Advanced scheduling algorithm for multi resource scheduling with minimum time consumption

Santosh Shakya Santosh Shakya
National institute of technology, raipur

Priyanka Tripathi Priyanka Tripathi (ptripathi.mca@nitrr.ac.in)
National institute of technology, raipur

Research Article

Keywords: virtual machine, scheduling, cloud, resources, IoT

Posted Date: April 7th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2759565/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Abstract

The scheduling of appropriate resources for cloud workloads is a difficult task, as it depends on the quality of service needs of cloud applications. Due to their limited data storage and energy capabilities, IoT applications demand high-speed data transfer and low latency. Many IoT devices generate data continuously and want to store it on the cloud quickly and efficiently. Dynamic virtual machine (VM) allocation in cloud data centers (DCs) is taking advantage of the cloud computing paradigm. Each VM request is characterized by four parameters: CPU, RAM, disk, and bandwidth. Allocators are designed to accept as many VM requests as possible, considering the power consumption of the IoT device’s network. Resource scheduling and time consumption is the two most significant problems in cloud computing. To overcome this problem, in this paper, the author has extended CloudSim with a multi-resource scheduling and minimum time consumption model that allows a more accurate valuation of time consumption in dynamic multi-resource scheduling. The author proposes a new scheduling algorithm advance scheduling algorithm (ASA), which provides a better solution to other scheduling algorithms like Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). Also, tries to reduce energy consumption and time to give a task to the VM.

I. Introduction

The Internet of Things (IoT) is receiving a lot of interest across all industries owing to its potential to link numerous devices that produce a lot of data. Because of its cutting-edge intelligent communication capabilities, the Internet of Things (IoT) is welcomed in every industry. IoT allows for the development of a variety of applications, some of which are well-known in the industrial, healthcare, and transportation sectors, etc., The standardization of architecture and high quality of services enhance the global economy and the level of living for each individual. Since the Internet of Things (IoT) networking technologies can link millions of devices, this lowers processing complexity and boosts an organization's or an individual's productivity. According to Cisco, in 2021, IoT devices will create almost 2013 ZB of data. However, using conventional computer techniques to manage such an abundance of data is complex for data management firms. Consequently, cloud computing is employed in many industries to collect vast amounts of data. IoT and cloud integration offers advanced processing and storage capabilities.

Cloud computing has become popular due to its real-time data analysis, storage, and processing capabilities for managing IoT device operations. Smartly designed IoT devices use cloud computing, and IoT apps use the cloud to connect and interact with other connected items. From this how the cloud of things paradigm has developed. This integrated strategy provides wide-ranging computing prospects in the areas of business, healthcare, smart cities, agriculture, transportation, etc., The pay-per-use model of cloud computing, which supports IoT applications, provides customers with a wide range of services and enables access to resources from anywhere, at any time. To manage the data in real-time applications and maintain a higher level of service, IoT applications need high-capability systems with a short latency [1]. Cloud computing is an Internet-based method in which all programs and information are stored in a
cloud made up of a large number of machines connected in complex ways. The main difficulty for cloud data centers is demonstrating how millions of customer requests are examined and handled efficiently and effectively. Load-balancing techniques are required to boost the adaptability and scalability of cloud data centers. The load-balancing method is one of the most critical problems with the distributed computing system. The key reason why many researchers regarded and handled the cloud computing load-balancing problem as an NP-hard problem might be due to the large-scale resources and numerous user requests [2]. Figure 1 show the architecture of VM and working of VM manager.

The process must be mapped to the available resources according to their requirements and characteristics to schedule tasks. Cloud load balance is one of the crucial factors that should always be taken into account while scheduling. When scheduling, two options should always be taken into account in such a way that when users' jobs are in high demand, each job should be handled in a different processor in order to balance the load, and the system must select how to process tasks in order to shorten the time it takes to complete them. Each job should be handled in a different processor to balance the load, and the system must select how to process tasks to shorten the time it takes to complete them [2]. A client can request information, shared resources, software, and other services using cloud computing at any time and to his requirements. It is an on-demand service, and the phrase is often used online. The entire internet may be seen as a cloud. Additionally, using the cloud lowers capital and operating costs. However, load balancing is a major issue in cloud computing, and a distributed solution is always required.

One of the most crucial but challenging aspects of cloud computing is task (resource) scheduling. Tasks (user requests) submitted by end users to the cloud system are processed by a scheduling algorithm and distributed to the accessible virtual machines (VMs). Job scheduling aims to improve task execution and make the most use of available resources. Task scheduling is a multi-objective optimization issue that falls under complex non-deterministic polynomial problems [3]. An essential index in the cloud computing service is the performance of the VM resources, which can be used to gauge the service, likelihood, and degree of service capability of the cloud service providers. Knowledge about available VM resources is often essential for evaluating VM resource performance. The evaluation of VM resource availability can result in the description of the currently utilized and available VM resources, which is advantageous for resource scheduling, distribution, and transfer [4].

A. Load Balancing in Cloud

The load balancing method ensures that any node in the system is ever idle or overloaded for any length of time by distributing the workload among various nodes in the environment. The fundamental goal of load balancing in the cloud domain is to dynamically allocate the load across the nodes to meet user demands and to provide optimal resource utilization by classifying the total amount of available load to different nodes. The researcher also investigated the load balancing algorithm's potential to enhance cloud services using simulation. Cloud computing is a new, developing technology in academia and business [5]. Strategies for resource allocation and prediction Future resource demands were forecasted
for energy-efficient resource use in dynamic resource allocation and virtual machine deployment by constructing an exponential weighted moving average based on the previous behaviors of the VMs, and the future state of the VMs was projected. For power-efficient data centers suggested, a strategy for VM consolidation based on the forecast of multiple resource use. [5].

B. Resource Allocation in Cloud Computing:

Resource allocation in cloud computing refers to the selection of a virtual machine to satisfy specific requirements set out by customers. Another feasible method for allocating resources in the cloud is to manage these workloads by assigning them to virtual machines [11]. Simply expressed, this means deciding when a computational activity should start or terminate based on: 1) Resource distribution, 2) Duration and 3) the prior party's behavior and Relationships with the forerunner [6].

C. Need of Resource Scheduling

The initial goal of resource scheduling is to determine the best resources for the on-time scheduling of the right workloads and to improve resource utilization. The resources needed for a task should be kept to a minimum to optimize throughput or maintain acceptable service quality. The optimal resource workload mapping is necessary for improved resource scheduling. Finding the appropriate and sufficient workload that enables the scheduling of various workloads and can satisfy the quality of service criteria, such as CPU usage, availability, dependability, and security, for cloud workloads, is the second goal of resource scheduling [7].

II. Motivation

The scheduler is one of the essential service of the cloud middleware that can be combined to create an actual cloud environment. Task scheduling is responsible for choosing the optimal virtual machines or computing resources while considering resource use. We want to use as many resources as possible while minimizing the overall completion time (makespan) and response time, load task scheduling and balancing work together to satisfy both users and service providers. We make an effort to balance the workloads across the virtual machines and utilize the best algorithm to carry out the activities while optimizing resource consumption and shortening completion time.

III. Related Work

The duties in the space-sharing system will be carried out one at a time. It denotes that each CPU core of its CPU only performs one task at a time. The waiting queue should contain the tasks still allocated to that VM. By selecting a job from the overloaded VM's waiting queue and distributing it to the under loaded VM, the task migration in load balancing will be simpler [11]. However, in a time-shared system, the tasks are completed concurrently in a time-shaded way that mimics the completion of work in parallel mode. In this load-balancing scenario, task migration would be pretty challenging due to the time-sliced execution of the processes. Therefore, using the time-sliced method, a set number of job instructions will
be in the finished condition approximately 90% of the time [12]. Due to the loss of previously finished sections in the higher-loaded VM and the job's early execution affecting the other tasks' execution time in the higher-loaded VM, migrating from a higher-loaded VM to a lower-loaded VM is quite costly [13]. Depending on their capabilities, the scheduling and load balancing method should accomplish the best/minimal task migration with equal load distribution across the resources. In a cloud setting, this method/process helps to get the best/fastest execution time [14]. By considering the jobs that do various tasks and their interdependencies, the algorithm should also consider the unpredictable nature of task allocation to the appropriate VMs. The algorithm should work well in homogeneous and heterogeneous settings with a range of task durations. The literature has been examined with this goal in mind, and an algorithm suggestion has been made[15]. To address load-balancing problems, the developers also employed heuristic and metaheuristic models. The protected equilibrium is tracked on a distributed network. Three models were also used to distribute network security and load balancing issues. For instance, it allows a mobile agent to broadcast to every node on a distributed network [16]. Load balancing is a crucial problem that cloud computing experts have dealt with. Load balancing works by dividing work among resources fairly and effectively, eventually resulting in high user satisfaction and increased system productivity [17].

IV. Problem Formulation

The scheduling and load-balancing architecture shown in Table 1 illustrates how the scheduler has the logic to select the optimal VM and distribute the work among VMs under the defined methodology. If the runtime arrival jobs are planned for the least-used virtual machine at that moment, the best virtual machines will be assigned to them based on the quality of service parameters.

The load balancer decides at runtime whether to move the burden from the severely loaded VM to the idle or least loaded VM whenever it discovers an idle or least loaded VM. To know more about each VM's capacity, current workload, and the number of jobs either being executed or in the waiting queue, the resource monitor establishes a connection with each VM's resource probe. The user communicates the task requirements, which include the time required to accomplish the tasks, to the scheduler for operational choices.

V. Proposed Methodology

The main objective of the research proposed in this paper is to provide a new algorithm for resource scheduling challenges in cloud computing environments by using a noble tree-based technique. In the proposed approach, the number of VMs is supposed to be like a node in a tree and find a better node suitable for taking a load from IoT device networks. In physical layer hardware is divided in number of virtual machine which are equal capabilities. VM divided into some group according to generation of that and we also use some of comparator which identifies low workload VM (Higher fitness value). In Fig. 2 show the internal VM and comparator.
Suppose, in a cloud environment, N number of different VMs are present, and all have a different load status because of CPU busy in different task provided by task provider. The task provided select a higher fitness value VM according to the fitness value.

Comparator use Tournament tree algorithm (TTA) for finding the VM in decreasing order according to fitness value. All comparator use same technology for finding VM machine for taking a load which is provided from distributor. All comparator send their list to the VM selector and again TTA apply on selector for finding final list of VM which have higher fitness value.

THE BASICS OF ACO, PSO AND ABC Algorithms:

The primary lesson learned from ACO is to mimic how ants forage to find plentiful food sources. To communicate with one another, they use a specific type of chemical pheromone [18]. Ants first seek out food randomly, and when they spot a trail leading there, they release a certain quantity of pheromone along the route. Other ants can advance to the food source by detecting pheromones on the ground. This procedure continues until most ants are persuaded to choose the shortest path with a significant concentration of pheromones gathered along it [19]. The ACO algorithm uses an internal parallelization system and a positive feedback mechanism. The downsides of the algorithm converging to a local optimum solution include overhead and the stagnation problem.

Particle Swarm Optimization (PSO) PSO was proposed in 1995. Other particles follow the best swarm members as they fly across an environment, guiding them into advantageous locations [20]. This algorithm begins by giving each particle in the search space a random place. It uses the global best-known site and the best position a particle is aware of to advance the status of each particle consecutively based on its velocity. The particles eventually congregate around one optimum or many optima [21]. The method keeps track of the stopping value, which indicates when the algorithm should finish, and the global best-known location (gBest). Personal best (pBest) value, velocity value, and data representing a solution make up each particle. The velocity value indicates how much the data may be altered. The velocity value is determined by how far the particle's data is from the target. The next particle requires a large amount of velocity. The people furthest from the meal would attempt to fly faster toward the best bird to catch up with the others, just as the bird did [21].

The Man-Made Bee Colony (ABC) The 2005 proposal of the honey bee swarm's expert foraging behavior served as the basis for the ABC algorithm. A regular hive may house between 5,000 and 20,000 bees. Honey bees believe that as time passes, their colony will have new roles. Mature bees (those between 20 and 40 days old) often become foragers. The three roles that foraging bees typically play are active foragers, scout foragers, and inactive foragers [22]. Bees actively forage, visit a food source, investigate nearby sources, gather food, and then return to the hive. Bee scouts scan the vicinity around the pack for various fresh food sources. Some of the foraging bees might stop foraging at any time. These dormant foragers are positioned behind the hive entrance. Depending on the quality of the food supply, active foragers or scouts may perform a waggle dance for the waiting inactive bees as they return to the hive. This waggle dance communicates the location and amount of food to the dormant bees. Waggle dancing
provides this information to inactive bees, who may become active foragers. Typically, a busy bee gathers food from a particular source until that supply is depleted. An active bee now turns into an inactive forager [23].

CLOUD SCHEDULING BASED ACO

ACO is utilized in this part to identify an ideal resource allocation for dynamic tasks in the cloud to reduce the makespan of jobs. The pseudo-code of the ACO procedure is shown in Fig. 3. Ants are initially more guided in their choice to develop a solution by the quantity of virtual pheromone trail $T_{ij}(t)$ on the edge that connects the job with VM.

CLOUD SCHEDULING BASED ABC

A collection of active, inactive, and scout bees are produced using the ABC algorithm. At the initial stage of the ABC algorithm, the quantity of active, passive, and scout is determined. The overall optimal solution discovered by any bee at any iteration is manipulated by this method. Figure 3 displays the suggested ABC procedure's pseudo code.

CLOUD SCHEDULING BASED PSO

A certain amount of particles are used to create PSO.

Each particle relates to a potential fix for the current scheduling issue. The order of the VMs is represented by an array of VM IDs, which is how the solution is shown. In the variety of VM IDs, the first task will be mapped to the first ID, the second to the second ID, and so on. The PSO algorithm searches for the best solutions based on the chosen particles. The makespan length, calculated by Equation, is related to the solution quality (3). The proposed PSO procedure's pseudo code is displayed in Fig. 5. The number of particles and the parameters are initialized during a step of the proposed PSO algorithm called initialization. The V Max variable determines the maximum velocity. The best variable, which symbolizes the overall ideal solution, is null. Each particle has a p Best variable that denotes the solution's best fitness value.

B. Proposed ASA Algorithm:

1. Collect the first task slot $T$ Created by IoT devices

   $T = (T_1, T_2, T_3, \ldots, T_n)$

2. Task is arranged in a task queue and forward to the TTA task Distributer.

   2.1 Generate the new virtual machines as requirement and store in VM collector. Let us assume that $n$ number of VMs is specified in the list. VM collector is arranging all the VM according to fitness value.

   2.2 Each collector work on TTA algorithm
2.2 In collector arranges VM according to Fitness value Find the Highest fitness value VM.

Step 3: collect all VM from all collector and rearrange it on selector.

3.1 Selector apply again TTA algorithm for arranging the all VM according to fitness values.

Step 4: Distributer allocate first task to the highest fitness value VM and next task to the next fitness value VM and so on.

Step 4: After completion of first slot for next slot again applying the proposed algorithm and find highest fitness value VM.

Step 5: Step 5: End.

Algorithm for TTA:

Step 1: Initialize VM from 0 to N-1

Step 2: Do

Find Fitness value of all VM

Fitness value: % use of CPU, RAM, Disk, Bandwidth

Step 3: Each VM like a Node of tree

Step 4: Input 0 to N-1

While len (nodes) > 1:

Winners = []

For i in range (0, len (nodes), 2):

Left = nodes[i]

Right = nodes[i + 1] if i + 1 < len (nodes) Else None

Winner = left.winner if not right or left.winner > right.winner Else right.winner

winners.append (Node (winner = winner, left_child = left, right_child = right))

Nodes = winners
Step 5: Stop

C. Fitness Function:

Suppose cloud computing contains \( n \) VMs, and each VM has a fitness value. The fitness value is based on CPU, RAM, disk, and bandwidth. The purpose of the proposed algorithm is to reduce power (P), migration cost (MC), and memory utilization (MU) while the load is balanced. Based on the above condition, the fitness function is designed, which is given by the following equation. Table 1 shows the fitness value calculated by four major parameters according to the quality of service.

\[
\text{Fitness function } F = \text{logical Addition of parameter values}
\]

Vi. Result

The entire VM store in Distributer and ready for take a task from task scheduler.

Suppose there are 2 queue on distributer

\[Q1 = \{T1, T2, T3, \ldots, Tn\}\]

\[Q2 = \{VM1, VM2, VM3, \ldots, VM4\}\]

As shown in Fig. 3, the highest fitness value virtual machine (VM) is more suitable for upcoming loads coming from the task queue. After assigning the load to the first VM, the next task is provided to the next fitness value VM. All parameter values are set on the bases of their present time working load and total working capacity.

Utilizing CloudSim and datasets, the implementation of these techniques is validated. Here, we're looking at distinct tasks whose burst quality of service parameters are shown in the table and also offering the average fitness value based on the parameter value.

The average waiting time and turnaround time and makespan time for all job scheduling strategies are shown in Fig. 3. The Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and The Artificial Bee Colony (ABC) algorithm work on previous resources pattern and give a average result according to the QoS parameter. TTS is superior in every way, as shown by the algorithm analysis and their Gantt charts. For the task scheduling algorithms, various load-balancing criteria have been
considered. The waiting periods and typical turnaround times and makespan time for those resource scheduling criteria in the algorithms are summarized.

**Parameters Setting of Cloud Simulator**

The trials are run on a simulation platform with 10 Datacenters, 50 VMs, and 100–1000 processes. The job has between 1000 MI (Million Instructions) and 20000 MI in it. Table 1 displays the cloud simulator's parameter settings according to cloud structure.

Using two opposing criteria, the ABC, PSO, and ACO may efficiently and effectively explore a search space: investigation of the search space and use of the most effective answers. They keep track of the best solution overall that is created by any member and related to the makespan length.

To address the issue of cloud job scheduling, the ABC and PSO algorithms use models of the behavior of swarming flies and honey bees, respectively. Because there are multiple activities for each type of bee in the ABC algorithm, it performs better than the PSO algorithm, which has the same activity for all particles. In the ABC algorithm, there are scouts, actives, and inactive bees, and each kind has a unique search technique. This explains why the ABC algorithm works better than the PSO method.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Total number of VMs</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS</td>
<td>500–2000</td>
<td></td>
</tr>
<tr>
<td>VM memory(RAM)</td>
<td>128–2048</td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td>500–1000</td>
<td></td>
</tr>
<tr>
<td>cloudlet Scheduler</td>
<td>Space shared and Time shared</td>
<td></td>
</tr>
<tr>
<td>Number of PEs requirement</td>
<td>01–04'</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datacenter</th>
<th>Number of Datacenter</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Host</td>
<td>2–6'</td>
<td></td>
</tr>
<tr>
<td>VM Scheduler</td>
<td>Space shared and Time shared</td>
<td></td>
</tr>
</tbody>
</table>

Make span (MS): Total amount of time needed to do all tasks and distribute resources to system users. It is an important metric for the scheduling procedure in the cloud environment. It is employed to determine how long it takes to do a series of tasks. A successful resource scheduling method needs a minimum MS.
Graph 1 shows the average waiting time for the entire algorithm which are used in the paper. Clearly show the makespan time of proposed ASA algorithm has lowest makespan time to other algorithms on different number of tasks.

Average waiting time (AWT): Average waiting time is the average of the waiting times of the processes in the queue, waiting for the scheduler to pick them for execution.

Graph 2 shows the average waiting time for the entire algorithm which is used in the paper. Clearly show that waiting time of proposed ASA algorithm has lowest waiting time to other algorithms on different number of tasks.

Turnaround time (TT): Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to its completion.

Turnaround time $= \text{Completion Time} - \text{Arrival Time}$

Graph 3 shows the average waiting time for the entire algorithm which is used in the paper. Clearly show that turnaround time of proposed ASA algorithm has lowest turnaround time to other algorithms on different number of tasks.

Vii. Discussion

In the discussion area, we've included a graphic showing the top three job scheduling algorithms and the ASA algorithm proposal. Table 1 shows the parameter setting perfuming on the cloud and comparing 4 task scheduling algorithms with the ASA algorithm to give the best result compared to other algorithms. With the compression of the algorithm, the author finds the makespan time, waiting time, and turnaround time compression of 3 algorithms (PSO, ABC, ACO) on the different numbers of tasks (100, 200, 400, 600, 800, 1000). Finally proposed ASA algorithm is a better solution for the come and energy consumption.

Viii. Conclusion

To increase the performance of the cloud, we have developed an effective cloud scheduling method in this research. The suggested technique (ASA) locates the cloud-maintained virtual machines with the most significant fitness values to provide specific scheduling patterns for the virtual machines situated in those compute nodes. The incoming task was assigned to an appropriate VM. Incoming tasks generated by IoT devices yield to the highest fitted virtual machine and operate on task.

Declarations

Author has declared that the Submitted Research Paper is original work and no part of it has been published anywhere else in the past. I take full responsibility, that if in future, the paper is found invalid
according to basic rules, the last decision will be of the Authorities concerned. Any form of plagiarism will lead to disqualification of the paper.

**Ethical Approval:** Not applicable.

**Competing interests:** Not applicable.

**Funding:** Not applicable.

**Availability of data and materials:** Not applicable.

**References**


Graphs

Graphs 1-3 are available in the Supplementary Files section.

Figures

![Diagram]

**Figure 1**

Task provided to VM through VM manager
Figure 2

Working of Comparator
Figure 3

VM selection for upcoming task
Fig 3: Pseudo code of the ACO procedure

Steps: I. Initialize:
Set t=1.
Set B solution=null.
Set an initial value $\varphi$ = c for each path between tasks and VMs.
2. Place the no a ants on the starting VMs randomly.
3. For k: =1 to no a do
   Do ants_trip while all ants don't end their trips
   Every ant chooses the VM for the next task according to Eq. (I).
   End Do
4. For k: =1 to m do
   Compute the length $L_k$ of the tour described by the ant k according to Eq. (3).
   Check to update the B solution with the best founded solution.
5. Apply local pheromone update.
6. Apply global pheromone update.
7. Increment t by one.
8. If (t < tma,) Go to step 2
   Else
      Print B solution
   End If
9. Return
Input: List of Cloudlet (Tasks) and List of VMs  
Output: the best solution for tasks allocation on VMs  

Steps: I. Initialize:  
Set value of parameters Number _of particles, V_Max,  
tmax.  
Set t=1.  
Set gBest=null.  
Generate random solution for each particle  

2. For each particle  
Calculate solution fitness value  
{  
If the fitness value is better than pEest  
Set pEest = current fitness value  
End If  
If pEest is better than gEest Set gEest = pEest  
End If  
}  

3. Sort particles by their pEest, from best to worst  
4. For each particle  
{  
Calculate particle Velocity  
Use Velocity to update particle  
}  
5. Increment t by one.  
6. If (t <tma,)  
Goto step 2  
Else Print  gBest.  
End If  
7. Return  

Figure 5  

Fig 4: Pseudo code of the ABC procedure
Fig 5: Pseudo code of the PSO procedure

Steps: 1. Initialize:
Set value of parameters Number _of_Bees, Number _of_inactive, Number _of_active, Number _of_Scout, Max_number _of_Vists, Prob _Mistake, Prob _Presuasion, tmax.
Set t= I.
Set BSolution=null.
2. Generate Random Solution for each bee
3. Check to update BSolution
4. For k= I to Number _of_Bees
   IF k is active
   Perform ActiveBee ()
   Else IF k is scout
   Perform ScoutBee ()
   Else
   Perform InactiveBee ()
   End IF
5. Increment t by one.
6. If (t <tmax)
   Goto step 4
   Else
   Print Bsolution
   End if
7. Return
Figure 7

Fig 4. VM selection on the basis of tournament

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Graph1.png
- Graph2.png
- Graph3.png