EFFICIENT DYNAMIC RESOURCE ALLOCATION IN CLOUD ENVIRONMENT USING SALB ALGORITHM

Kanakaraj D[1], Sreedhar Kumar S[2]

[1, 2] *Computer Science and Engineering, DBIT,VTU, Karnataka, (India)*

IJARSE

ABSTRACT

To efficiently utilize their infrastructure and thus increase their revenue, cloud providers need mechanisms to provide resource allocation and performance isolation for different tenants in the shared platform. For cloud service providers, packing VMs onto a small number of servers is an effective way to reduce energy costs, so as to improve the efficiency of the data center. However allocating too many VMs on a physical machine may cause some hotspots which violate the SLA of applications. Load balancing of the entire system is hence needed to guarantee the SLA. In this paper, we present a simulated-annealing load balancing algorithm for solving the resource allocation and scheduling problem in a cloud computing environment.

Index Terms*: *Cloud Computing, Simulated Annealing, Resource Allocation

I. INTRODUCTION

Virtualization is causing a disruptive change in enterprise data centers and giving rise to a new paradigm: shared virtualized infrastructure. Unlike the traditional hosting model where applications run on dedicated nodes, resulting in low resource utilization, this model allows applications to be consolidated onto fewer nodes, reducing capital expenditure on infrastructure as well as operating costs on power, cooling, maintenance, and support. It also leads to much higher resource utilization on the shared nodes[1].

Server consolidation can be static or dynamic. In static consolidation historical average resource utilizations are typically used as input to an algorithm that maps VMs to PMs. After initial static consolidation the mapping may not be recomputed for long periods of time, such as several months, and is done off-line. In contrast, dynamic allocation is implemented on shorter timescales, preferably shorter than periods of significant variability of the resource demand. Dynamic allocation leverages the ability to do live migration of VMs[2].

The policies for allocating resources in a hosting center, with a principal focus on energy management[3]. We present the design and implementation of a flexible resource management architecture—called Muse—that controls server allocation and routing of requests to selected servers through a reconfigurable switching infrastructure. Muse is based on an economic model in which customers "bid" for resources as a function of service volume and quality. We show that this approach enables adaptive resource provisioning in accordance with flexible Service Level Agreements (SLAs) specifying dynamic tradeoffs of service quality and cost. In addition, Muse promotes energy efficiency of Internet server clusters by balancing the cost of resources (e.g., energy) against the benefit realized by employing them. We show how this energy-conscious provisioning allows the center to automatically adjust on-power capacity to scale with load, yielding a significant energy savings for typical Internet service workloads.

Virtual machine monitors (VMMs) like Xen provide a mechanism for mapping virtual machines (VMs) to physical resources [3]. This mapping is largely hidden from the cloud users. VM live migration technology makes it possible to change the mapping between VMs and PMs while applications are running [5], [6]. However, a policy issue remains as how to decide the mapping adaptively so that the resource demands of VMs are met while the number of PMs used is minimized. This is challenging when the resource needs of VMs are heterogeneous due to the diverse set of applications they run and vary with time as the workloads grow and shrink. The capacity of PMs can also be heterogeneous because multiple generations of hardware coexist in a data center.

In this paper, we propose a Simulated Annealing Load Balancing (SALB) algorithm to find the optimal resource allocation in cloud computing systems. It minimizes the standard deviation of the hosts' load and balance the entire system load. The authors in [7] proposed a VM allocation strategy on load balancing of VM resources applying a genetic algorithm, which considers the VM to be deployed one by one. This approach may fall into a local optimal solution. Instead, we consider the VM allocation based on the whole system state. We use the CloudSim toolkit to simulate this new scheduling strategy[9], [8]. In the experiments we consider the Round Robin scheduling strategy and basic Simulated Annealing algorithm and compare them with the SALB algorithm. The experiment results show that the proposed SALB algorithm can achieve a better system balanced load.

II.PROPOSED WORK

Load balancing problem has been an important topic in the management of data centers of cloud computing. It aims to guarantee that each computing resource is distributed effectively, ultimately to improve resource utilization.

2.1 The Traditional Method

Cloud computing is developed based on the technology of centralized management of distributed resources through virtualization. In traditional computing environments a range of static, dynamic and mixed load-balancing resource allocation algorithms are proposed. In static scheduling, ISH[11], MCP[12] and ETF[13] based on BNP are appropriate for small distributed environments, and in those environments the communication cost is ignorabled. MH[14] and DSL[15] algorithms based on APN are suitable for large distributed environment because they take the communication delay and execution time into account. In dynamic scheduling, some methods ensure the load balancing through self-adapting and intelligent distribution of tasks. In mixed scheduling, equal distribution of allocated computing task and the communication cost of computing nodes are emphasized. It achieves balanced scheduling according to the computational capability of each node. In addition, other scheduling methods are also considerate such as autonomic scheduling, central scheduling and agent negotiated scheduling.

IJARSE

2.2 Load Balancing In Cloud Environment

An important characteristic for resource scheduling in the cloud computing environment is its target, that is, VM resources are scheduled so the granularity is extraordinarily large and the communication data is also large. Currently, research on load balancing in the data centers is mainly based on live migration of virtual machines[10]. Sandpiper[16] considered dynamic monitoring and hotspot exploring on the utilitiation of system's CPU, memory resources and network bandwidth. Based on the white-box and black-box strategies, a resourcemonitoring method was also proposed. The system focuses on how to define the hotspots and how to deal with the hotspots when they occur. It uses the VM migration method to realize the remapping of resources to achieve load balancing. VMware's Distributed Resource Scheduler (DRS) [13] makes use of dynamic migration to realize automated load balancing as the respond to the changes of CPU and memory. VMware DRS also uses a userspace application to monitor memory usage which is similar to Sandpiper's gray-box monitor. However it cannot use application logs to respond directly to potential SLA violations. With the dramatic change of workload in VMs, VMware DRS will remap VMs between physical servers and migrate VMs among physical servers through VMware and VMotion. Many meta-heuristic algorithms have also been proposed to schedule cloud resources [17], [18]. The authors in [7] proposed a scheduling strategy according to historical data and the current state of the system. It computes in advance the influence it will have on the system after the deployment of the needed VM resources and then chooses the least-affective solution.

III. PROBLEM DESCRIPTION

In Figure 1, we show a typical virtualized IT environment consisting of a set of physical machines and virtual machines. Each physical server can host one or more VMs and each VM implements one application. Let J = {1, 2,….m} denote the

set of physical machines in the system and I = {1, 2,….n} the set of VMs needed to be deployed. In order to simplify the presentation, we assume the physical server environment is homogeneous, i.e., all physical machines have the same capacity c. We assume the whole capacity of physical servers is enough to hold all the VMs.

Through resource monitoring modules we can collect the resource utilization of the VMs. Each VMs_i ($i \in I$) has a workload V_i. We use binary variables x_{ij} to indicate the deployment of VMs:

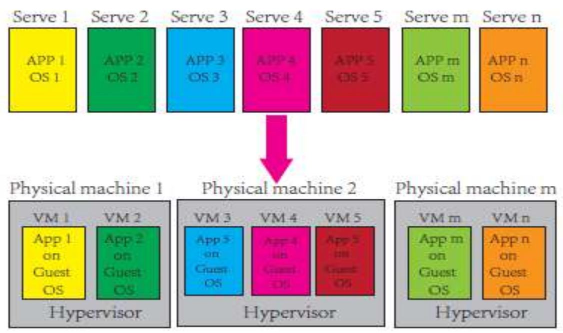


Figure 1. A Typical Virtualized I/T Environment

IJARSE

$$x_{ij} = \begin{cases} 1, & \text{the } i^{th} \text{ VM deploied on } j^{th} \text{ Physical Machine(PM)} \\ 0, & \text{otherwise} \end{cases}$$

Then the workload of a physical machine can be acquired by adding the loads of the VMs running on it. Let Pj be the load of physical machine j ϵ J:

$$P_j = \sum_{i=1}^{m} V_i \cdot x_{ij}$$

To aviod the SLA violation, we should keep the workload balanced among the physical machines. We use the amount of available resource on each physical machine as the load balancing metric. Let r_j be the residual capacity of physical machine j ϵ J, r be the average of the residual capacities of all physical machines :

$$r_j = c - P_j,\ \bar{r} = \frac{1}{m}\sum_{j=1}^{m} r_j$$

Let σ(s) be the standard deviation of the residual capacity in each physial machine:

$$\sigma(s) = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(r_j - \bar{r})^2}$$

where s denotes the mapping solutions.

The object is to find the best mapping solution s so as to achieve the best system load balancing. Then the model can be describ ed as follows:

$$min \quad C(s) = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(r_j - \bar{r})^2}$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij} = 1 \qquad (1)$$

$$P_j = \sum_{i=1}^{m} V_i \cdot x_{ij} \leq c \qquad (2)$$

$$r_j = c - P_j, \quad \bar{r} = \frac{1}{m}\sum_{j=1}^{m} r_j \qquad (3)$$

$$x_{ij} \in \{0,1\}, \quad y_i \in \{0,1\} \qquad (4)$$

$$i \in I, \quad j \in J \qquad (5)$$

Here C(s) is the objective function, constraint (1) means each VMs can be deployed on only a physical machines, and constraint (2) denotes the workload of a physical machine cannot exceed its capacity.

IV. A SIMULATED ANNEALING SCHEDULING STRATEGY

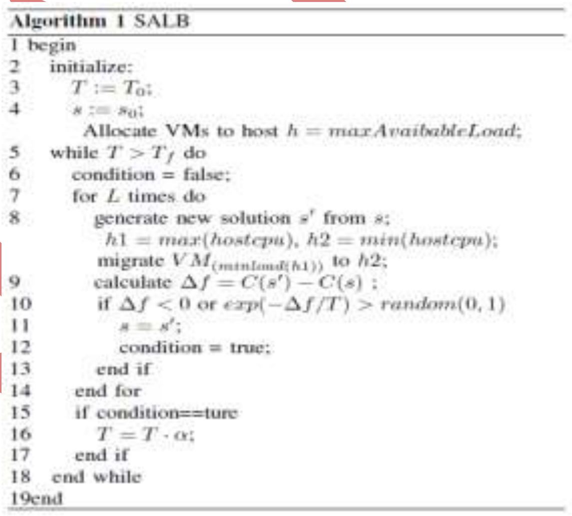
The simulated annealing technique (SA) was initially proposed to solve the hard combinatorial optimization problems through controlled randomization by simulating the temperature falling procedure of particular systems in thermodynamics. It is a technique to find a better solution for an optimization problem by trying random variations of the current solution. The main feature is that a worse variation may be accepted as a new solution with a

IJARSE

probability, which results in the SA's major advantage over other searching methods, that is, the ability to avoid becoming trapped at local minima. Theoretically SA is able to find the global optimal solution with probability equal to 1.

In the basic simulated annealing algorithm, an initial solution is always selected from the variation range of each of the parameters at random. But in this work we obtain the initial solution s_0 as follows: first we maintain a list of all the VMs in descending order according to their loads, allocating them to the physical servers in sequence. Then, we map the first virtual machine in the list which has the most workload to a physical machine which has the most residual capacity, and repeat this process for the next virtual machine until all the VMs have been allocated to a physical server. At last, we obtain an initial solution s0 as an input of the simulated annealing load balancing algorithm. Through this initial allocation, we can get a relatively good feasible initial solution, the standard deviation of the residual capacity in each physical machine $\sigma(s_0)$ will not be too large. A feasible neighboring solution is generated from the neighboring function when migrating a virtual machine. On the current solution s, we select the VM with the highest resource utilization from the physical machine with the least residual capacity, move it to a physical machine which has the most capacity, to balance the workload among physical machines. Then we get the new solution s_0 from s. In order to reduce the amount of calculation, we maintain a list of all the physical machines in ascending order according to their residual capacities. As the result, each time when we migrate a VM from the first physical machine on the list to the last physical machine. After moving the VM, the residual capacity is re-calculated and the list is re-ordered.

The detailed steps of the load-balancing simulated annealing algorithm can be described as following:



Algorithm 1 SALB

```
1 begin
2   initialize:
3     $T := T_0$;
4     $s := s_0$;
        Allocate VMs to host $h = maxAvaibableLoad$;
5   while $T > T_f$ do
6     condition = false;
7     for $L$ times do
8       generate new solution $s'$ from $s$;
          $h1 = max(hostcpu)$, $h2 = min(hostcpu)$;
        migrate $VM_{(minload(h1))}$ to $h2$;
9       calculate $\Delta f = C(s') - C(s)$ ;
10      if $\Delta f < 0$ or $exp(-\Delta f/T) > random(0,1)$
11        $s = s'$;
12        condition = true;
13      end if
14    end for
15    if condition==ture
16      $T = T \cdot \alpha$;
17    end if
18  end while
19end
```

V.EXPERIMENTAL RESULTS

We compared the proposed SALB algorithm with the Round Robin (RR) and the basic Simulated Annealing [1]. The RR algorithm allocates VMs to the host one by one, for example the first VM is allocated to the first host, the

IJARSE

second VM is allocated to the second host, the third VM is allocated to the third host and so on. If the host is the last, then the next VM will allocated to the first host circularly. The basic SA algorithm gains the initial solution randomly and stochastically generates the new solution. The SALB algorithm chooses optimal resources to perform tasks according to system status and the size of the given task in the Cloud environment. In the following experiments, we compared the average standard deviation (SD) of the host load of the RR, the basic SA and SALB algorithm with different tasks (represented by cloudlets). And the average utilization of the system of each algorithm is showed in the following experiments.

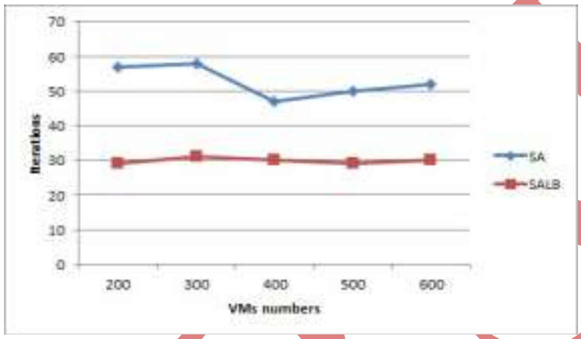


Figure 2. Number of physical machines vs virtual machines using SALB algorithm

Fig. 3 which shows that the average numbers of APMs remain essentially the same with or without load prediction (the difference is less than 1 percent). This is appealing because significant overload protection can be achieved without sacrificing resources efficiency. As the number of virtual machines increases the number of physical machines also increases which decreases the efficiency of cloud system.

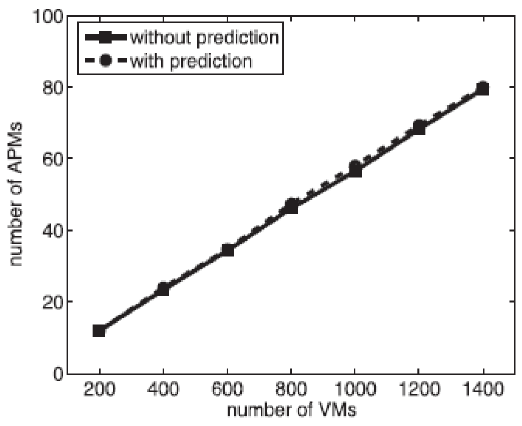


Figure 3. Number of physical machines vs virtual machines using skewness algorithm

VI. CONCLUSION

In this paper we proposed the Simulated Annealing Load Balancing (SALB) algorithm for solving the resource allocation and scheduling problem in a cloud computing environment. We evaluated it on the CloudSim environment, compared it with basic SA algorithm and skewness algorithm. By comparing both graphs we can ensure that number of active physical machines increases as number of VMs increases in case of skewness algorithm, but number of active PMs remain stable with increase in number of VMs in case of SALB algorithm. The experiment result demonstrates that the SALB can balance the load in the Cloud system effectively. In the future, we will take the multi-dimensional resource (such as memory, disk space, network bandwidth etc) into account, instead of only considering the CPU utility as the balance objective.

REFERENCES

[1] P. Padala, K.-Y. Hou, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, “Automated Control of Multiple Virtualized Resources,” Proc. ACM European conf. Computer Systems (EuroSys ’09), 2009.

[2] N. Bobroff, A. Kochut, and K. Beaty, “Dynamic Placement of Virtual Machines for Managing SLA Violations,” Proc. IFIP/IEEE Int’l Symp. Integrated Network Management (IM ’07), 2007.

[3] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle, “Managing Energy and Server Resources in Hosting Centers,” Proc. ACM Symp. Operating System Principles (SOSP ’01),Oct. 2001.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization,” Proc. ACM Symp. Operating Systems Principles (SOSP ’03), Oct. 2003.

[5] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live Migration of Virtual Machines,” Proc. Symp. Networked Systems Design and Implementation (NSDI ’05), May 2005.

[6] M. Nelson, B.-H. Lim, and G. Hutchins, “Fast Transparent Migration for Virtual Machines,” Proc. USENIX Ann. Technical Conf., 2005.

[7] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pages 89 –96, dec. 2010.

[8] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, CšŠsar A. F.De Rose, and Rajkumar Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. software: Practice and Experience, 41:23–50, 2011.

[9] Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, and Rajkumar Buyya. CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. CoRR, abs/0903.2525, 2009.

IJARSE

[10]Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live Migration of Virtual Machines. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation – Volume 2, pages 273–286, 2005.
[11] Hesham El-Rewini, Theodore G. Lewis, and Hesham H. Ali. Task Scheduling in Parallel And Distributed Systems. Prentice Hall, 1994.
[12] M.-Y. Wu and D.D. Gajski. Hypertool: A Programming Aid for Message-passing Systems. Parallel and Distributed Systems, IEEE Transactions on, 1(3):330 –343, jul 1990.
[13] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box And Gray-box Strategies for Virtual Machine Migration. In Proceedings of the 4th USENIX conference on Networked systems design &#38; implementation, pages 17–17, 2007.

IJARSE