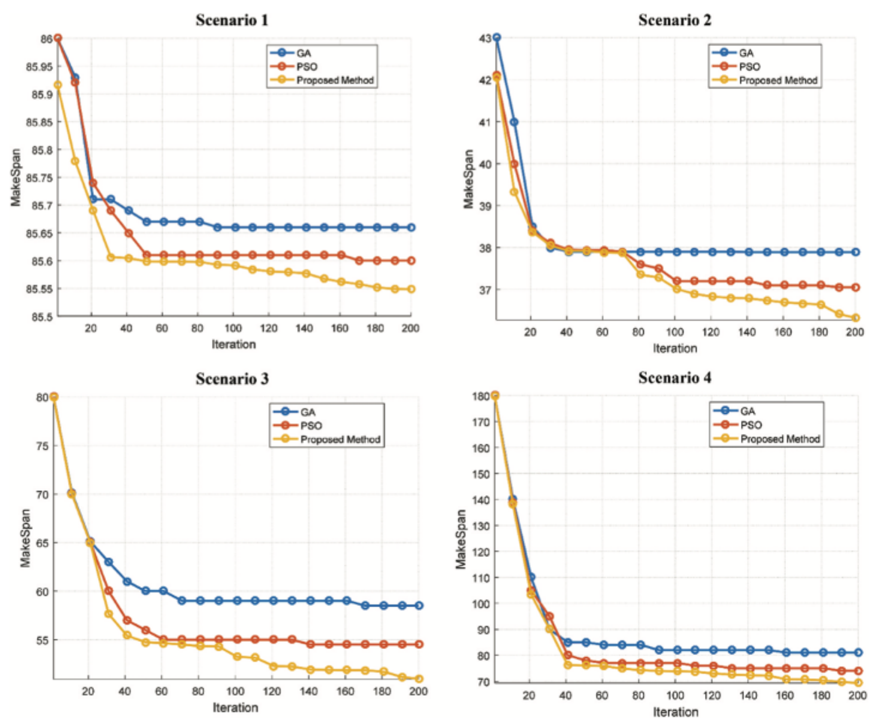


Figure 8, Comparison of FA with GA and PSO in the proposed method, [1] In a comprehensive comparative study, FA-DRAS was tested against First-Come, First-Served (FCFS), Task Scheduling with Dynamic Queue based on Fuzzy Logic and Particle Swarm Optimization (TSDQ-FLPSO), and the Improved Cuckoo Search Algorithm (ICFA). The assessment was based on the makespan of a variety of tasks. The findings revealed that FA-DRAS outperformed the other algorithms in terms of makespan. It is noteworthy to mention that the experiment maintained a constant number of VMs throughout.

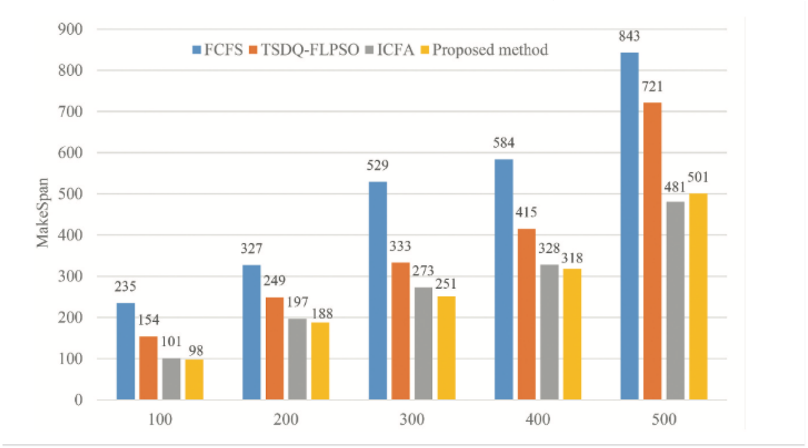


Figure 9, Comparison of IFA-DSA with similar methods in the makespan criteria, [1]

Different parameters related to DRAS in the cloud environment can be considered, such as workload balance, fairness, efficiency, resource utilization, runtime, and energy consumption. The table below compares some of these parameters.

Table 2, Parameters used by different methods for DRSA, [1]

Parameters	**FCFS**	**TSDQ-FLPSO**	**ICFA**	**FA-DRAS**
Workload Balance	yes	yes	yes	Yes
Being Fair	-	yes	-	-
Efficiency	yes	-	-	-
Utilization of Resource	-	-	Yes	-
Run Time	-	-	-	yes
Energy Consumption	-	-	Yes	-
Processing cost	Yes	-	-	Yes
Response Time	Yes	Yes	Yes	-
Prioritize task	-	-	-	Yes

2.2. Resource Allocation

In this resource allocation strategy for cloud environments, the assumption is that there is a common pool of different types of resources that are allocated for a certain amount of time for each request. The focus is on processing ability (C) and bandwidth (N) resources, which have varying levels of demand.

The hardware resources for cloud computing are assumed to be distributed across different data centers, represented by K=1,2, 3, …, K, where each data center has a maximum processing ability and maximum bandwidth denoted by (Cmaxj) and (Nmaxj).

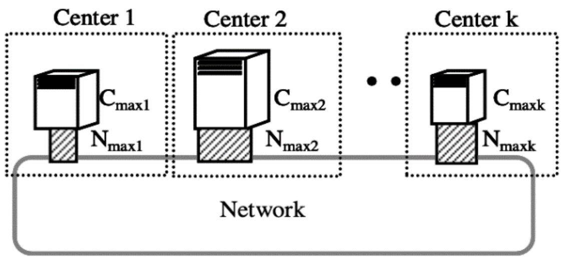


Figure 10, System model for cloud computing services, [2]

The objective of the strategy is to choose the best data center to allocate resources for each request within a specific time frame. If there are insufficient resources in the data centers to fulfill the request, it will be declined.

2.2.1. Optimal joint Resource Allocation

2.2.1.1.Assumption

We assume that the cloud environment is a non-delayed system with static resource allocation. The goal is to effectively allocate processing ability and bandwidth resources to the maximum number of requests.

2.2.1.2. Impact of Joint Multiple Resource Allocation

This passage discusses the limitations of a resource allocation method that only considers one type of resource, such as processing ability, in selecting a data center. In such cases, the method may fail to allocate resources efficiently and may result in a deadlock state if the request requires both processing ability and bandwidth. This method is known as Method I.

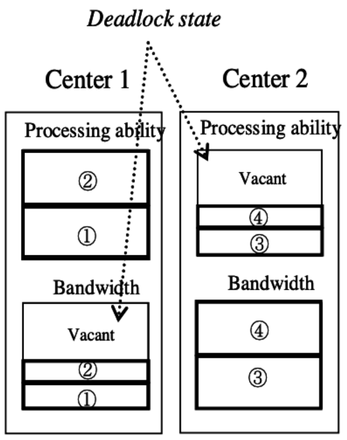


Figure.11.1, Only processing ability is considered in the selection of a center, Joint multiple resource allocation K=2, [2]

To address this issue, the joint multiple resource allocation method considers both processing ability and bandwidth in selecting the data center, thus improving the efficiency of resource allocation and avoiding the deadlock state.

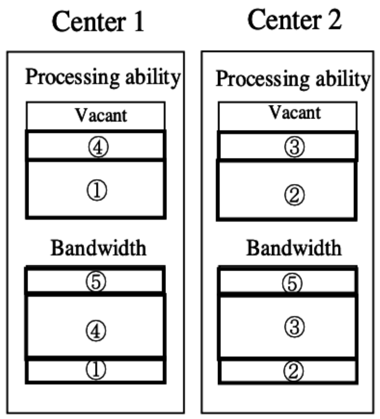


Figure.11.2, Both Processing ability and bandwidth are considered in the selection of centers, Joint multiple resource allocation K=2, [3]

2.2.1.3.Optimal joint multiple resource allocation method

When allocating resources, it is beneficial to refer to multiple resource types. This approach makes it easier to identify a resource that has a significant impact on allocation. This is called Method II. When selecting a center using this method, only the recognized resource type is considered. This method utilizes the best-fit approach to reserve resources for future requests and reduces the possibility of a deadlock situation.

Comparing Method I and Method II, it is clear that the joint allocation of multiple resources is more effective. Method I selects the center in a predefined order, whereas Method II considers both resource types in the selection of a center.

2.2.1.4.Resource Allocation Method II

To select a resource type, we compare the required resource size to the maximum resource size for each type. The type with the largest size proportion is chosen as the "identified resource." We then select a center with the least capacity of the identified resource from a group of k centers. Processing ability and bandwidth are measured differently, one in percentage of CPU power and the other in bits per second (b/s). To compare resource sizes, consider this example: a center has a maximum bandwidth of 100Mb/s. A request for 20% of CPU power and 30Mb/s requires 20% of processing ability and 30% of bandwidth. In this example, bandwidth would be the identified resource because it is more demand than processing ability.

The chosen center must satisfy certain criteria - it should have the minimum available size of the identified resources and have available the two resource types that are larger than or equal to the required resource type. If this condition is not met, the request is rejected.

Once a center is chosen, both resource types are allocated to the request and released after a certain period of time. If resources are not available in the first selected center, the algorithm selects the second possible center chosen by Round-Robin fashion, which is a pre-defined order regardless of which center was selected for the last request. If no center with adequate resources is available, the request is rejected.

Finally, after finding a center with the required resource type, both resource types are allocated simultaneously to the request and are released after a period of time. This makes the method highly effective.

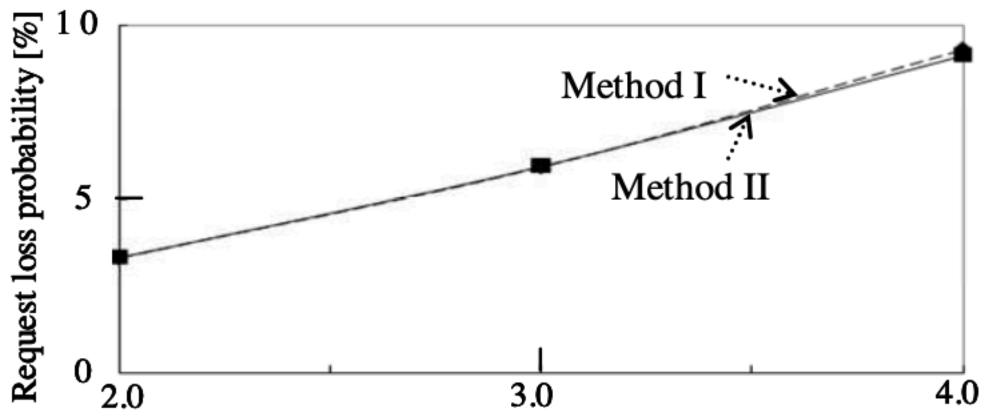
2.2.1.5. Simulation

2.2.1.5.1. Assumption and Condition

The simulation is implemented using the C language, with a total of 2 centers (k=2) available. The maximum processing ability for center 1 and center 2 are Cmax1 and Cmax2, respectively. The corresponding maximum bandwidths are Nmax1 and Nmax2. The Gaussian distribution is used to specify the size of the required processing ability and bandwidth, with C and N representing the distribution for processing ability and bandwidth, respectively.
The requests are generated based on the exponential distribution with the average interval being q. Once a request is created, it will use both resource types (processing ability and bandwidth) from the time of creation until the completion time of the service, which is given by H. The system will repeatedly generate m requests, with each request requiring a specific amount of processing ability and bandwidth. The pattern for r requests is {C=a1, N=b1; C=a2, N=b2; ...; C=ar, N=br}.
The goal of the simulator is to compare method II with method I in terms of rejected requests, resource utilization, and waiting time for requests.

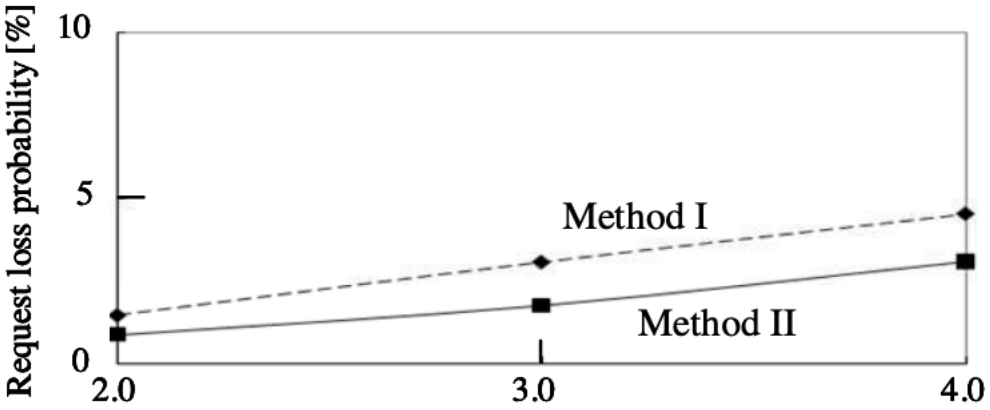
2.2.1.5.2. Result

Figure 12.1 compares the probability of losing a request, due to unavailability of processing ability or bandwidth, when C=N.



Average size of required resource α, Request generation pattern 1 {C=α, N=α}

Figure 12.1, Comparative evolution of Method I and Method II,[2]

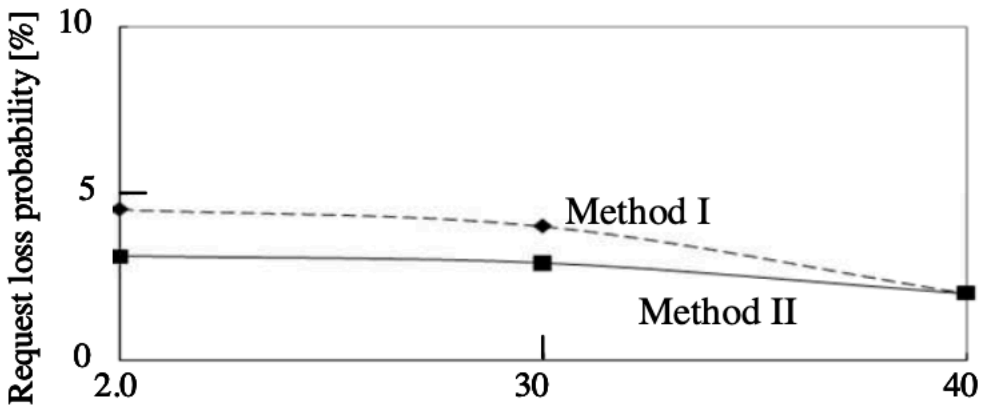


The average size of required resource β, Request generation pattern 2 {C=β, N=1; C=1, N=β} Cmax1=Cmax2=20, Nmax1=Nmax2=20, H=6.

Figure 12.2, Comparative evolution of Method I and Method II, [2]

Figure 12.2 compares the request loss probability when processing ability and bandwidth (C and N) rise and fall in anti-phase.

Figure 13 examines how the maximum resource size ratio of each center affects request loss probability, assuming constant processing ability (Cmax1 +Cmax2) and bandwidth size (Nmax1 +Nmax2).



Cmax1+Cmax2=40, Nmax1+Nmax2=40, H=6 {C=4, N=1; C=1, N=4}

Figure 13, Impact of the size of Cmax and Nmax, [2]

Figure 14 assesses the effect of the number of centers on the request loss probability using the same simulation parameters as those in Figure 12.2.

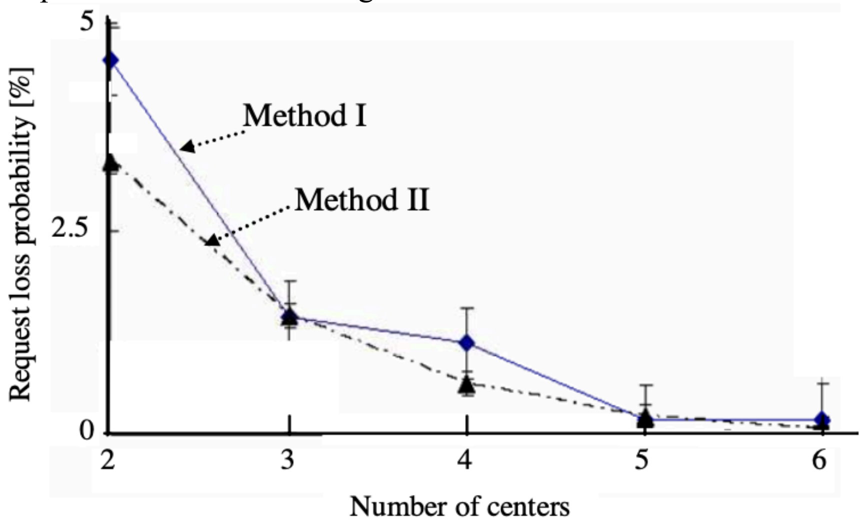


Figure 14, Impact of the number of centers, [2]

The results indicate that method II can reduce the request loss probability and, as a result, reduce the total amount of resources when the size of the processing ability and bandwidth rise and fall. The same applies even when the number of centers increases, with the exception of an odd number of centers. When there are an odd number of centers and processing capability and bandwidth sizes increase and decrease, the basic method may lead to a deadlock state. As more centers are available to process requests, the request loss probability decreases as the number of centers increases.
In addition, the simulator showed that allocating resources to a single specific center is better than distributing resource allocation to multiple centers.

2.2.2. Fairness

Fairness in cloud resource allocation refers to the equitable distribution of resources among multiple users, applications, or tenants. This ensures that all users obtain a fair share of resources without being negatively affected by the resource demands of others. Achieving fairness is important to prevent resource starvation, maintain the quality of service, and ensure overall user satisfaction. We attempt to improve the quality of method I in a situation that may involve the possibility of unfair resource allocation: when resource allocation to a specific user occupies most of the resources, other users may face starvation, which can be solved by using the proposed method. The available resources are divided into blocks of resources, and each block is allocated to each request. This method prevents an application, user, or task form from occupying most of the resources and providing fair resource allocation.

2.2.2.1.Fairness in joint multiple resource allocation

The proposed method has four objectives for obtaining fairness:

1. Although fairness should be achieved without queuing, it may be necessary to delay resource allocation.
2. Multiple types of resources should be considered when tracking fairness, as fairness for one resource type may not be fair for another.
3. As some users may require more bandwidth than others, it may not be fair to balance the amounts of the two resource types.
4. The allocation of resources to each user is unfair if there are no rejections for all users.

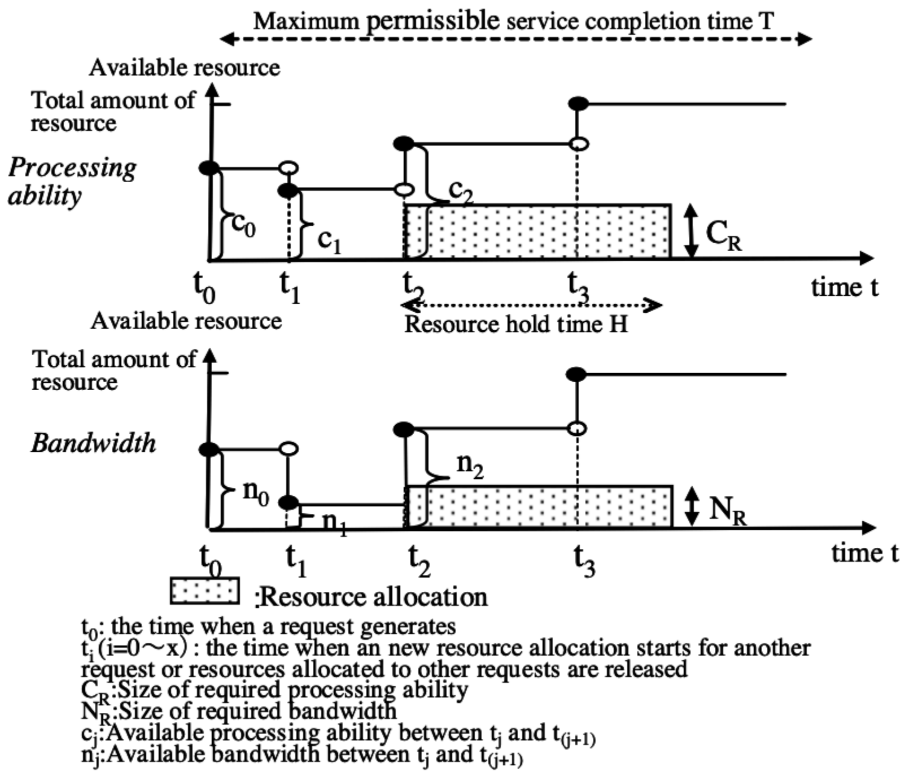
2.2.2.1.1. Proposed Method for Fairness

This study introduces a technique that aims to achieve fairness in resource allocation. The proposed method distributes resources to each request in each time slot based on their individual requirements. To determine fairness, the technique identifies the key resource type, and measures the total amount of all allocated key resource types in each time slot, similar to Method II.

2.2.2.1.2. Extension of fair joint multiple resource allocation method

The paper presents a new approach that extends method II and aims to prevent an imbalance in resource allocation in the allocation of key resources. The proposed method employs delayed resource allocation in the following time slot to address issues related to the previous time slot during the computation time T.

If there is a request and there are insufficient required resources, requests are delayed instead of being rejected, and resources become available in the next time slot. This way, the required resources in the previous time slot are filled. Figure 15 illustrates the resource management diagram that depicts the method used to determine when to start the service time during delayed resource allocation, which is called method III and is utilized in unfair cases following method II.

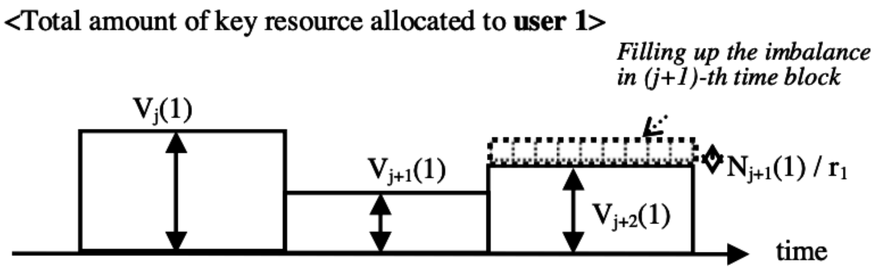

Maximum permissible service completion time T
Available resource
Total amount of resource
Processing ability
c_0
c_1
c_2
C_R
t_0 t_1 t_2 t_3 time t
Resource hold time H
Available resource
Total amount of resource
Bandwidth
n_0
n_1
n_2
N_R
t_0 t_1 t_2 t_3 time t
:Resource allocation
t_0: the time when a request generates
t_i (i=0～x) : the time when an new resource allocation starts for another request or resources allocated to other requests are released
C_R:Size of required processing ability
N_R:Size of required bandwidth
c_j:Available processing ability between t_j and $t_{(j+1)}$
n_j:Available bandwidth between t_j and $t_{(j+1)}$

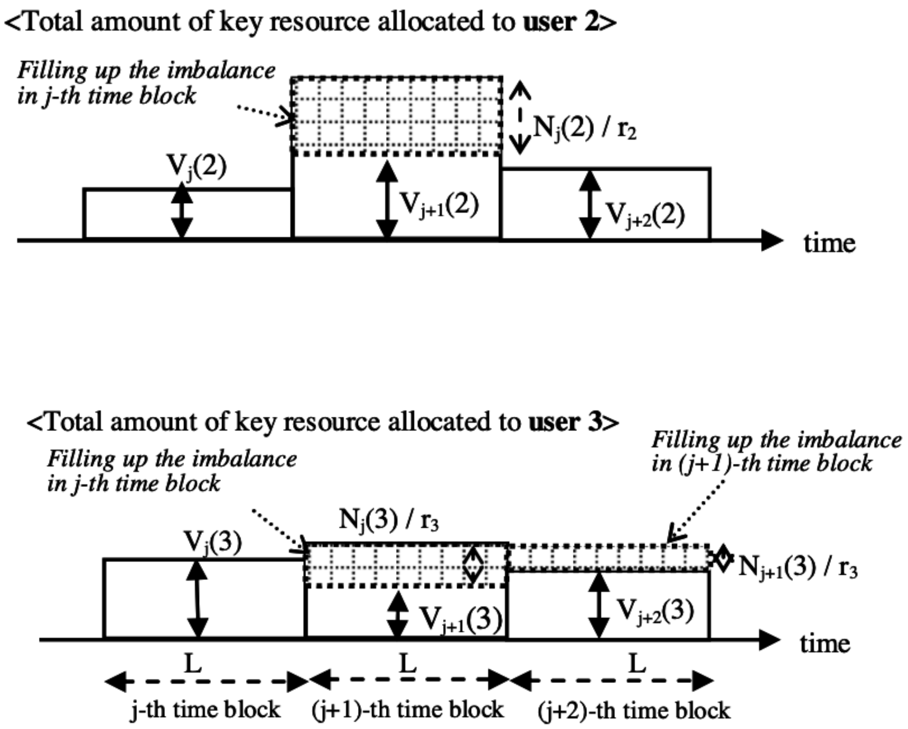
Available resource management diagram

Figure 15, Impact of the number of centers, Reference [18]

An example of filling up the imbalance is presented in Figures 15, 16, and 17 for three users. It shows how imbalanced the required resources are allocated to each user in the next flowing time slot.

The user which has the largest value of Vj(g) is called 'user g1' in the j-th time block, where Vj(g) is given by {Total amount of key resource allocated to user g in j-th time block} *rg.


<Total amount of key resource allocated to user 1>
Filling up the imbalance in (j+1)-th time block
$V_j(1)$
$V_{j+1}(1)$
$V_{j+2}(1)$
$N_{j+1}(1) / r_1$
time



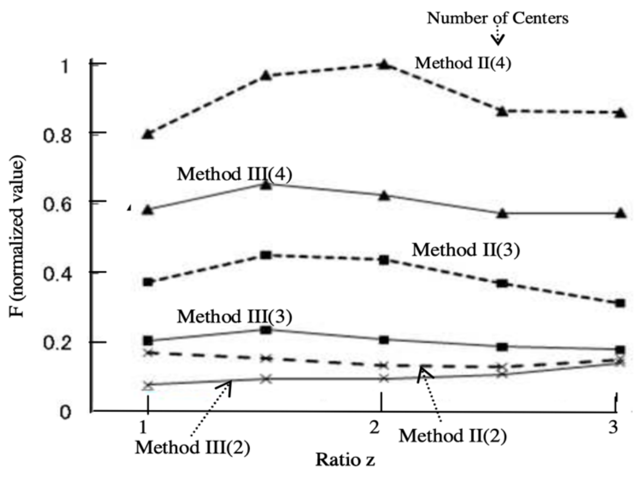
<Total amount of key resource allocated to user 2>
Filling up the imbalance in j-th time block
$N_j(2) / r_2$
$V_j(2)$
$V_{j+1}(2)$
$V_{j+2}(2)$
time
<Total amount of key resource allocated to user 3>
Filling up the imbalance in j-th time block
Filling up the imbalance in (j+1)-th time block
$N_j(3) / r_3$
$V_j(3)$
$N_{j+1}(3) / r_3$
$V_{j+1}(3)$
$V_{j+2}(3)$
time
L
L
L
j-th time block
(j+1)-th time block
(j+2)-th time block

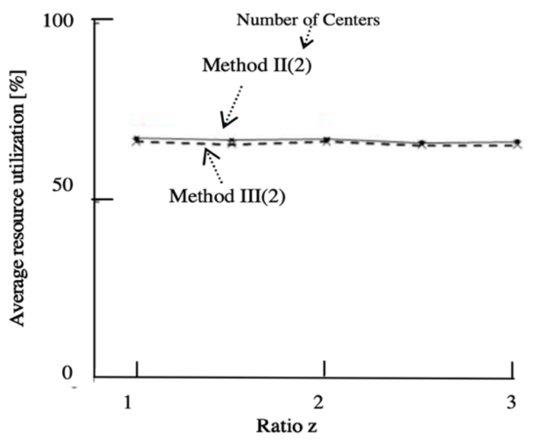
Example of filling up the imbalance in the previous time block

Figure.16,[2]

2.2.3. Result

The pattern represents the request generation for users and is presented by {C= X, N=X} and {C= Z.X, N=Z.X}, Z is the ratio of the size of User 2's request to User 1's request.

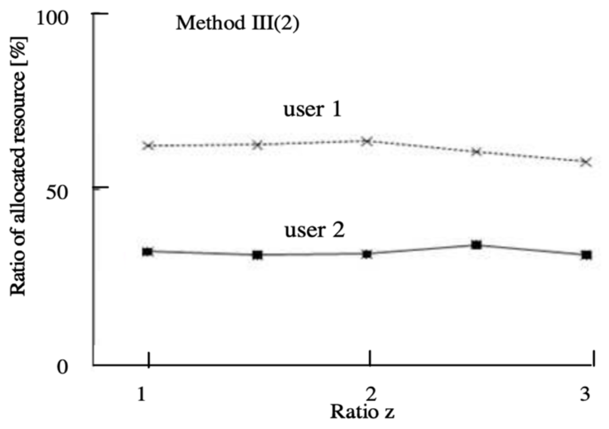




Evaluation of fairness (value F) and resource efficiency
*Cmax1=Cmax2=20, Nmax1=Nmax2=20, H=6; {C=2, N=1} for user 1, {C=2*z, N=1*z} for user 2, r1=r2=1*

Figure.17, [2]

According to Figure 17, the average utilization of processing ability and bandwidth, or resource utilization, is evaluated. Method III is shown to significantly decrease the F value, resulting in a fair distribution of resources, when compared to Method II, which does not consider fairness. This holds true even when the number of centers increases.
As illustrated in Figure 17, the F value for Method II rapidly increases as ratio z grows, due to the increasing disparity in the number of allocated resources. However, when z exceeds 3.0, the F value for Method II decreases as z increases because the resources required to fulfill user 2's demands are too large to obtain. Conversely, the F value for Method III experiences a slight increase as z magnitudes increase.
Method III allows for a fair distribution of resources in line with the number of resources that each user is expected to request, as illustrated in Figure 18.



Evaluation of the allocated resource
*Cmax1=Cmax2=20, Nmax1=Nmax2=20, H=6; {C=2, N=1} for user 1, {C=2, N=1} for user 1, {C=2*z, N=1*z} for user 2, r1=1/2, r2=1*

Figure.18, [2]

It is commonly observed that when we aim for fairness in resource allocation, resource efficiency tends to suffer. However, it has been found that Method III can achieve almost the same level of resource efficiency as Method II while still striving for fairness, as demonstrated in Figure 17.

2.3. Femtocell Networks

Femtocell networks are designed for residential and small business settings, providing coverage for users within a limited area, usually spanning a few hundred meters. However, managing femtocell networks is challenging due to the limited resources, such as bandwidth and power. These resources must be efficiently utilized to serve multiple users. In their paper, "Dynamic Resource Allocation Algorithm for Femtocell Networks," Tsai and Chiang proposed a real-time resource allocation method that ensures efficient distribution of resources for each user, preventing congestion or overloading of resources.

This algorithm considers several factors to determine the most efficient resource allocation, as briefly mentioned below.

1. Signal strength, the algorithm evaluates the strength of users to ensure that resources are allocated to the user with a weak signal to make a reliable connection.

2. Quality, the algorithm evaluates the quality of the channel between the user and the femtocell to determine the most efficient resource utilization.

3. Priority, the algorithm evaluates the priority of the user and allocates resources to the user with a higher priority.

4. Location, the algorithm evaluates the location of each user and allocates more resources to the user with higher priority to improve resource allocation.

5. Current resource usage, the algorithm evaluates the resources that are currently allocated to the users to ensure there is no congestion or overload of resources.

6. Cost, the algorithm considers the cost of resources to determine the most cost-effective allocation of resources.

7. Performance, the algorithm can consider the performance requirement of each user to ensure that the allocated resources meet the requirement.

The integration of femtocell networks into cloud computing resource allocation presents an opportunity to leverage unique capabilities in new areas. Originally developed for cellular communications, femtocells possess characteristics such as small coverage areas and low power consumption, which make them ideal for cloud computing.

The aim of cloud computing resource allocation algorithms for femtocell networks is to allocate computing resources efficiently based on task and application demands and priorities. This process involves resource provisioning, load balancing, and resource optimization. By deploying effective resource allocation strategies for femtocell networks, cloud computing can achieve optimal performance while minimizing energy consumption and associated costs.

Dynamic resource provisioning algorithm allocates computing resources to tasks based on specific requirements. It monitors the workload, making real-time adjustments for optimal performance and responsiveness.

1. Load-balancing algorithms distribute workloads evenly across resources to prevent bottlenecks and optimize utilization through consolidation.

2. The Resource Usage Optimization algorithm is designed to maximize resource utilization while minimizing wastage. It takes into account factors like application requirements, the availability of resources, and energy efficiency. Advanced optimization techniques, such as genetic algorithms and machine learning, are used to identify the best resource allocation strategies.

Cloud computing environments require efficient dynamic resource allocation to improve resource utilization performance and reliability. An algorithm that considers multiple factors can be used to allocate resources such as processing ability, bandwidth, and storage in real-time on demand. This would ensure that resources are allocated efficiently, resulting in an improved overall performance of cloud environments.

2.3.1. Femtocell Networks and Fairness in Resource Allocation

Ensuring fairness in the allocation of resources for cloud computing intends to offer an equitable chance to all users and applications to access and use computing resources based on their individual requirements. Its objective is to prevent the monopolization of resources, promote user satisfaction, and establish an even playing field for various entities within the cloud ecosystem. Any spelling, grammar, and punctuation errors have been corrected.

1- proximity-based resource allocation system:This means that users or applications located closer to a femtocell can get priority access to available resources. By using this method, resources are distributed more fairly, taking into account the location of users and their proximity to the femtocell. This approach ensures that all users receive a fair share of resources.

2- Load Balancing: Femtocell networks are capable of enabling load-balancing strategies to evenly distribute computing resources among users or applications. By utilizing the small coverage areas of femtocells, the algorithm can dynamically allocate resources based on workload demands, ensuring fairness in resource utilization.

2.4. Linear Scheduling Strategy

A Linear Scheduling Strategy is a popular method used in the cloud computing environment to optimize the utilization of shared resources. It is a dynamic resource allocation method that allocates available resources into time slots and assigns them to users based on a linear schedule. This linear schedule is computed dynamically based on the current state of the system, taking into account changes in project scope, activity duration, and resource availability. The effectiveness of LSS in optimizing resource allocation has been widely acknowledged by project managers.

LSS works by dividing available resources into time slots and assigning them to users based on their current demand for resources. The optimal linear schedule is then computed based on the current demand, and resources are allocated to users based on this schedule. As the demand for resources changes, the algorithm recalculates the linear schedule and reallocates the resources accordingly.

This algorithm has been widely studied and applied in cloud computing environments, with many research studies proposing LSS-based algorithms for dynamic resource allocation in the cloud. These studies have found that LSS is effective in maximizing resource utilization and minimizing user waiting time, while also balancing resource utilization and fairness among users.

Overall, LSS improves project performance by enabling project managers to identify and address potential bottlenecks and conflicts in resource allocation. By visualizing the project timeline, managers can optimize resource allocation to ensure that critical activities receive necessary resources promptly. This leads to reduced project delays, enhanced productivity, and improved overall project performance measures such as on-time delivery and cost efficiency.

2.4.1. Fairness

In the context of dynamic resource allocation, fairness plays a critical role in ensuring an equitable distribution of resources among various activities or project teams. The LSS framework provides transparency in resource allocation, empowering project managers to allocate resources fairly and avoid any favoritism or bias. The LSS takes into account the sequence and interdependencies of activities, ensuring that resources are allocated based on project priorities and the unique requirements of each activity, thereby promoting fairness in resource distribution.

2.4.2. Resource Utilization

The integration of technology and advanced modeling techniques has improved resource utilization in the Linear Scheduling Strategy (LSS). By using optimization algorithms, artificial intelligence, and machine learning, it is possible to optimize resource allocation by considering factors such as resource skill sets, availability, and project constraints. These technologies enable more accurate predictions and real-time adjustments, resulting in improved resource utilization and better project outcomes.

When it comes to project management, LSS provides valuable benefits by ensuring performance, fairness, and resource utilization. It optimizes resource allocation, enhances project performance, ensures fairness in resource distribution, and maximizes resource utilization, ultimately leading to a successful project execution.

3. Conclusion

3.1. Firefly

Cloud computing enables the delivery and consumption of IT services over the Internet. Virtualization plays a crucial role in dynamically allocating resources based on user requests and providing flexibility in scheduling policies. By using virtualization in cloud computing, many users can have concurrent access to cloud services. Resource allocation is a complex task for both consumers and providers. Providers should ensure the availability of adequate resources for each user or application in a cloud environment, while consumers and providers need to achieve profitability while balancing factors such as QoS, fairness, processing costs, and workload.

The proposed algorithm makes use of the advantages of the Firefly algorithm and fuzzy approaches. It is clear that the algorithm is efficient in terms of load balancing. By using this proposed approach, maximum resource utilization can be achieved, and SLA violations can be reduced.

3.2. Optimal Joint Resource Allocation

The paper proposes an optimal joint multiple resource allocation method, method II, which allocates processing ability and bandwidth simultaneously for each request on an hourly basis, with a dedicated resource for each service request. Based on the results obtained from the simulation evaluation, it appears that Method II could be a more viable option when compared to conventional allocation methods. This may potentially result in a decrease in the probability of request loss and a reduction in resource consumption. The findings suggest that Method II presents a promising solution to address the challenges of resource allocation in the current business environment.

The proposed paper outlines an equitable approach to multiple resource allocation in cloud computing environments. Method III distributes resources by taking into account the anticipated resource requirements of each user. Through simulation evaluations, it was determined that Method III achieves a just distribution of resources among multiple users, while maintaining resource efficiency comparable to the conventional method.

3.3. Femtocell Dynamic Resource allocation

Managing resource allocation, such as power and bandwidth, is crucial in a femtocell network. The algorithms used for this purpose are both robust and weak. On one hand, they enhance the efficient utilization of resources, dynamic provisioning, load balancing, and computing resource optimization, thereby leading to better network performance and user experience. Additionally, these algorithms aid in increasing the scalability of the network. However, on the other hand, the implementation of such algorithms can be complex and can result in slower response times, leading to increased latency.

3.4. Linear scheduling algorithm

This algorithm is commonly used in cloud-computing environments for resource allocation. However, these algorithms have weaknesses and strengths when allocating resources to the environment. weaknesses in LSS are that it is inflexible, not scalable, and can be complex to implement in large-scale cloud environments. They are best suited for a predictable, consistent workload, but struggle with sudden changes in demand, which results in poor resource allocation. In terms of robustness, the linear scheduling algorithm is predictable, fair, and efficient in terms of resource allocation, quickly allocates resources to tasks, and ensures fairness based on task priority.

4. Result

Resource allocation is a crucial factor in determining user experience and the overall efficiency of cloud computing systems. In this context, the Firefly algorithm, when combined with fuzzy approaches, emerges as a promising solution for load balancing. By leveraging virtualization, concurrent access to cloud services for multiple users becomes a feasible option, thereby enhancing the flexibility of scheduling policies. The proposed algorithm not only draws on the benefits of the Firefly algorithm but also demonstrates its efficiency in load balancing. Employing this approach holds the potential to maximize resource utilization while minimizing SLA violations.

The pursuit of optimal joint resource allocation has prompted the introduction of method II, a strategy that simultaneously allocates processing ability and bandwidth on an hourly basis. This method is supported by dedicated resources for each service request and has proven effective in reducing request loss probability and total resource usage.

Furthermore, the emphasis placed on equitable resource allocation in a cloud computing environment has led to the development of method III, which strikes a balance between fairness and resource efficiency. Our findings affirm that method III achieves fairness without compromising resource efficiency, setting it apart from conventional methods.

These results hold significant implications for organizations seeking to optimize resource allocation in a cloud computing environment. The implementation of method II and method III can be expected to yield tangible benefits, including reduced request loss probability and optimized resource usage.

Within the domain of femtocell networks, the dynamic resource allocation algorithm endeavors to effectively manage power and bandwidth utilization. Despite its robustness in enhancing resource utilization, facilitating dynamic provisioning, and improving load balancing, the algorithm is not without its challenges. Specifically, the complexity and potential latency issues pose a significant hurdle. Thus, it is crucial to carefully evaluate the trade-off between efficiency and implementation complexity. Given these considerations, it is imperative to approach the algorithm with a judicious and meticulous mindset. Within the domain of femtocell networks, the dynamic resource allocation algorithm endeavors to effectively manage power and bandwidth utilization. Despite its robustness in enhancing resource utilization, facilitating dynamic provisioning, and improving load balancing, the algorithm is not without its challenges. Specifically, the complexity and potential latency issues pose a significant hurdle. Thus, it is crucial to carefully evaluate the trade-off between efficiency and implementation complexity. Given these considerations, it is imperative to approach the algorithm with a judicious and meticulous mindset.

Meanwhile, the linear scheduling algorithm, a common player in cloud-computing environments, exhibits a mix of strengths and weaknesses. Its predictability, fairness, and efficiency make it suitable for consistent workloads. However, limitations surface in terms of inflexibility and scalability, particularly in large-scale cloud environments. Sudden changes in demand pose challenges for this algorithm, leading to suboptimal resource allocation. Despite its drawbacks, the linear scheduling algorithm stands out for its predictability and efficiency in allocating resources based on task priority. Resource allocation algorithms in cloud computing present unique challenges and advantages.

5. References

1- Dynamic Resource Allocation Using Improved Firefly Optimization Algorithm in Cloud Environment, to link to this article: https://doi.org/10.1080/08839514.2022.2055394

2- Optimal Joint Multiple Resource Allocation Method for Cloud Computing Environments, International Journal of Research and Reviews in Computer Science (IJRRCS), Department of Computer and Information Science, Seikei University, Japan

3- Dynamic Resource Allocation in Hybrid Access Femtocell Network, to link to this article: https://www.hindawi.com/journals/tswj/2014/539720/

4- Dynamic Resource Allocation for Distributed Systems and Cloud Computing, May-June2020 ISSN: 0193-4120 Page No.22417–22426

5- Linear Scheduling Strategy for Resource Allocation in Cloud Environment, International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.2, No.1, February 2012, link to this article: https://doi.org/10.5121/IJCCSA.2012.2102

6- A game-theoretic approach for dynamic resource allocation in cloud computing with linear scheduling strategy. Journal of Network and Computer Applications, Chen, Y., Huang, X., Li, J., & Liu, X. (2018), 108, 39-48 link to this article: https://link.springer.com/article/10.1007/s11227-009-0318-1.