TASK SCHEDULING AND RESOURCE ALLOCATION IN CLOUD COMPUTING USING A HEURISTIC APPROACH

S. Dharmaraj1 **Dr. P. Kavitha2**

1 Department of computer application, Loyola college, vettvalam, Thiruvannamalai, Tamilnadu, India

2 Department of computer science,Muthayammal College of Arts and Science, Namakkal , Tamilnadu, India

Abstract

Cloud computing is required by modern technology. Task scheduling and resource allocation are important aspects of cloud computing. This paper proposes a heuristic approach that combines the modified analytic hierarchy process (MAHP), bandwidth-aware divisible scheduling (BATS) + BAR optimization, and longest expected processing time preemption (LEPT) each task is processed before its actual allocation to cloud resource using a MAHP process. The resources are allocated using the combined BATS + BAR optimization method, which considers the bandwidth and load of the cloud resource as constraints. In addition, the proposed system preempts resource-intensive tasks using LEPT preemption. The divide- and-conquer approach improves the proposed system, as is proven experimentally through comparison with the existing BATS and improved differential evolution algorithm (IDFA) framework when turnaround time and response time are used as performance metrics.

Keywords: *Cloud computing, Task scheduling, Heuristic, Resource management, analytic hierarchy system, BATS, BAR.*

Introduction

Cloud computing is an accelerating technology in the field of distributed computing. Cloud computing can be used in applications that include storing data, data analytics and IoT applications. In each types service, the users are expected to submit the requests to the service provider through the medium of the managing the resource to fulfill the requests generated by users. Service providers employ scheduling algorithms to scheduling the incoming request (task) and to manage their computing resource efficiently. In practice, in terms of the performance of cloud computing resource are important hurdles. For this reason, researchers have been attracted to studies of task scheduling in cloud computing. Task scheduling is the process of arranging incoming request (task) in a certain manner so that the available resources will be properly utilized. Because cloud computing is the technology that delivers services through the medium of the internet, services users must submit their requests online. Because each services has a number of users, a number of request (tasks) may be generated at a time. At the time of scheduling, the scheduler needs to consider a number of constraints, including the nature of the task, the size of the task, the task execution time, the availability of resources, the task queue, and the load on the resource. Task scheduling is one of the core issues in cloud computing. Proper task scheduling may result in the efficient utilization of resources. The major advantage of cloud computing is that it promotes proper utilization of resources, thus task scheduling and resource allocation are two sides of a single coin. Cloud computing enhances the capabilities of such infrastructure, which can access the internet. Cloud services providers earn profits by providing services to cloud services users. The cloud service end user can use entire stack of computing services, which ranges from hardware to applications. The cloud services user can rent the resource at any time and release them with no difficulty. The cloud service user has the freedom to employ any service based on application need. The freedom of service choice for users has led to problems that is the next user request cannot be perfectly predicted. Thus task scheduling and resource allocation are mandatory parts of cloud computing research. The efficiency of resource uses depends on the scheduling and load balancing methodologies, rather than the random allocation of resource. Cloud computing is widely used for solving complex tasks (user requests). In solving complex task issues, the use of scheduling algorithms is recommended. Such scheduling algorithms leverage the resources. The propose system employ feature of the Cybershakescientific work flow and the Epigenomics scientific workflow, which are described in section input data.

The major contribution of this paper are summarized as follows.

1. The analytic hierarchy process is modified to rank scientific tasks.
2. To manage the resource given bandwidth constraints and the load on the virtual machine, the proposed system incorporates a version of the existing BATS algorithm that has been modified by introducing BAR system optimization.
3. Bipartite graphs are utilized to map tasks to appropriate virtual machines once the condition is satisfied.
4. A preemption methodology gives us the status of the virtual machine, and a modified divide-and conquer methodology has been proposed to aggregate the results after tasks preemption.
5. The proposed solution is experimentally investigated using the cloud Sim simulator.

Cloud computing and its outstanding issues, especially task scheduling and resource allocation. Section related work focused on related studies that investigate task scheduling and resource allocation. Section “input data” describes the input data provided to the Cybershake scientific workflow. Section “evaluation of the proposed heuristic approach” focused on evaluating the proposed heuristic approach section “results and describe the results and existing BATS and IDEA algorithms.

Input Data:

Cybershake Scientific workflow-

Cloud computing is the service provider paradigm in which users submit requests for execution. Thus, the responsibility of the cloud service provider is to scheduling various requests and manage resource efficiently. However the actual procedure of scheduling tasks and resource management begins with how the service provider addresses incoming tasks. The proposed system useCybershake scientific workflow data as input tasks . Fig. 1 show a visualization of the Cybershake scientific workflow, which is used by the Southern California Earthquake Center (SCEC) to characterize earthquake hazards using the probabilistic Seismic Hazard Analysis (PSHA) technique. It also generates Green strain tensors (GSTs). Table 1 shows the Cybershake seismogram synthesis tasks with their sizes and execution times.The Cybershake is a collection of Cybershake scientific workflow sample tasks are available with task size 30, 50,100 and 1000. The Cybershake spends a lot of time on seismogram synthesis during its execution. These types of tasks also require large amount of computational resources, such as CPU time, and memory.

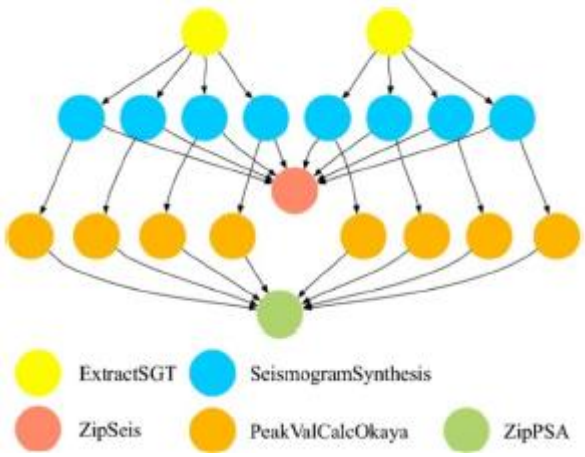

Fig 1. Cybershake scientific workflow

Cybershake scientific workflow has been divided into steps.

1. Extract GST- this step of the workflow extracts the GST (Green strain tensor) data for processing.
2. Seismogram synthesis – These tasks are the most computationally intensive. Most of the time spent in running the Cybershake algorithm is employed on this step.
3. ZipSeis – This step aggregates is processed data.

Table 1 Cybershake seismogram synthesis tasks.

Tasks	Size of tasks	Time
Task 3	62,69,51,663	39.06
Task 5	69,47,76,323	38.49
Task 7	58,57,63,637	36.27
Task 9	53,68,97,326	32.29
Task 11	67,05,35,542	62.25
Task 14	40,67,28,38,798	96.91
Task 16	45,23,96,996	45.6
Task 18	50,27,64,231	28.67
Task 20	62,41,88,532	24.56
Task 22	42,65,77,006	31.05
Task 24	51,58,32,878	54.87
Task 26	68,14,99,417	23.99
Task 28	44,14,51,516	26.46

4. PeakValCalcOkaya – the heights – strength values of each seismogram are calculated in this step.
5. ZipPSA – This step aggregates the processed data.

Proposed system

The architecture of the proposed system. In practice, various types and sizes of tasks arrive at the cloud data centers for execution. In general, scientific tasks represent collections of different types and sizes. To manage the tasks that come into a cloud data center, the proposed system uses the analytic hierarchy process (AHP). The primary aim of this proposed system is to manage incoming tasks. Therefore, the proposed system uses the AHP methodology to assign a rank to each tasks based on its length and run time as soon as the tasks are assigned individual rankings, they are collected and arranged into tasks queues. The tasks in the task queue are strictly arranged following the AHP ranking. Thus, the first stage of the proposed system is completed. Next, in the second stage, the proposed system also addresses the computing resource of cloud data centers, such as CPU, memory and bandwidth using the proposed BATS + BAR optimized allocation methodology. This methodology works as follows. It takes the task to be execution from the task queue. This stage is the second part of the procedure in which the allocations of resource have carried out using BATS + BAR optimizes algorithm. The allocation of research have been carried out this algorithm.

In another part of proposed technique is preemption methodology namely LEPT method are used, it continuously checks the load of the virtual machine, if the current status of the virtual machine is overloaded and other is idle. The proposed method divide and conquer methodology which breaks up the task and distribute other virtual machine. The proposed system is overcome the limitations of the allocation resource in the basis of CPU, memory and bandwidth.

METHODOLOGY

Analytic hierarchy process

The analytic hierarchy process is solving composite problems with various criteria. The proposed system uses this process in cloud computing environments to rank the inward tasks in a certain manner. The proposed system uses scientific.Workflow tasks, namely Cybershake for testing because such require long execution times. Initially, the workflow is divided into five stages, which are introduced in the input data section. Before proceeding with the proposed system, the AHP methodology is applied for the for cybershake flow. it is control flow dependent; thus, the second stage will execute only after the execution of the first stage.

To evaluate preferences, the proposed system uses the Saaty preference table, which is given in Table 2 with its numerical ratings. To promote understanding while accounting for space limitations, the proposed system divides each calculation table into two parts. The first part extends from Task 3 to Task 18, whereas the other part shows the calculations from Task 20 to Task 28. Here, the proposed system considers two significant criteria that are involved in scientific tasks; task length and taskrun time. The comparison numerical ratings are given in Table 2, which is known as the Saaty preference table. Before the actual calculation is begun, the proposed system assigns preference values to the tasks. Here, the preferences associated with the tasks are based on their lengths and the execution times of the different tasks. The proposed system slightly modifies the Saaty table preferences because, as tasks
with different ranks are on a server, the ranks of subsequent tasks change, and new rankings must be calculated. The proposed system calculates such rankings of task

BATS+ BAR system

The proposed system has two aspects, which involve scheduling tasks and managing resources. Here, we improve upon the BATS algorithm, which was originally proposed by Weiwei Lin [7]. Independent tasks of equal size are considered in the design of this system. However, in allocating resources, the system does not consider the load on virtual machines because the waiting period for the tasks is long. In other cases, one virtual machine is busy while it executes a task, whereas others are occupied and waiting for jobs. The bar systems (BSs) algorithm. The social behavior of bartenders is the basis of BS
systems. Swarm intelligence has added an optimization aspect to BS. In a bar, bartenders must act in a highly dynamic, asynchronous and time-critical environment,and no obvious greedy strategy (such as serving the best customer first, serving the nearest customer first or serving the first-arriving customer first) gives good results.Thus, multi-agent systems provide a good framework within which to address the challenge of developing a new class of adaptive and

robust systems. we propose modifying BATS by adding a BAR system. The procedure is as follows:

1. Aggregate all of the task information that is ordered by rank.
2. Virtual machine (server) information is collected. This information includes the initial load on the virtual machine, its bandwidth and the time required,to process the tasks on the server.
3. A bipartite graph is generated with the number of tasks. The ranking priorities, can be used to construct a graph, by which each task is allocated to a virtual machine.

The Load on the virtual machine(S) is calculated as,

$$L^{ini} = L_s^{ini} \; \text{l} \; s \subset S - (1)$$

The bandwidth is calculated as,

$$DB_W = bi <= bi - (2)$$

The total time taken to process the tasks is calculated as,

$$L_S^{fin}(\alpha) = L_s(\alpha) - (3)$$

Bipartite graph

A bipartite graph is produced based on the following conditions:

1. A bipartite graph is constructed as-, G = (Tn U S, E) in which ‘Tn’ represents the number of tasks, ‘S’ presents the servers, and ‘E ⊆ T X S’ that is, the set of edges that are present between the task and the server. An edge represents the tasks ‘Ti ⊂ Tn’, which are present on virtual machine ‘s ⊆ S. A graph is constructed using a bipartite graph with the number of tasks.

2. Balance the constructed graph with constraints including the local cost, the initial load and the bandwidth.

3. Based on the local cost and the initial load we compute the total load on the virtual machine.

$$L_s - L_s^{ini} + T_N(S_l).C_{bc}$$

If this condition is satisfied, then we allocate the tasks to that particular virtual machine. If this condition is not satisfied by that virtual machine, then we move on the next server and check this condition.

Divide-and-conquer methodology

After, the tasks have been preempted; we apply the divide-and-conquer methodology by following the steps shown below.

Evaluation of the proposed heuristic approach is simulated on a cloud computing environment {25} that provides a real-time

Algorithm :LEPT Preemption

Input: Tasks allotted to VMs

Output: Preempt task

```
Begin
For(int t=; t<size; t++) do
hasChecked.add(false);
end
for(int t=0l t<sizel t++) do
int minIndex = 0; Cloudlet minCloudlet =null;
if(!hasChecked.get(j)) then
miCludlet = cloudlet; minindex=j; break;
   end
end
   if(minCloudlet ==null) then
break;
end
for (int j=0; j<size; j++) do
        Cloudlet cloudlet = (Cloudlet) getCloudletList()..get(i);
 If (Lengt<minCloudlet.getCloudletLength())
Then
minCloudlet =cludlet; minIndex= j;
end
end
hasCheckded.set(minindex, true);
int VMSize= get VMLust().size; CondorVM firstidleVM =null;
(CondorVM) getVMList().get(0);
For(int j=0; j<VMSize;j++) do
condorVM VM= (CondorVM) getVMList().get(j);
if(VM.getState()WorkflowSimTags. VMstatsidle) then
break;

end
ene
if(firstidleVM==null) then
break;
end
            for(int j=0; j<VMSize; j++) do
```

```
CondorVM VM= (CondorVM) getVmList() get(j);
end
en
```

cloud computing scenario. The configuration details the date center used in the customized simulation setup are given in Table 10 and consist of general information on the data centers such as the number of processing units,

Algorithm : Divide-and-Conquer Algorithm

```
Input: free VNs, and bus VMs
Output: Task allotted to free Vms
Begin
      If(index==1) then
Return sum;
 Else if (index,=4 && index>1) then
 For (first< last) do
Sum + = A[i];
End
Else
Return sum;
Return (sumArray(first, last/2, A.Leagth, A));
```

And capacity. Every data center component generates a set of strategy for allocating bandwidth, memory and storage devices for hosts and virtual machines Table 11 shows the configuration for the data center, including its allocation policy, architecture, OS, hypervisor scheduling and monitoring interval, and threshold value, among other properties. The host in the data center is built with the configuration such as RAM, bandwidth, storage capacity, power, processing elements etc. of the given task, the processing of which by a data center is listed in Table 12. Table 13 provides the details of the customer configuration. Table 14 provides a detailed description of the virtual machines.

Results and discussion

`As a second performance metric, we consider the response time of the algorithm to incoming tasks. The response time is essentially the time during which the request is actually considered. In other works, the response time is directly dependent on the availability of tasks is performed properly, then the resources will naturally be free early or in advance of deadlines, the response times will be less in such cases.

By, comparing the response times obtained for our proposed heuristic approach with those obtained using the existing BATS and IDEA frameworks, we can see that our system's response time is almost 50% less. The response time comparisons for Cybershake is presented. To consider two parameters the response time and turnaround time compare the proposed heuristic approach with the existing BATS and IDEA frameworks. Because we are evaluating these frameworks in a cloud computing environment, the response time is general less effective. On the other hand, we also evaluated our proposed heuristic approach to determine its resource performance compare it to those of the existing BATS and IDEA frameworks. The proper utilization of resources produces profits for cloud computing service providers. The experimental results show that the proposed heuristic approach utilized the CPU resource more efficiently than the existing BATS framework. the second key comparison of resource utilization between the proposed heuristic approach and the experimental results shows that the proposed heuristic approach utilizes memory resources more efficiently that the existing BATS and IDEA frameworks. Bandwidth, and important resource, is not considered in most existing frameworks. We take bandwidth into account as a third important aspect of cloud computing data centers. We also compare our proposed heuristic approach with the existing BATS and IDEA frameworks. The proposed heuristic approach utilizes bandwidth more efficiently than the existing BATS and IDEA frameworks.

Conclusion

In this study, we proposed heuristic algorithm that performs task scheduling and allocates resources efficiently in cloud computing environments. We use real Cybershake scientific workflows as input tasks for the system. When we compare our proposed heuristic approach with the existing BATS and IDEA frameworks with respect to turnaround time and response time, we find that our approach gives improved results. On the other hand, from the viewpoint of resource utilization, the proposed heuristic approach efficiently allocates resources with high utility. We obtained the maximum utilization result for computing resources such as CPU, memory and bandwidth. Most exiting systems consider only two resources, CPU and memory, in evaluating their performance the proposed system adds bandwidth as a resource. Future work will focus on more effective scheduling algorithms in which turnaround time and response time will be improved.

References:

1. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) *Internet of things (iot): a vision, architectural elements, and future directions*. Futur Gener Comput Syst 29(7):1645–1660

2. Mezmaz M, Melab N, Kessaci Y, Lee YC, Talbi E-G, Zomaya AY, Tuyttens D (2011) *A parallel bi-objective hybrid meta heuristic for energy-aware scheduling for cloud computing systems.* J Parallel Distributed Computing 71(11):1497–1508

3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al (2010) *A view of cloud computing*. Commun ACM 53(4):50–58

4. Tsai J-T, Fang J-C, Chou J-H (2013) *Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm.* Comput Oper Res 40(12):3045–3055

5. Maguluri ST, Srikant R (2014) *Scheduling jobs with unknown duration in clouds.* IEEE/ACM Trans Netw (TON) 22(6):1938–1951
6. Cheng C, Li J, Wang Y (2015) *An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing*. Tsinghua Sci Technol 20(1):28–39
7. Lin W, Liang C, Wang JZ, Buyya R (2014) *Bandwidth-aware divisible task scheduling for cloud computing.* Software: Practice and Experience 44(2):163–174
8. Ergu D, Kou G, Peng Y, Shi Y, Shi Y (2013) *The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment*. The Journal of Supercomputing. 64(3):835-848
9. Zhu X, Yang LT, Chen H, Wang J, Yin S, Liu X (2014) *Real-time tasks oriented energy-aware scheduling in virtualized clouds.* IEEE Transactions on Cloud Computing 2(2):168–180
10. Liu X, Zha Y, Yin Q, Peng Y, Qin L (2015) *Scheduling parallel jobs with tentative runs and consolidation in the cloud.* J Syst Softw 104:141–151
11. Shamsollah G, Othman M (2012) *Priority based job scheduling algorithm in cloud computing.* Procedia Engineering 50:778–785
12. Polverini M, Cianfrani A, Ren S, Vasilakos AV (2014) *Thermal aware scheduling of batch jobs in geographically distributed data centers.* IEEE Transactions on Cloud Computing 2(1):71–84
13. Rodriguez MA, Buyya R (2014) *Deadline based resource provisioningand scheduling algorithm for scientific workows on clouds*. IEEE Transactions on Cloud Computing 2(2):222–235
14. Keshk AE, El-Sisi AB, Tawfeek MA (2014) *Cloud task scheduling for load balancing based on intelligent strategy*. Int J Intell Syst Appl 6(5):25
15. Ghanbari S, Othman M, Leong WJ, Bakar MRA (2014) *Multi-criteria based algorithm for scheduling divisible load.* In: Proceedings of the first international conference on advanced data and information engineering (DaEng-2013), pp 547–554