International Journal of Technology and Engineering System (IJTES)
Vol 7. No.4 2015 Pp. 263-267

available at : www.ijcns.com
ISSN: 0976-1345

KEY-AGGREGATE CRYPTOSYSTEM AND PARALLEL ALLOCATION IN CLOUD STORAGE

M.SATHISHKUMAR,R.SATHYAMOOTHI, B.VENKATESH,S.VIGNESHKUMAR
Dept. Of CSE, V.R.S.College Of Engg.And Tech,
Villupuram

ABSTRACT

Today Cloud computing is on demand as it offers dynamic flexible resource allocation, for reliable and guaranteed services in pay-as-you-use manner, to Cloud service users. So there must be a provision that all resources are made available to requesting users in efficient manner to satisfy their needs. This resource provision is done by considering the Service Level Agreements (SLA) and with the help of parallel processing. Recent work considers various strategies with single SLA parameter. Hence by considering multiple SLA parameter and resource allocation by preemption mechanism for high priority task execution can improve the resource utilization in Cloud. In this paper we propose an algorithm which considered Preemptable task execution and multiple SLA parameters such as memory,network bandwidth, and required CPU time. An obtained experimental results show that in a situation where resource contention is fierce our algorithm provides better utilization of resources.

Keywords- Cloud computing, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Resource management, Software as a Service (SaaS), Virtual machine, Virtualization.

I . INTRODUCTION

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software and information are provided to users over the network. Cloud computing providers deliver application via the Internet, which are accessed from web browser, while the business software and data are stored on servers at a remote location. Cloud providers are able to attain the agreed SLA, by scheduling resources in efficient manner and by deploying application on proper VM as per the SLA objective and at the same time performance of the applications must be optimized. Presently, there exists a more work done on scheduling of applications in Clouds [1], [2], [3]. These approaches are usually considering one single SLA objective such as cost of execution, execution time, etc. Due to combinatorial nature scheduling algorithm with multiple SLA objective for optimal mapping of workload with multiple SLA parameters to resources is found to be NPhard [4]. The available solutions are based on the use of heuristics.

When a job is submitted to the clouds, it is usually partitioned into several tasks. Following two questions are to be consider when applying parallel processing in executing these tasks: 1) how to allocate resources to tasks; 2) task are executed in what order in cloud; and 3) how to schedule overheads when VMs prepare,

terminate or switch tasks. Task scheduling and resource allocation can solve these three problems. In embedded systems [5], [6] and in high performance computing [7], [8] task scheduling and resource allocation have been studied. Typically, efficient provisioning requires two distinct steps or processes: (1) initial static planning step: the initially group the set of VMs, then classify them and deployed onto a set of physical hosts; and (2) dynamic resource.

II . RELATEDWORK

In [9] author proposed architecture, using feedback control theory, for adaptive management of virtualized resources, which is based on VM. In this VM-based architecture all hardware resources are pooled into common shared space in cloud computing infrastructure so that hosted application can access the required resources as per their need to meet Service Level Objective (SLOs) of application. The adaptive manager use in this architecture is multi-input multi-output (MIMO) resource manager, which includes 3 controllers: CPU controller, memory controller and I/O controller, its

goal is regulate multiple virtualized resources utilization to achieve SLOs of application by using control inputs per-VM CPU, memory and I/O allocation.

The seminal work of Walsh et al. [10], proposed a general two-layer architecture that uses utility functions, 2013 International Conference on Intelligent Systems and Signal Processing (ISSP) adopted in the context of dynamic and autonomous resource allocation, which consists of local agents and global arbiter. The responsibility of local agents is to calculate utilities for given current or forecasted workloads and range of resources, for each AE and results are transfer to global arbiter. Where, global arbiter computes near-optimal configuration of resources based on the results provided by the local agents. In global arbiter, the new configurations applied by assigning new resources to the AEs and the new configuration computed either at the end of fixed control intervals or in an event triggered manner or anticipated SLA violation. In [11], authors propose an adaptive resource allocation algorithm for the cloud system with preempt able tasks in which algorithms adjust the resource allocation adaptively based on the updation of the actual task executions. Adaptive list scheduling (ALS) and adaptive min-min scheduling (AMMS) algorithms are used for task scheduling which includes static task scheduling, for static resource allocation, is generated offline. The online adaptive procedure is used for re-evaluating the remaining static resource allocation repeatedly with predefined frequency. In each re-evaluation process, the schedulers are re-calculating the finished time of their respective submitted tasks, not the tasks that are assigned to that cloud. The dynamic resource allocation based on distributed multiple criteria decisions in computing cloud explain in [12].

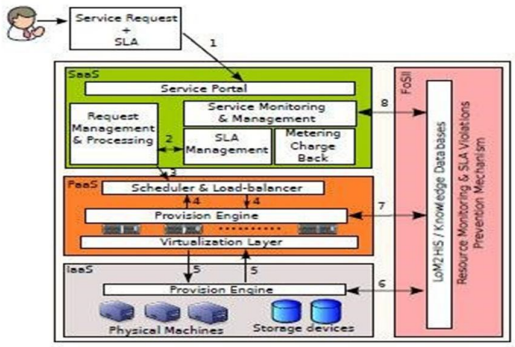
In it, the author's contribution is tow-fold, first distributed architecture is adopted, in which resource management is divided into independent tasks, each of which is performed by Autonomous Node Agents (NA) in ac cycle of three activities: (1) VMPlacement, in it suitable physical machine (PM) is found which is capable of running a given VM and then assigned VM to that PM, (2) Monitoring, in it total resources use by hosted VM are monitored by NA, (3) In VMSelection, if local accommodation is not possible, a VM need to migrate at another PM and process loops back to into placement. And second, using PROMETHEE method, NA carry out configuration in parallel through multiple criteria decision analysis.

This approach is potentially more feasible in large data centers than centralized approaches. The problem of resource allocation is considered in [13], to optimize the total profit gained from the multidimensional SLA contracts for multi-tire application. In it the upper bound of total profit is provided with the help of force-directed resource assignment (FRA) heuristic algorithm, in which initial solution is based on provided solution for profit upper bound problem. Next, distribution rates are fixed and local optimization step is use for improving resource sharing. Finally, a resource consolidation technique is applied to consolidate resources to determine the active (ON) servers and further optimize the resource assignment.

III. USETECHNIQUES

In this section we are describing SLA based resource provisioning and online adaptive scheduling for Preemptable task execution, these two methodologies which are combined in proposed algorithm for effective utilization of cloud resources to meet the SLA objective.

A. Cloud Resource provisioning and schedulingheuristic



The service requests from customers host by combining the three different layers of resource provisioning as shown in following figure 1[24]. Service deployment requests from customers is placed to the service portal (step 1 in Figure1), which forward the requests to the request management and processing component to validate the requests with the help of SLA(step 2). If the request is valid, it is then passed to the scheduler and load-balancer (step 3). For deploying the requested service, scheduler selects the appropriate VMs, as per SLA and priority, through the provisioning engine in PaaS layer and the load-balancer balances the service provisioning among the running VMs (step 4). The VMs on the virtualization layer manage by provision engine and the virtualization layer interacts with the physical resources with the help of the provision engine in IaaS layer (step 5). The LoM2HiS framework monitors the lowlevel resource metrics of the physical resources at IaaS layer [25] (step 6). If SLA violation occurs, reactive actions are provided by the knowledge database techniques [26] in FoSII (step 7). The requested service status and the SLA information are communicated back with the service portal (step 8). Provisioning can be done at the single layers alone. However, approach which we considered in [24] aims to provide an integrated resource provisioning strategy.

The SLA terms are used to find a list of appropriate VMs (AP) capable of provisioning the requested service (R). The load-balancer is presented below in Algorithm 1. Appropriate VM list is provided as input to it, (line 1 in Algorithm 2). In its operations, in order to know how to balance the load among the VMs it first finds out the number of available running VMs in the data centre (line 2). In the next step, it gets a list of VMs which are

already allocated to job i.e. list of used VMs. (line 3). It clears the list if this list is equal to the number of running VMs, because that means all the VMs are currently allocated to some applications (lines 4-7). Algorithm 1 Load Balancer Input: AP(R,AR) availableVMList //list of available VMs form each cloud usedVMList //list of VMs,currently provision to certain job deployableVm=null if size(usedVMList)= size(availbleVMList) then clear usedVMList End if for vm in (AP,R,AR) do if vm not in usedVMList then Add VM to usedVMList deployableVm= vm Break End if End for Return deployableVm Therefore, the first VM from the appropriate and available VM list can be selected for the deployment of the new job request. Lastly, the selected VM will be added to the list of used VMs so that the load-balancer will not select it in the next iteration (lines 8-15).

B.Preemptable task execution

When a scheduler receives customer's service request, it will first partition that service request into tasks in the form of a DAG. Then initially static resource allocation is done. In [11] authors proposed two greedy algorithms, to generate the static allocation: the cloud list scheduling (CLS) and the cloud min-min scheduling (CMMS).

1) Cloud list scheduling (CLS): This CLS is similar to CPNT [27]. The definitions used for listing the task are provided as follow. The earliest start time (EST) and the latest start time (LST) of a task are shown as in (1) and (2).The entry-tasks have EST equals to 0. And The LST of exit-tasks equal to their EST.

Algorithm 2 Forming a task list based on priorities

Require (input): A DAG, Average execution time AT of every task in the DAG Ensure (output): A list of task P based on priorities

The EST is calculated for every task

The LST is calculated for every task 3. Empty list P and stack S, and pull all task in the list of task U

4. Push the CN task into stack S in decreasing order of their LST 5. While the stack S is not empty do

If top(S) has un-stacked immediate predecessors then

S the immediate predecessor with least

LST

Else

P top(S)

Pop top(S)

End if

End while

Once the above algorithm 2 forms the list of task according their priority, we can allocate resources to tasks in the order of formed list. When all the predecessor tasks of the assigned task are finished and cloud resources allocated to them are available, the assigned task will start its execution. This task is removed from the list after its assignment. This procedure is repeats until the list is empty.

2) Cloud min-min scheduling (CMMS): Min-min scheduling is popular greedy algorithm [28]. The dependencies among tasks not considered in original minmin algorithm. So in the dynamic min-min algorithm used in [2], authors maintain the task dependencies by updating the mappable task set in every scheduling step. The tasks whose predecessor tasks are all assigned are placed in the map able task set. Algorithm 3 shows the pseudo codes of the CMMS algorithm.

A cloud scheduler record execution schedule of all resources using a slot. When an AR task is assigned to a cloud, first resource availability in this cloud will be checked by cloud scheduler. Since besteffort task can be preempted by AR task, the only case when most of resources are reserved by some other AR task. Hence there are not enough resources left for this AR task in the required time slot. submitted to this cloud, not the tasks that are assigned to this cloud.

IV . SCHEDULING ALGORITHM

In proposed priority based scheduling algorithm we have modified the scheduling heuristic in [24] for executing highest priority task with advance reservation by preempting best-effort task as done in [11]. Algorithm 4 shows the pseudo codes of priority based scheduling algorithm (PBSA).

Algorithm 3 Priority Based Scheduling

Algorithm (PBSA)

Input: UserServiceRequest

//call Algorithm 2 to form the list of task based on priorities

3.getglobalAvailableVMList and gloableUsedVMList and also availableResourceList from each cloud schedular

4. // find the appropriate VMListfromeach cloud scheduler 5. if AP(R,AR) != then

// call the algorithm 1 load balancer

deployableVm=load-

balancer(AP(R,AR))

8. Deploy service on deployableVM

deploy=true

Else if R has advance reservation and best-effort task is running on any cloud then 11. // Call algorithm 3 CMMS for executing

R with advance reservation

12. Deployed=true

13.Else if global Resource Able To Host

Extra VM then

Start new VM Instance

Add VM To Available VM List

Deploy service on new VM

Deployed=true

Else

queue service Request until

queue Time > waiting Time

Deployed=false

End if

If deployed then

return successful;

terminate;
Else 27. return failure;
28. terminate.

As shown above in Algorithm 4, the customers' service deployment requests (R), which are composed of the SLA terms (S) and the application data (A) to be provisioned, are provided as input to
scheduler (line 1 in Algorithm)
1). When service request (i.e. job) arrive at cloud scheduler, scheduler divide it in tasks as per their dependencies then the Algorithm 2 is called to form the list of tasks based on their priority.
2). In the first step, it extracts the SLA terms, which forms the basis for finding the VM with the appropriate resources for deploying the application. In next step, it collects the information about the number of running VMs in each cloudand the total available resources .
3). According to SLA terms appropriate VMs (AP) list is form, which are capable of provisioning the requested service .

Once the list of appropriate VMs is formed, the Algorithm 1- load-balancer decides which particular VM is allocated to service request in order to balance the load in the data center of each cloud (lines 6-9). When there is no VM with the appropriate resources running in the data center of any cloud, the scheduler checks if the service request (R) has advance reservation then it search for best-effort task running on any cloud or not, if it found besteffort task then it calls Algorithm 3 CMMS for executing advance reservation request by preempting best-effort task (lines 10-12).

V . EXPERIMENTAL RESULTS

Experiment setup
We evaluate the performance of our priority based scheduling algorithm using CloudSim simulator. CloudSimis a scalable simulation tool offering features like support for modeling service brokers, resource provisioning, application allocation policies, and simulation of large scale Cloud computing environments including data centers, on a single computing machine. Further information about CloudSim can be found in [29]. With different set of jobs simulation is done in 10 runs. In each run of simulation, we simulate a set of 70 different service requests (i.e. jobs), and each service request is composed of up to 18 sub-tasks. We consider 4 clouds in the simulation. All 70 service requests will be submitted to random clouds at arbitrary arrival time. Among these 70 service request, 15 requests are in the AR modes, while the rest are in the best-effort modes, with different SLA objectives. The scheduler will re-schedule these tasks with a predefined probability . The parameters in Table 1 are set in simulation randomly according to their maximum and minimum values. Since we focus only on the scheduling algorithms, we do our simulations locally without implementing in any exiting cloud system or using VM interface API.

We consider two situations for arrival of service request. In first situation, called as loose situation, we spread arrival time of request widely over time so that request does not need to contend resources in cloud. In other situation we set arrival time of requests close to each other, so known as tight situation. The time elapses from request is submitted to the request is finished, is defined as execution time.

VI . RESULTS

The average job execution time in loose situation. We find out that the PBSA algorithm has the minimum average execution time. The resource contentions occur when best-effort job is preempted by AR job. As resource contention is less in loose situation, so that estimated finish time of job is close to the actual finish time. Hence adaptive procedure does not impact the job execution time significantly.

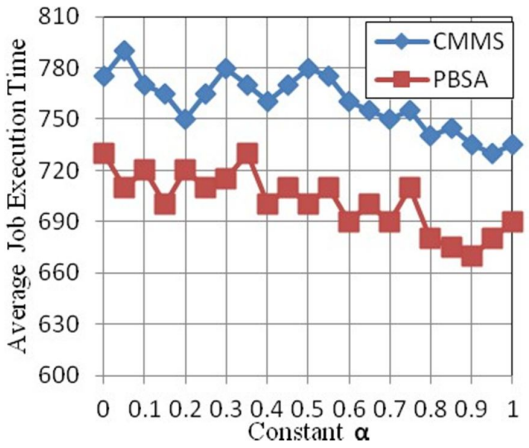


Figure 2. Average job execution time in loose situation
In figure 3 tight situation results are shown in which PBSA performs better than CMMS. In tight situation resource contention is more so the actual finish time of job is often later than estimated finish time. As AR job pre-empt best-effort job, the adaptive procedure with updated information works more significantly in tight situation.

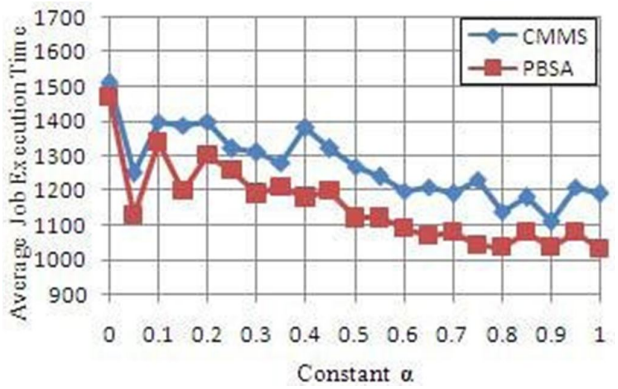


Figure 3. Average job execution time in tight situation

VII . CONCLUSIONS

In this paper, we present dynamic resource allocation mechanism for Preemptable jobs in cloud. We propose priority based algorithm, in which considering multiple SLA objectives of job, for dynamic resource allocation to AR job by preempting best-effort job. Simulation results show that PBSA perform better than CMMS in resource contention situation. The extension or future work can be the prediction VM which will be free earlier and according its capability selecting the task from waiting queue for execution on that VM.

REFERENCES

S. K. Garg, R. Buyya, and H. J. Siegel, "Time and cost trade off management for scheduling parallel applications on utility grids," Future Generation. Computer System, 26(8):1344–1355, 2010.

S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimizationbased heuristic for scheduling workflow applications in cloud computing environments," in AINA '10: Proceedings of the 2010, 24th IEEE International Conference on Advanced Information Networking and Applications, pages 400–407, Washington, DC, USA, 2010, IEEE Computer Society.

M. Salehi and R. Buyya, "Adapting market-oriented scheduling policies for cloud computing," In Algorithms and Architectures for Parallel Processing, volume 6081 of Lecture Notes in Computer Science, pages 351–362. Springer Berlin / Heidelberg, 2010.

J. M. Wilson, "An algorithm for the generalized assignment problem with special ordered sets," Journal of Heuristics, 11(4):337–350, 2005.

M. Qiu and E. Sha, "Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems," ACMTransactions on Design Automation of Electronic Systems (TODAES), vol. 14, no. 2, pp. 1–30, 2009.

M. Qiu, M. Guo, M. Liu, C. J. Xue, and E. H.-M. S. L. T. Yang, "Loop scheduling and bank type assignment for heterogeneous multibank memory," Journal of Parallel and Distributed Computing(JPDC), vol. 69, no. 6, pp. 546– 558, 2009.

A. Dogan and F. Ozguner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," IEEE Transactions on Parallel and Distributed Systems, pp. 308–323, 2002.

T. Hagras and J. Janecek, "A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems," Parallel Computing, vol. 31, no. 7, pp. 653–670, 2005.

"Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control," in First International Conference on Information Science and Engineering, April 2010, pp. 99-102.

W. E. Walsh, G. Tesauro, J. O.
Kephart, and R. Das, "Utility Functions in Autonomic Systems," in ICAC '04: Proceedings of the First International Conference on Autonomic Computing.
IEEE Computer Society, pp. 70–77, 2004.

Jiayin Li, MeikangQiu, Jian-Wei Niu,
Yu Chen, Zhong Ming, "Adaptive Resource Allocation for Preempt able Jobs in Cloud Systems," in 10th International Conference on Intelligent System Design and Application, Jan. 2011, pp. 31-36.