

Network routing and scheduling architecture in a fully distributed cloud computing environment

Vijaya Kumar S¹, Muthusamy Periyasamy¹, R. Radhakrishnan², Tamilarasi Karuppiah³,
Thenmozhi Elumalai³

¹Department of Computer Science and Engineering, Shri Venkateshwara University, Uttar Pradesh, India

²School of Computer Science and Engineering, Galgotias University, Delhi NCR, India

³Department of Information Technology, Panimalar Engineering College, Chennai, India

Article Info

Article history:

Received Mar 11, 2024

Revised Jul 30, 2024

Accepted Aug 5, 2024

Keywords:

Central server

Clustering selection algorithm

Distributed computing

Network routing

Three-layer cloud dispatching

ABSTRACT

Distributed computing has turned into an indispensable application administration because of the colossal development and fame of the internet. However, determining the allocation of various tasks to suitable service nodes is crucial. For the reasons expressed over, an effective booking strategy is expected to work on the framework's exhibition. As a result, three-layer cloud dispatching (TLCD) design is introduced to further develop mission planning execution. The assignments should be arranged into various sorts in the primary layer in radiance of about their personalities clustering selection algorithm is composed of then recommended in second layer towards dispatch the undertakings to significant help bunches. Likewise, to further develop booking effectiveness, another planning technique for third stage proposes dispatching that job here to system thinking in a central server. As a rule, the proposed TLCD design yields the quickest work finishing time. Moreover, in cloud computing network architecture, load balancing and stability can be achieved.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Muthusamy Periyasamy

Department of Computer Science Engineering, Shri Venkateshwara University

244236 Uttar Pradesh, India

Email: muthu.namakkal@gmail.com

1. INTRODUCTION

Distributed computing has turned into a fundamental and profoundly sought-after application administration in light of the fast turn of events and ubiquity of the internet [1]. A colossal number of heterogeneous servers, stockpiling, and supporting hardware will be required in a distributed computing climate to address the developing stockpiling requests of big data. Programming-as-a-service, platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS) are instances of distributed computing server types [2]. The greater part of the ordinarily utilized programming and instruments, like Gmail and Maps, are presented as SaaS [3]. One kind of PaaS stage is the Google APP Engine, which gives a stage to clients to run programs [4]. Ultimately, IaaS permits clients to lease the equipment assets expected to make their own structure, for example, cloud-based service [5], [6]. Regardless of such cloud's administration utilized, there was indeed common trademark: each cloud server has an interesting arrangement of computational capacities. A productive planning technique is required in light of the reasons framed above, and this is a critical trouble in contemporary distributed computing settings.

As per the normal cloud bunching design, the framework possibly considers heterogeneity while executing booking processes and disregards the heterogeneity of errands that come from various stages and its classifications are not conspicuous by hubs in the group [7]. Accordingly, run of the mill cloud structures

presently contain difficult issues brought about by heterogeneous positions. Cloud administration types, then again, are incredibly changed. Along these lines, distributed computing is more challenging to oversee and has more terrible dependability. Along these lines, booking cloud conditions in the future is more troublesome.

Consequently, the three-layer cloud dispatching (TLCD) engineering is introduced while diverse hubs and missions coexist inside the cloud framework, it is necessary to address the scheduling problems. In the base is quite a clustering for allocating classes (CAC) [8], [9] has been founded primarily characterize heterogeneous exercises to decrease mission idleness and over-burden. In the CAC layer, errands can be partitioned into three classes in light of the IaaS, SaaS, and PaaS classifications [10]. A while later, the homogenous undertakings can be shipped off the suitable help classification bunches in the accompanying layer.

The cluster scheduling algorithm (CSA) can allocate homogenous responsibilities to suitable bunches [11] in the subsequent layer, which is alluded to as the cluster selection (CS) layer, to work on the framework's unwavering quality. Also, this layer can bring down the expense and culmination season of errand booking. The third layer, client source node sampling (SNS), is liable for dispatching responsibilities to support hubs. Advanced clustering suffrage scheduler (ACSS) at a high level is presented here to further develop asset utilization and accomplish load adjusting.

2. LITERATURE REVIEW

For different application conditions, numerous booking strategies have been introduced. Shojafar *et al.* [12], for instance, proposes an energy-effective versatile asset scheduler design for offering continuous cloud administrations to vehicular clients. The proposed convention's key commitment is to amplify by and large correspondence in addition to processing energy productivity while likewise meeting application-actuated rigid quality of services (QoS) demands minimum transfer rates, maximum delay, including delay butterflies [13]. Shojafar *et al.* [13] present a joint processing in addition to correspondence streamlining system that utilizes virtualization innovations to give clients QoS, boost energy reserve funds, and meet green distributed computing objectives in a totally dispersed way. As a general rule, these proposed calculations and designs can help with the exhaustive reconsidering of booking calculation plan.

The continuous kind and the clump type are the two kinds of planning calculations that can be provided in light of the booking time. Fundamentally, continuous errands are doled out to cloud server hubs when they are gotten. In the bunch mode, then again, got errands are collected for a while prior to being conveyed to cloud server hubs. A cluster planning calculation can give improved results than a constant booking approach. This is on the grounds that a bunch-based planning calculation can consider the Mission consequences of all positions [13], [14].

To send the project to internet servers' hub inside the distributed software cloud organization, various booking calculations [15], [16] have been created, including the minimum, maximum-minimum, suffrage, and maximum suffrage calculations [15], [17], [18]. Notwithstanding, such estimates primarily take into account the factor of estimated consumption time (ECT) and disregard the server hub's heap state. Accordingly, the exhibition misses the mark concerning assumptions, and the insignificant fruition time can't be met. The assignment has the lowest ECT between all unallocated duties. T, for example, is referred to as minimum ECT in the min calculation. Following that, the occupation with the least ECT can be picked and dispatched to the suitable waiter hub. The recently paired work is then removed out of a designated T, and practice continues till all the undertakings have been sent. In this present circumstance, the responsibility is probably going to end up being unequal since there are such a large number of undertakings that should be planned. The ECT is likewise used for work dispatching in the maximum-minimum calculation. In the maximum-minimum calculation, the occupation with the most noteworthy complete ECT is constantly relegated, bringing about an extensive ascent in generally ECT [10], [17]. Besides, each of the previously mentioned methods could without much of a stretch outcome in higher-limit server hubs being doled out additional positions than lower-limit server hubs. Cloud server hub responsibilities are lopsided and wasteful. Thus, the Suffrage calculation [15], [17] was proposed as an improvement technique for decreasing the jobs of cloud server hubs. To assign labor, overall suffrage value (SV) is subtracted from the primary ECT to get an anticipated value. The occupation with the most elevated SV worth can then be picked and despatched to the cloud server hub with the least ECT. At the point when the quantity of delaying assignments is extremely perfect, be that as it may, the Suffrage calculation can't accomplish proficient execution.

Accordingly, to resolve the above issue, the MaxSuffrage technique [3], [18], [19], which is an enhancement for the Suffrage calculation, incorporates three stages. In the SV_i computation stage, all assignments' SV values should not set in stone. In the MSV_i calculation step, the ECT from phase I that has the second-highest SV value will be chosen as the MaxSuffrage value (MSV) esteem. In the final phase, called the mission dispatch phase, if the ECT_{ij} of server's hub j is not precisely MSV_i, the highest-SV job I

might be sent to the contrasting server hub j . Mission I with the most noteworthy ECT_{ij} esteem, then again, can be shipped off server hub j . Huge undertakings, then again, are essentially shipped off low-ability server hubs in heterogeneous settings utilizing this methodology.

The advanced MaxSufferance (AMS) computation [19] is presented to accommodate the issue above by further developing the MaxSufferage calculation's inadequacy. Notwithstanding, autonomous of the bunch or sort of administration, the AMS just considers mission booking of administration hubs. Accordingly, a steady methodology is proposed for at the same time tackling the booking of administration types, bunches, and administration hubs. Moreover, regardless of whether the server hubs are in a different climate, all positions can be conveyed to the fitting server hubs in the distributed computing organization.

3. RESEARCH METHOD

An ordinary bunch design delivery calculation doesn't dispatch errands in light of group limit, which might bring about Mission idleness, low dependability, and a long makes container. Subsequently, as delineated in Figure 1, a TLCD design as well as comparing booking calculations will be introduced for use in cluster storage configurations. All documentation and their clarifications as utilized in the calculation are coordinated in Table 1 preceding giving the subtleties of the proposed convention.

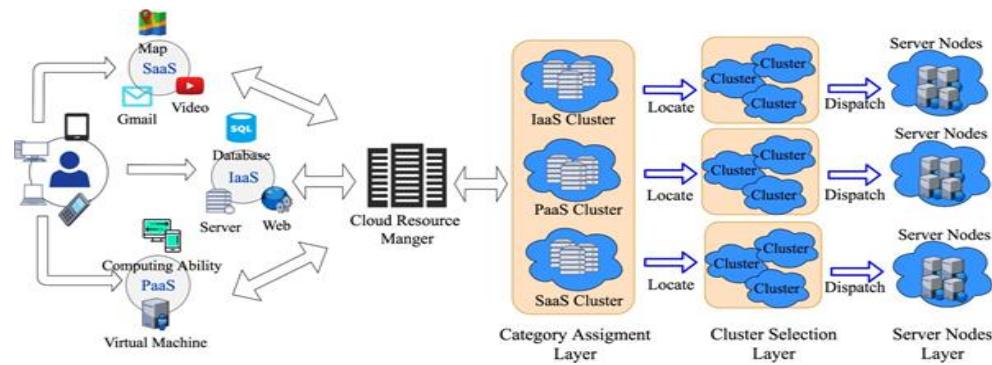


Figure 1. TLCD topology

Table 1. Transcription explanations in the clustered selecting phase

Phonetic symbols	Summary
N	Overall number of obligations
L	Maximum amount of clusters
K	Clusters
n_k	The sum of all operations in the k -cluster
T_r	The threshold of reliability
T_c	The cutoff point of the total price
A_i	Work order: i
R_k	Weighing the cluster's dependability k
M_k	Dimensions of the cluster's lifetime k

Working on the exactness of cloud administration search in the distributed computing climate is troublesome [14]. Accordingly, the initial phase in the recommended convention is to characterize cloud administration classifications to improve by and large execution. As indicated by Shojafar *et al.* [13], the framework will pick a cloud administration mission as the main bunch's center work indiscriminately. The likenesses between this haphazardly picked cloud administration mission and other cloud administration undertakings will be determined after that. All cloud administration occupations with likeness scores more noteworthy than or equivalent to the edges have specified will be incorporated into the main set. After that, the list of cloud locations containing these coordinates would be removed. administration mission competitors. As of now, the framework will construct a subsequent cloud group by haphazardly choosing another center occupation from the excess cloud administration undertakings. To add comparative cloud administration exercises to this group, the framework will utilize a comparative choice cycle. The determination method will go on until all cloud administration occupations have been assembled together. In the wake of distinguishing and classifying the undertakings, the proposed TLCD engineering will continually execute every ordered assignment. Coming up next are the particulars of TLCD.

3.1. Clustering for allocating class layer

Conventional bunch engineering utilizes cloud asset chiefs to gather and appropriate responsibilities to the group. In any case, group heterogeneity might change the portion cycle, making the assignment unessential regarding planning. This is because of cloud assets the executives apportion the jobs to an inactive group. Thus, the booking result is not great. The distributed computing framework will turn out to be more modern subsequently. Besides that, the assortment of obligations adds to the handling time delay. The heterogeneous errand can be ordered into various classes in light of interest characterized in the CAC layer [7], [8] to diminish the deferral and intricacy of planning. There are essentially three different ways to categorize groupings of clouds: IaaS, SaaS, and PaaS. The intricacy of booking heterogeneous positions and planning deferrals can be diminished by utilizing these three characterization gatherings.

3.2. Cluster selection layer

The arranged undertakings can be dispatched to the comparing classification group after the classification mission bunch layer has been ordered. Following that, making use of dependability's (D_i) core components, price (P_i), and MarketSpan (M_i), a cluster selection algorithm (CSA) is created towards relegate responsibilities as suitable bunches (3). The timeframe it takes to follow through with a job is called M_i . As a general rule, when the M_i esteem expands, the framework will call for greater investment to work. The figuring power will develop as the reliability expands because of the comparability of group adaptation to internal failure. Thus, we should focus on the computational ability of bunches [20]. At last, the expense factor is characterized as the expense of sending and answering a mission. Undertakings can be relegated to proper groups and framework proficiency can be improved when these three models are thought of. Moreover, clients and specialist co-ops can fit those three elements to their particular necessities. With the guidance of the following models, they D_i and P_i and allot the positions to the suitable bunches in light of the above portrayal. Mission I's trustworthiness and cost are shown by (1)-(3).

$$D_i = \frac{\sum_{k=1}^N CR_k n_k}{N} \quad (1)$$

$$P_i = \sum_{k=1}^N C_k n_k \quad (2)$$

$$M_i = \frac{\sum_{k=1}^N CM_k n_k}{N} \quad (3)$$

From that point forward, we'll give a guide to exhibit Algorithm 1. We coordinate the blend of occupations in all bunches in line (4) of Algorithm 1. The bunch will choose a reasonable errand mix and help the hub with mission changes. Lines (5) to (8) are likewise proposed for checking whether the D_i and P_i of every mission I concur with $D_i Tr$ and $P_i Tc$. After that comes the mission involving the youngest M_i is reserved out of every one of them. In the event that multiple gatherings are accessible, we look at D_i , M_i , and pick A_i as the most dependable group in the shortest amount of time.

Clients can change the nature of administrations in CSA in view of boundaries like dependability, evaluating, and M_i . Subsequently, calculations can take care of the requirements of a wide scope of customers while likewise expanding the proficiency of work planning. The CSA calculation is then made sense of with a model, and the connected presumptions are recorded.

Algorithm 1. Selection of clustering algorithm

```

1: for number of amount  $N$ 
2:   for number of cluster  $L$ 
3:     give  $R_k$ ,  $M_k$ , and  $C_k$  of each cluster  $k$ 
4:     Cluster all jobs, and give each one a number of  $A_i$ 
5:     for finding up  $R_i$ ,  $C_i$ , and  $M_i$  concerning the mission  $A_i$ 
6:       if  $R_i \geq Tr$  and  $C_i < Tc$  in mission  $A_i$  then  $A_i$  is member mission
7:     End
8:     select the youngest  $M_i$  in member mission  $A_i$ 
9:   End
10: End
11. End

```

For this example, let's say we have 10 jobs that need to be distributed among three different cloud service providers;

1. The values for T_r and T_c may be modified by the user. Here, we see that T_r as well as T_c are equal to 22 and 260, accordingly.

2. Find the R_i , C_i , and M_i for each possible allocation. In accordance with the breakdown shown in Table 2, five jobs are placed in cluster 1, two in cluster 2, and three in cluster 3. Here, we averaged R_i , C_i , and M_i with the help of (1)-(3). The same method is used for all of the assignments.
3. In this case, we choose schedules A1, A2, and A6 since they satisfy the constraints $R_i \leq 22$ and $C_i \leq 260$.
4. Considering the criteria of step 3, they choose among A1 because to its high degree of dependability and low M_i . Because assignment A1 yields a superior outcome, it has been chosen as the active combination for this case.
5. The aforementioned method is ideal whenever the M_i is really the driving force.
6. M_i and cost becomes disguising considerations when reliability is the major concern, revealing the best-reliability schedule. After the cloud service abstraction (CSA) layer is finished, the jobs could be transferred to the model's clusters clustering layer. Next, on the underlying layer, the appropriate server nodes must be selected to carry out the mission.

3.3. Layer for selecting server nodes

Homogeneous positions can be allotted to homogeneous cloud bunches after the initial two levels are finished. Notwithstanding, when countless positions fall flat attributable to mistaken mission, the cloud responsibility can become lopsided. In view of the previous, this work recommended the advanced cluster suffrage scheduling (ACSS), an original methodology that tends to the downsides of the maximum Suffrage calculation [12], [13], particularly in a location with a lot of different types of weather. To restrict the effect of mistaken mission, each undertaking in ACSS can indeed be assigned to the appropriate nodes in the data center by S_j , not entirely settled from the typical ECT of the server hub. Table 4 and Algorithm 2 show the terminology and points of interest of the ACSS calculation.

Typically, there are three sections to the ACSS algorithm. The SV_j calculation stage, the $EECT_i$ (earliest anticipated timing of completion) and $SEECT_i$ predicted arrival time after the soonest possible arrival time, but before the S_j are first determined sections (8) and (9) of the ACSS technique provide a full explanation of how to determine the SV value. In the second phase, known as the MSV_i calculation phase, whenever work I has the greatest SV value across all missions, the MSV number would be adjusted in mission I , to the very first ECT value that comes after.

Assignment since $EECT_i$ is superior to MSV_i and ECT_{ij} , and superior to $AECT_j$ of S_j , in the final step, job sending, I will be transferred to S_j . In cases when ECT_i is close to $AECT_j$ and ECT_i should be bigger than $AECT_j$, allocation I may be passed to client node j , where its ECT_i is smaller than $AECT_j$. As a result, the fundamental idea is distinct from earlier methods; the specifics of the technique are presented in sections 11 and 12 of Algorithm 2.

Algorithm 2. Scheduling for cluster suffrage with advanced technology

```

1: for all unallocated duties  $i$ 
2:   for all network elements  $S_j$ 
3:      $TCT_{ij} = ECT_{ij} + r_j$ 
4:     do schedule every Mission.
5:       make every network node unallocated;
6:       for every Mission  $i$  in  $S_j$ ;
7:         locate host nodes  $S_j$  this results in the quickest delivery time;
8:         determine out the Suited to the Mission Index. (  $SV = EECT_i - SEECT_i$  );
9:         If the highest priority of  $SV_i$  pick the Mission if there are several or more that
           are the same  $SEECT_i$  to  $MSV$  with greatest;
           Else, the Mission will have the most  $SEECT_i$  is comparable to certain other  $EECT_i$  ;
10:        end if;
11:        If (  $MSV_i < ECT_{ij}$  of  $S_j$  ) therefore, the job  $i$  with greatest  $ECT$  able to communicate
           with host node  $S_j$ ;
12:        else if (  $MSV_i > ECT_{ij}$  ) && (  $EECT_i > AECT_j$  of  $S_j$  ) therefore, the job  $i$  able to
           communicate with host node  $S_j$  ;
13:        else if (  $MSV_i > ECT_{ij}$  ) && (  $EECT_i < AECT_j$  of  $S_j$  ) subsequent Mission allocation  $i >$ 
            $AECT_j\_AVG$  of  $S_j$  &&  $task\ i \approx AECT_j$  of  $S_j$  able to be sent to host node  $S_j$  ;
14:        end if;
15:      end for
16:    end do
17:  end for
18:   $r_j = r_j + ECT_{ij}$ 
19:  update  $TCT_{ij} = ECT_{ij} + r_j$ 
20: end for
21: End

```

The errand culmination time and load adjusting can be effectively brought down among the diverse networking for cloud applications utilizing the methodology portrayed previously. The asymmetric cryptographic method approach may be used to alleviate the security concern [21], [22]. It is related to a crucial session to ensure that perhaps the message is delivered veritable and has not been altered. Following that, a model is given to help the perception out of host hubs identification gradient ACSS computation.

4. RESULTS AND DISCUSSION

In this section, it is explained the results of research and at the same time is given the comprehensive discussion. Hub for highly diverse servers (HiHi), low-latency ring (LoLo), high-latency ring (HiHi), and LoLo [18], [23] are four heterogeneous conditions that can be concentrated on around here. HiHi is, by a long shot, the most convoluted of the multitude of territories. Accordingly, in the HiHi heterogeneous climate, an illustration of undertaking mission with four server hubs and twelve errands is talked about. First, the SVj estimate generation and the MSVi calculation stage may resolve both SV renown and MSV worth independently. Following that, during the undertaking dispatching stage, the AECTj of not set in stone.

4.1. The SVi calculation phase

Stage 1: as demonstrated in Table 2, a list of all expected running time undertakings I on Sj.

Stage 2: for each errand, ascertain the SV esteem. For example, Job a’s SV value is comparable to SEECTi less EECTi, which is 23526 - 22345 = 1181, similarly comparable periods are used to decide the SV values for undertakings b through l. The calculated outcomes are shown in Table 3.

4.2. Stage of MSVi computation

Stage 1. meanwhile work I have most noteworthy SVi worth of all SV values, the MSV worth can be picked as the next-oldest ECT worth of errand I. Job g have the most elevated SVi esteem, as expressed in Table 3, since 44746 is the next earliest ECT in ECTaB, it is used as the MSV.

Table 2. The problem of calculating the SV levels

	Node A	Node A	SV
Mission a	22345	23500	1181
Mission b	16667	17901	1263
Mission c	31083	31825	814
Mission d	24712	25123	444
Mission e	17018	18008	1051
Mission f	12050	13218	1239
Mission g	19035	21476	2451
Mission h	13911	14678	691
Mission i	80160	85490	520
Mission j	13618	14517	978
Mission k	27861	28149	295
Mission l	40000	44746	1395

Table 3. Establishing what the MSV levels should be requires some calculation

	Node A	Node B	SV	MSV
Mission a	22345	23526	1181	-
Mission b	16667	17930	1263	-
Mission c	31083	31897	814	-
Mission d	24712	25156	444	-
Mission e	17018	18069	1051	-
Mission f	12050	13289	1239	-
Mission g	19035	21486	2451	-
Mission h	13900	13502	8910	-
Mission i	70162	95368	558	-
Mission j	17818	18566	998	-
Mission k	27181	27896	695	-
Mission l	43226	44561	2095	-

4.3. Job distribution stage

By and large, three circumstances in different heterogeneous settings should be analyzed, and thus the instances are described and presented in the following order.

4.3.1. Case 1. EECTi > AECTj of Sj and MSVi > ECTij of Sj

In the MSVi computation step, comparing the MSV value to the earliest anticipated finishing time of other jobs. While MSVi is running, Mission I may be sent to any established norms node j.

$> EECT_i > AECT_i$ of S , and ECT_{ij} of $S_j > EECT_i$. Since Node D has the highest MSV_i and lowest ECT_{dD} in Tables 4 and 5, respectively,

Table 4. Activities' MSV results are calculated

	Node C	Node D	SV	MSV
Mission a	52978	25328	27650	-
Mission b	47351	20187	27164	-
Mission d	54429	26330	28099	54429
Mission e	47319	20348	26971	-
Mission f	43566	15918	27648	-
Mission g	51026	24031	26995	-
Mission h	44310	17678	26632	-
Mission j	37603	9685	27918	-
Mission l	43986	16558	27428	-
Average	-	19556	-	-

Table 5. Analysis of the node D jobs' median ECT levels for instance 1

	Node C	Node D	SV	MSV
Mission a	52978	25328	27650	-
Mission b	47351	20187	27164	-
Mission d	54429	26330	28099	54429
Mission e	47319	20348	26971	-
Mission f	43566	15918	27648	-
Mission g	51026	24031	26995	-
Mission h	44310	17678	26632	-
Mission j	37603	9685	27918	-
Mission l	43986	16558	27428	-
Average	-	19556	-	-

4.3.2. Case 2. $MSV_i > ECT_{ij}$ of S_j

You may see how early other missions are likely to be finished by comparing the MSV value determined during the MSV_i calculation step. If $MSV_i > ECT_{ij}$, then mission I with the highest ECT may be sent to server node j . Since node A 's $EECT_A$ is higher than that of the MSV value in Table 6 ($43336 > 21486$), A is given the assignment of mission l .

Table 6. Node A's median ECT in case 2

	Node A	Node B	SV	MSV
Mission a	22345	23526	1181	-
Mission b	16667	17930	1263	-
Mission c	31083	31897	814	-
Mission d	24712	25156	444	-
Mission e	17018	18069	1051	-
Mission f	12050	13289	1239	-
Mission g	19035	21486	2451	21486
Mission h	13911	14602	691	-
Mission i	8016	8536	520	-
Mission j	13618	14596	978	-
Mission k	27861	28156	295	-
Mission l	43336	44731	1395	-

4.3.3. Case 3. When comparing ECT_{ij} to MSV_i , the latter is superior

As part of the MSV_i calculation, the MSV value is compared to the earliest estimated completion time of other tasks. For example, in Table 7, mission i is assigned to client node j if $MSV_i > ECT_{ij}$ and $EECT_i AECT_j$ of S_j , assuming that ECT_i is more than or equal to $AECT_j$. Thus, we can reproduce the above situations in our trial. To adjust to the high-heterogeneity climate, the quantity of positions is first changed in accordance with 50 to 100, while the 5 online servers' hubs' combined processing power is limited to 500 to 1,000 units. Besides that, the Sufferage, MaxSufferage, and ACSS calculations are thought about as far as MakeSpan and load adjusting multiple as shown in Figures 2 to 5, and the usual value of computation time is then taken. Figure 2 illustrates that the suggested procedure beats the others concerning MakeSpan, particularly while managing huge jobs. The $rmin/rmax$, where $rmin$ is the briefest gotten done with job season $rmax$ is the shortest completed work season of all missions, and of all missions. Maheswaran *et al.* [23], can likewise be utilized to decide the heap adjusting file.

Table 7. Comparing average ECT values for node D across case 3 scenarios

	Node A	Node B	SV	MSV
Mission b	60023	49827	10196	-
Mission e	60374	49966	10408	-
Mission f	55406	45186	10220	-
Mission h	57267	46499	10768	-
Mission i	51372	40433	10939	51372
Mission j	56974	46493	10481	-
Average	-	46400	-	-

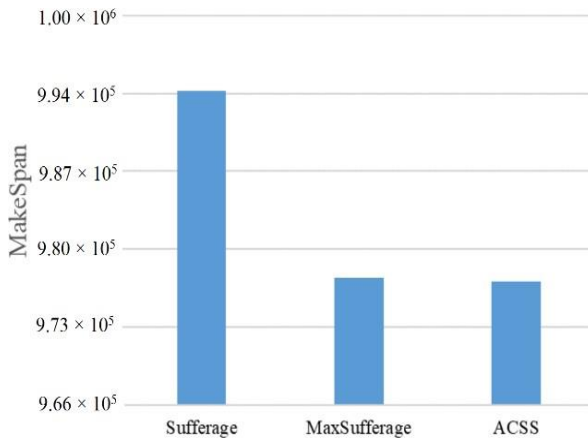


Figure 2. ACSS, Sufferage, and MaxSuffering in MakeSpan with n=4 and 100 missions

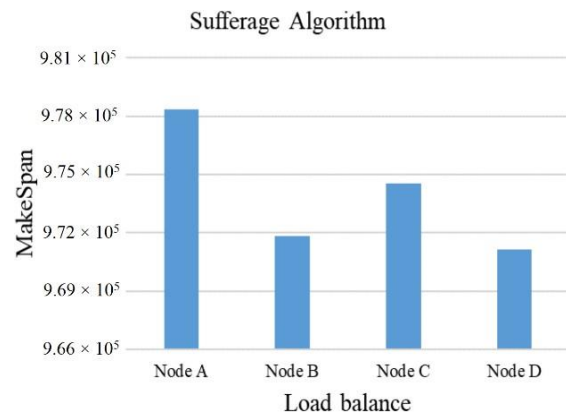


Figure 3. Load balancing index of Sufferage for n=4 and 100 missions

As a general rule, the heap adjusting file has a worth of 0 to 1, with 0 addressing the most unfortunate burden equilibrium and 1 addressing the best burden balance. The ACSS strategy can accomplish the best burden adjusting list (0.88) over Sufferage (0.87) and MaxSufferage (0.88), as outlined in Figures 3. The MakeSpan of every hub can yield tantamount outcomes in light of the fact that the ACSS calculation utilizes the scattering of the typical worth. Be that as it may, while MaxSufferage has a quicker finish time than Sufferage, the heap adjusting results are comparable. This is because of the way that while choosing position, MaxSufferage didn't consider the hub's heap condition. Proposed ACSS algorithm optimizes heterogeneous distributed computing network for efficient time and load balancing, yielding superior outcomes.

Besides this, the formula $RU = \frac{j-1}{N} \times \text{test}$ if the paper's consumption of resources is ideal, the ratio of resourceNm is calculated as 100%. The TCj in factor RU represents the all-out expected consummation time on a computer simulation j, N for the total number of VMs, and m for such number of cores virtual machine's last fulfillment time. Figure 4 portrays the asset use related proportion results. Figure 5 shows that ACCS might accomplish an asset use proportion of 89%, which is higher than different calculations. This is on the grounds that in the ACSS calculation, the typical worth is used to think about the assignment status of hubs.

Next, we utilize the matching vicinity boundary [24] to evaluate the proximity of different booking procedures. In order to establish if the mission can be finished quickly, Figure 6 makes use of the minimum execution time (MET) as well as the expected computing time (ECT). Numerous Missions are sent to the most powerful computer because of the great value placed on the proximity between the two systems.

The matching proportion of the three calculations is near one, as displayed in Figure 6. These three calculations are very great at coordinating. Table 8 looks at the presentation and intricacy of calculations. As far as assessment factors, for example, MakeSpan, load balance, asset utilization, and matching vicinity, the discoveries of the correlation table uncover that ACSS beats any remaining calculations.

The Big O documentation is utilized to gauge the intricacy of different calculations in view of [25]. The intricacy of MaxSufferage and ACSS is accordingly $O(n^2rm)$, in light of the fact that boundary r shows that the elective condition is picked when $(MSVi > ECTij)$ and $(EECTi AECTij \text{ of } S_j)$ are fulfilled. Thus, the ACC calculation's intricacy is practically identical comparison to the suffering tally.

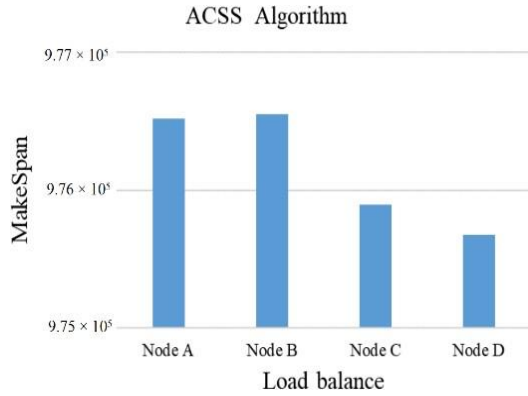


Figure 4. Load distribution indexes in advanced HA cluster for n=4 and 100 missions

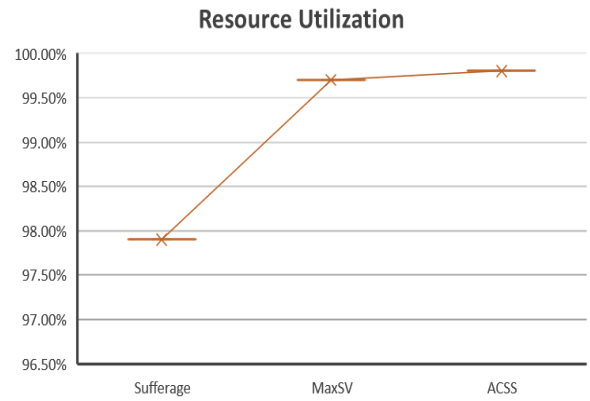


Figure 5. ACSS resource utilization ratio for N=4 and 100 missions

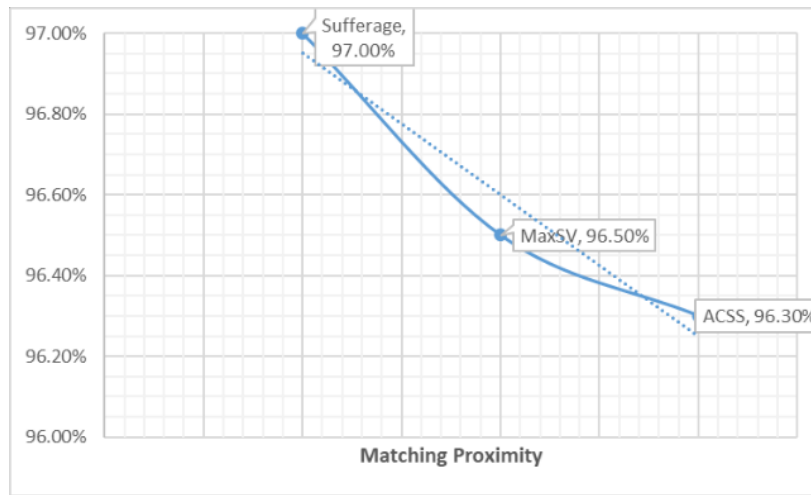


Figure 6. Comparing all rescheduling methods based on their matching distance ratio

Table 8. All algorithms' performance and complexity are compared

	SufferMax	SufferMin	ACSS
Make span	8.57×10 ⁴	7.97×10 ⁴	8.96×10 ⁴
Balance of the loads	0.87	0.84	0.88
Utilization of available resources	91%	95.5%	96%
Like-within-close-range matching	0.98	0.97	0.96
Ambiguity	O(n2m)	O(n2r)	O(n2rm)





5. CONCLUSION

The TLCD design is proposed in this work to give secure and reliable booking as well as to resolve the issue of slow reaction in cloud frameworks. TLCD is comprised of three layers of techniques. The framework can dispatch heterogeneous positions into proper class bunches in the primary layer, known as the CAC layer, to decrease mission delay and over-burdening. The work is then dispatched to a suitable bunch utilizing a CSA calculation on the CS layer, which further develops dependability and decreases cost and finishing time. The framework can further develop load adjusting and decrease fruition length of time spent within the final layer, symbolically represented as such SNS layer, by utilizing components of MSV and the typical ECT of S_j. The recommended calculations beat any remaining calculations as far as MakeSpan, load adjusting, asset usage, and matching vicinity in different settings, as per reproduction information. Future directions include assessing TLCD scalability, exploring performance factors, extending TLCD applications, and evaluating scheduling algorithms to enhance TLCD's effectiveness and applicability in diverse computing environments.





REFERENCES

- [1] V. V. Vegesna, "A critical investigation and analysis of strategic techniques before approving cloud computing service frameworks," *International Journal of Management, Technology*, vol. XIII, no. Iv, pp. 132–144, 2023.
- [2] V. Yannibelli *et al.*, "An in-depth benchmarking of evolutionary and swarm intelligence algorithms for autoscaling parameter sweep applications on public clouds," *Scientific Programming*, vol. 2023, pp. 1–26, Feb. 2023, doi: 10.1155/2023/8345646.
- [3] N. Ghazy, A. Abdelkader, M. S. Zaki, and K. A. Eldahshan, "An ameliorated round robin algorithm in the cloud computing for task scheduling," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 12, no. 2, pp. 1103–1114, Apr. 2023, doi: 10.11591/eei.v12i2.4524.
- [4] K. Prabu and P. Sudhakar, "A hybrid deep learning approach for enhanced network intrusion detection," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 33, no. 3, pp. 1915–1923, Mar. 2024, doi: 10.11591/ijeecs.v33.i3.pp1915-1923.
- [5] Harintaka and C. Wijaya, "Automatic point cloud segmentation using RANSAC and DBSCAN algorithm for indoor model," *Telkonnika (Telecommunication Computing Electronics and Control)*, vol. 21, no. 6, pp. 1317–1325, Dec. 2023, doi: 10.12928/TELKOMNIKA.V21I6.25299.
- [6] K. Prabu and P. Sudhakar, "An automated intrusion detection and prevention model for enhanced network security and threat assessment," *International Journal of Computer Networks and Applications*, vol. 10, no. 4, pp. 621–636, Aug. 2023, doi: 10.22247/ijcna/2023/223316.
- [7] P. Neelakantan and N. S. Yadav, "An optimized load balancing strategy for an enhancement of cloud computing environment," *Wireless Personal Communications*, vol. 131, no. 3, pp. 1745–1765, 2023, doi: 10.1007/s11277-023-10520-2.
- [8] S. I. Watson, A. Girling, and K. Hemming, "Optimal study designs for cluster randomised trials: an overview of methods and results," *Statistical Methods in Medical Research*, vol. 32, no. 11, pp. 2135–2157, Nov. 2023, doi: 10.1177/09622802231202379.
- [9] S. R. Bharamagoudar and S. V. Saboji, "Location-aware hybrid microscopic routing scheme for mobile opportunistic network," *IAES International Journal of Artificial Intelligence*, vol. 12, no. 2, pp. 785–793, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp785-793.
- [10] S. Gangadharaiyah and P. Shrinivasacharya, "Effective privacy preserving in cloud computing using position aware Merkle tree model," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 13, no. 2, pp. 1424–1432, Apr. 2024, doi: 10.11591/eei.v13i2.6636.
- [11] K. Senjab, S. Abbas, N. Ahmed, and A. ur R. Khan, "A survey of Kubernetes scheduling algorithms," *Journal of Cloud Computing*, vol. 12, no. 1, p. 87, Jun. 2023, doi: 10.1186/s13677-023-00471-1.
- [12] M. Shojafar, N. Cordeschi, and E. Baccarelli, "Energy-efficient adaptive resource management for real-time vehicular cloud services," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 196–209, Jan. 2019, doi: 10.1109/TCC.2016.2551747.
- [13] M. Shojafar, C. Canali, R. Lancellotti, and J. Abawajy, "Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1162–1175, Oct. 2020, doi: 10.1109/TCC.2016.2617367.
- [14] S. Shivle *et al.*, "Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment," *Journal of Parallel and Distributed Computing*, vol. 66, no. 4, pp. 600–611, Apr. 2006, doi: 10.1016/j.jpdc.2005.10.005.
- [15] K. Etmnani and M. Naghibzadeh, "A min-min max-min selective algorithm for grid task scheduling," *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pp. 1–7, 2007, [Online]. Available: <https://api.semanticscholar.org/CorpusID:17785038>
- [16] T. D. Braun *et al.*, "A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems," in *Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99)*, 1999, pp. 15–29, doi: 10.1109/HCW.1999.765093.
- [17] N. M. Reda, A. Tawfik, M. A. Marzok, and S. M. Khamis, "Sort-Mid tasks scheduling algorithm in grid computing," *Journal of Advanced Research*, vol. 6, no. 6, pp. 987–993, Nov. 2014, doi: 10.1016/j.jare.2014.11.010.
- [18] A. M. Abdulghani, "Task scheduling for multi-objective optimization in cloud computing: a review," *Available at SSRN 4610623*, [Online]. Available: <https://ssrn.com/abstract=4610623>.
- [19] M. L. Chiang, H. C. Hsieh, W. C. Tsai, and M. C. Ke, "An improved task scheduling and load balancing algorithm under the heterogeneous cloud computing network," in *Proceedings - 2017 IEEE 8th International Conference on Awareness Science and Technology, iCAST 2017*, Nov. 2017, vol. 2018-January, pp. 290–295, doi: 10.1109/ICAwST.2017.8256465.
- [20] M. L. Chiang, Y. F. Huang, H. C. Hsieh, and W. C. Tsai, "Highly reliable and efficient three-layer cloud dispatching architecture in the heterogeneous cloud computing environment," *Applied Sciences (Switzerland)*, vol. 8, no. 8, p. 1385, Aug. 2018, doi: 10.3390/app8081385.
- [21] C. Huang, D. Liu, A. Yang, R. Lu, and X. Shen, "Multi-client secure and efficient DPF-based keyword search for cloud storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 1, pp. 353–371, Jan. 2024, doi: 10.1109/TDSC.2023.3253786.
- [22] K. Prabu and P. Sudhakar, "Design and implementation of an automated control system for anomaly detection using an enhanced intrusion detection system," in *Proceedings of the 3rd International Conference on Smart Technologies in Computing, Electrical and Electronics, ICSTCEE 2022*, Dec. 2022, pp. 1–7, doi: 10.1109/ICSTCEE56972.2022.10100003.
- [23] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 107–131, Nov. 1999, doi: 10.1006/jpdc.1999.1581.
- [24] P. Banerjee *et al.*, "MTD-DHJS: makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction," *IEEE Access*, vol. 11, pp. 105578–105618, 2023, doi: 10.1109/ACCESS.2023.3318553.
- [25] Z. Cui, T. Zhao, L. Wu, A. K. Qin, and J. Li, "Multi-objective cloud task scheduling optimization based on evolutionary multi-factor algorithm," *IEEE Transactions on Cloud Computing*, vol. 11, no. 4, pp. 3685–3699, Oct. 2023, doi: 10.1109/TCC.2023.3315014.





BIOGRAPHIES OF AUTHORS

Vijaya Kumar S     research scholar in Shri Venkateshwara University. His area of interests are cryptography and network security, cloud computing, big data, and machine learning. He is having a strong experience in SQL. He completed ME in Computer science from Anna University. He can be contacted at email: vksudev@gmail.com.







Dr. Muthusamy Periyasamy     professor, Department of Computer Science and Engineering, Shri Venkateshwara University, has 20 years of teaching experience. he holds a Ph.D. from Anna University. He has published 17 patents, 8 book chapters, and 30 research papers published in reputable international journals and conferences. His expertise includes cloud computing, cyber security, artificial intelligence, and machine learning. He can be contacted at email: muthu.namakkal@gmail.com.







R. Radhakrishnan     assistant professor in School of computer science and engineering at Galgotias University, holds 16 years of teaching experience With an MTech from Anna University, he has published 4 patents and 8 research papers, specializing in networks, cloud computing, and machine learning. He can be contacted at email: prof.rk8@gmail.com.



Dr. Tamilarasi Karuppiah     is an Associate professor in the Department of Information Technology, Panimalar Engineering College, accumulating 24 years of teaching experience. She earned his Ph.D. record with 8 patents, 9 book chapters, and 28 research papers published in esteemed international journals and conferences. Her expertise spans web services, cyber security, networks, cloud computing, and machine learning. She can be contacted at email: thamizhanna@gmail.com.



Dr. Thenmozhi Elumalai     is an associate professor in the Department of Information Technology at Panimalar Engineering College. With 22 years of teaching experience, she holds a Ph.D. and has authored 7 patents, 8 book chapters, and 18 research papers in renowned international journals and conferences. Her areas of expertise include cyber security, networks, and machine learning. She can be contacted at email: ethenmozhi22.pec@gmail.com.