AN ONLINE LOAD BALANCING SCHEDULING ALGORITHM FOR CLOUD DATA CENTERS CONSIDERING REAL-TIME MULTI-DIMENSIONAL RESOURCE

Minxian Xu, Wenhong Tian

School of Information and Software Engineering,
University of Electronic Science and Technology of China, Chengdu 610054, China
xmxyt900@gmail.com, tian_wenhong@uestc.edu.cn

Abstract: In general, load-balance scheduling is NP-hard problem as proved in many open literatures. We introduce an online load balancing resource scheduling algorithm (OLRSA) for Cloud datacenters considering real-time and multi-dimensional resources. Unlike traditional load balance scheduling algorithms which often consider only one factor such as the CPU load in physical servers, OLRSA treats CPU, memory and network bandwidth integrated for both physical machines and virtual machines. We develop and apply integrated measurement for each server and a Cloud datacenter. Simulation results show that OLRSA has better performance than a few related load-balancing algorithms with regard to total imbalance level, makespan, as well as overall load efficiency.

Keywords: Cloud computing; Load balance; On-line resource scheduling algorithm; Cloud data centers

1 Introduction

One key technology playing an important role in Cloud data centers is resource scheduling. There are quite many researches conducted in load balance scheduling algorithms. Most of them are for load balancing of traditional web servers or server farms. One of the challenging scheduling problems in Cloud data centers is to consider allocation and migration of reconfigurable virtual machines and integrated features of hosting physical machines. Unlike traditional load balance scheduling algorithms, which consider only physical servers with one factor such as CPU, OLRSA treats CPU, memory and network bandwidth integrated for both physical machines (PMs) and virtual machines (VMs).

The major contributions of this paper are:

- Providing a modeling approach to virtual machine scheduling problem with capacity sharing by modifying traditional interval scheduling problem and considering lifecycles and multi-dimensional characteristics of both VMs and PMs.
- Designing and implementing an online load balancing scheduling algorithm with computational complexity and approximation analysis.
- Providing performance evaluation of multiple metrics such as makespan, load efficiency and imbalance value by simulating different algorithms.

The remaining parts of this paper are organised as following: Section 2 discusses the related work on load balance algorithm. Section 3 introduces problem of formulation. Section 4 presents OLRSA algorithm in detail. Performance evaluation of different scheduling algorithms is shown in Section 5. Finally in Section 6, a conclusion is given.

2 Related works

Andre et al. [1] discussed the detailed design of a data center. Armbrust et al. [2] summarized the key issues and solutions in Cloud computing. Foster et al. [3] provided detailed comparison between Cloud computing and Grid computing. Buyya et al. Ref. [4] introduced a way to model and simulate Cloud computing environments. Wickremasinghe et al. [5] introduced three general scheduling algorithms for Cloud computing and provided simulation results. Wood et al. [6] introduced techniques for virtual machine migration and proposed some migration algorithms. Zhang [7] compared major load balance scheduling algorithms for traditional Web servers. Singh et al. [8] proposed a novel load balance algorithm called VectorDot to deal with hierarchical and multi-dimensional resources constraints by considering both servers and storage in a Cloud. Arzuaga et al. [9] proposed a quantifying measure of load imbalance on virtualized enterprise servers. Tian et al. [10] provided a comparative study of major existing scheduling strategies and algorithms for Cloud data centers. Sun et al. [11] present a novel heuristic algorithm to gain approximate optimal solution based on integrated resource scheduling. Tian et al. [12] introduced a dynamic load balance scheduling algorithm considering only current allocation period and multi-dimensional resource but without considering life-cycles of both VMs and PMs. Li et al. [13] proposed a cloud task scheduling policy based on ant colony optimization algorithm to balance the entire system and to minimize the makespan of a given task set. Galloway in Ref. [14] introduced an online greedy algorithm, in which PMs

978-1-4673-1857-0/12/$31.00 ©2012 IEEE

can be dynamic turned on and off but the life-cycle of a VM is not considered. Hu et al. [15] stated an algorithm named Genetic, which calculates the history data and current states to choose an allocation. Ref. [16] introduces load-balance techniques for VMware.

3 Problem formulation

3. 1 Problem description and formulation

In this paper we model the VM allocations as a modified interval-scheduling problem (MISP) with fixed processing time. More explanation and analysis about traditional interval scheduling problems with fixed processing time can be found in Ref. [17] and references there in. We present a general formulation of modified interval-scheduling problem and evaluate its results compared with well-known existing algorithms.

[Definition 1. Traditional interval scheduling problem (ISP) with fixed processing time]: A set of requests {1, 2, …, n} where the i-th request corresponds to an interval of time starting at si and finishing at fi , each request needs a capacity of 1, i.e. occupying the whole capacity of a machine during fixed processing time.

There are following assumptions:

1) All data are deterministic and unless otherwise specified, the time is formatted in slotted windows. As shown in Figure 2, we partition the total time period [0,T] into slots with equal length (s_0), the total number of slots is k=T/s_0. The start time s_i and finish time f_i are integer numbers of one slot. Then the interval of a request can be represented in slot format with (start-time, finish-time).

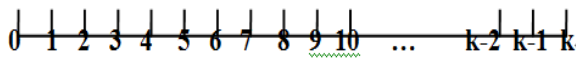

Figure 2 Time in slotted format

2) All tasks are independent. There are no precedence constraints other than those implied by the start and finish time.

3) The required capacity of each request is a positive real number between (0,1]. Notice that the capacity of a single physical machine is normalized to be 1.

4) Assuming that, when processed, each VM request is assigned to a single physical machine. Thus interrupting a request and resuming it on another machine is not allowed, unless explicitly stated otherwise.

5) Each physical machine is always available, i.e., each machine is continuously available in [0, ∞].

[Definition 2. Interval scheduling with capacity sharing (ISWCS)]: The only difference from traditional interval scheduling is that a resource (to be concrete, a physical machine) can be shared by different requests if the total capacity of all requests allocated on the single resource at any time does not surpass the total capacity that the resource can provide.

[Definition 3. Sharing compatible intervals for ISWCS]: A subset of intervals with total required capacity does not surpass the total capacity of a physical machine at any time; therefore they can share the capacity of a PM.

The formulation of ISWCS can be described as follows. Given a set of m identical machines (PMs) PM1, PM2, ..., PMm and a set of n requests (VMs), each request has a processing time (consider only CPU processing for example), the objective of load balance is to assign each request to one of PMs so that the loads placed on all machines are balanced.

[Theorem 1 The offline scheduling problem of finding an allocation of minimizing makespan in general case is NP-complete.

Remark: notice that Theorem 1 is considering offline load-balancing scheduling for single resource CPU on identical machines. When there are multiple resources to be considered and in heterogeneous case (like in this paper), the problem is more difficult and can be proved that it is NP-complete too (a detailed proof is provided in [17-18] by transforming the problem to 3-Dimensional matching problem or a multi-dimensional vector bin packing problem).

In this paper, each request needs only part of the whole capacity of a machine. So we redefine the makespan as capacity-makespan.

[Definition 4 Capacity-makespan] In any allocation of VM requests to PMs, we can let A(i) denote the set of VM requests allocated to machine PMi; under this allocation; machine PMi will have total load

$$L_i = \max_{j \text{ in } A(i)} c_j t_j \quad (1)$$

where cj is the CPU requests of VMj and tj is the span of request j (i.e., the length of processing time of request j). The goal of load balancing is to minimize the maximum load (makespan) on any PM, L = maxi Li. Some other related metrics such as imbalance value and load efficiency are also considered and will be explained in the following section.

3.2 Metrics for load balancing scheduling algorithms

In this section, a few existing metrics and new metrics for load balancing scheduling will be presented.

Zheng et al. in Ref. [19] introduced an integrated load balance index and load balance algorithm:

$$B = a \times \frac{N_{1i} \times C_i}{N_{1m} \times C_m} + b \times \frac{N_{2i} \times M_i}{N_{2m} \times M_m} + c \times \frac{N_{3i} \times D_i}{N_{3m} \times D_m} + d \times \frac{Net_i}{Net_m} \quad (2)$$

where i is the index of PM and m is the ID of referred PM, N1 is the capability of CPU, N2 is the parameter of memory, N3 refers to the parameter of bandwidth, C and M are the utilization of CPU and memory, D is the transferring rate of hard disk, Net is the network throughput, and a, b, c, d are the compared weighted value of CPU, memory, hard disk and network respectively and initialized as 1. The optimization goal is

finding the PM with the smallest B value to allocate requests.

For OLRSA algorithm, we take the following parameters into consideration:

1) PM resource: $PM_i(i, PCPU_i, PMem_i, PStorage_i)$, i is the index number of PM, $PCPU_i, PMem_i, PStorage_i$ are the CPU, memory, storage capacity of that a PM provides.

2)VM resource: $VM_j(j, VCPU_j, PMem_j, PStorage_j, T_j^{start}, T_j^{end})$, j is the VM type ID, $VCPU_j, VMem_j, VStorage_j$ are the CPU, memory, storage requirements of VM_j, T_j^{start}, T_j^{end} are the start time and end time, which are used to represent the life cycle of a VM.

3) Time slot: we consider a time span from 0 to T be divided into n parts with same length S_0. Then n parts can be defined as $[(t_1 - t_0), (t_2 - t_1), ..., (t_n - t_{n-1})]$, each time slot T_k means the time span $(t_k - t_{k-1})$.

4) Average CPU utilization of PM_i during some time period:

$$PCPU_i^U = \frac{\sum_{k=0}^{n}(PCPU_i^{T_k} \times T_k)}{\sum_{k-0}^{n} T_k} \quad (3)$$

And memory ($PMem_i^U$) and storage ($PStorage_i^U$) utilization of both PMs and VMs can be computed in the same way. Similarly average CPU utilization of a VM can be computed.

5) Integrated load imbalance value (ILB_i) of PM_i. The variance is widely used as a measure of how far a set of values are spread out from each other in statistics. Using variance, an integrated load imbalance value (ILB_i) of server i is defined as:

$$\frac{(Avg_i - CPU_u^A)^2 + (Avg_i - MEM_u^A)^2 + (Avg_i - Stor_u^A)^2}{3} \quad (4)$$

where

$$Avg_i = (PCPU_i^U + PMEM_i^U + PStorage_i^U)/3 \quad (5)$$

and CPU_i^U, MEM_u^A, $Stor_u^A$ are respectively the average utilization of CPU, memory and storage in a Cloud data center. ILB_i is applied to indicate load imbalance level comparing utilization of CPU, memory and network bandwidth of a single server itself. This metric is very similar to VMware DRS load balance metric—standard deviation as presented in Ref. [6].

6) Makespan and capacity-makespan:

In this paper, we define the makespan as capacity-makespan as given in Definition 4. Therefore the capacity-makespan of all PMs can be formulated as:

$$makespan = \max\left(\frac{PCPU_i^U + PMem_i^U + PStorage_i^U}{3}\right) \quad (6)$$

8) Load-efficiency (skew)

Load efficiency (skew) is defined as the (minimal average load /maximal average load) on all machines:

$$skew = \frac{\min(PCPU_i^U + PMem_i^U + PStorage_i^U)}{\max(PCPU_i^U + PMem_i^U + PStorage_i^U)} \quad (7)$$

Skew shows the load balancing efficiency to some degree.

9) Imbalance level (IBL)

Imbalance level of CPU is defined as:

$$IBD_{cpu} = \frac{\sum_{i=0}^{n}(PCPU_i^U - PCPU_{avg})^2}{n} \quad (8)$$

where $PCPU_{avg}$ is the average utilization of all CPUs in a data center. The imbalance level of memory IBD_{mem} and imbalance level of storage $IBD_{storage}$ can be obtained in the same way. Then total imbalance level of a data center is:

$$IBD_{total} = IBD_{cpu} + IBD_{mem} + IBD_{storage} \quad (9)$$

4 OLRSA algorithm

Figure 3 shows that the core process of OLRSA algorithm. For each request it firstly finds the PM with lowest average capacity-makespan, and a PM with next lowest average capacity-makespan would be turn-on only if there is no enough resource left on the first PM, so on so that all requests are allocated without rejection.

Input: VM requests (each indicated by their required VM type ID, start time, finish time, and requested capacity), the interval of start time and finish time of request i is denoted as I_i
Output: Assign a PM ID to each request and allocate an interval for each request.

```
1.  d=0;
2.  for j = from 1 to n do
3.      for all PMs, finding a PM with lowest average
    capacity-makespan, noted as PM_lowest (as in
    equation (3-6))
4.      if the request j still can share capacity of
    PM_lowest do
5.          allocate I_i to the PM
6.      else
7.          finding a PM with next lowest average
    capacity-makespan and can host the request
8.          d=d+1;
9.          allocate I_i to PM d
10.     endif
11. endfor
```

Figure 3 Pseudo code of OLRSA algorithm

Lemma 1. The computational complexity of OLRSA algorithm is *O(nlogm)* using priority queue data structure where n is the number of VM requests and m is the number of needed PMs.

Lemma 2. The approximation ratio (comparing to optimal solution) of OLRSA algorithm is (2-*1/m)* where m is the total number of machines [20].

5 Performance evaluation

In this section, we compare simulation results of different scheduling algorithms regarding total energy consumption.

5.1 Mythology and simulation setting

In this part, we will show the simulation results between the OLRSA algorithm and other existing algorithms. A Java discrete simulator is developed for this purpose. All simulations are conducted on a Pentium dual-core computer with 3.2GHz CPU and 2GB memory.

We compare the simulation results of our proposed algorithm with three existing algorithms:

1) Random Algorithm (Random): a general scheduling algorithm by randomly allocating the VM requests to the PM that can provide resource required.

2) Round-Robin (Round): a traditional load balancing scheduling algorithm by allocating the VM request one by one to each PM in turn that can provide resource required.

3) ZHJZ algorithm: as defined in Ref. [19], it selects a reference physical machine, and calculates the B value and chooses the physical machines with the lowest B value (as defined in Eq. (2)) and available resource to allocate virtual machines.

We do the simulation with enough PM that would satisfy all the VM requests (in the situation with VMs 500 and max duration 800, the PM number is type-1 92, type-2 93, type-3 63) and VM numbers vary from 100 to 750 (each type approximately 1/8). And we do the simulation with request duration time from 100 to 800 time slots (each slot is 5 seconds). The simulations for different algorithms are based on the same environment with same VM requests.

5.2 Simulation results and analysis

5.2.1 Random configuration of VMs and PMs

In this paper, we also adopt the following random configuration of VMs and PMs as shown in Tables I and II. Note that one compute unit (CU) has equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor [21].

Table I 8 types of virtual machine (VM) in Amazon EC2

Memory	Compute Units	Storage	VM Type
1.0GB	2units	2GB	1-1 (1)
4GB	10 units	8GB	1-2 (2)
12.0GB	16 units	15GB	1-3 (3)
91GB	3 units	5GB	2-1 (4)
20.0GB	6 units	15GB	2-2 (5)
36GB	13 units	25GB	2-3 (6)
1GB	1 units	25GB	3-1 (7)
4.0GB	2 units	50GB	3.1 (8)

Table II 3 types of physical machine (PM)

PM Pool Type	CPU (Compute Units)	Memory	Storage
Type1	64units	120GB	200GB
Type2	96units	180GB	300GB
Type3	128units	240GB	400GB

To simplify the corresponding relationship mentioned in section 3, we use VM type1, 2, 3 corresponds PM type1, type 4, 5, 6 corresponds PM type2 and VM 7, 8 corresponds VM type 4.

1) Fixing the total number of VM requests but vary their max durations

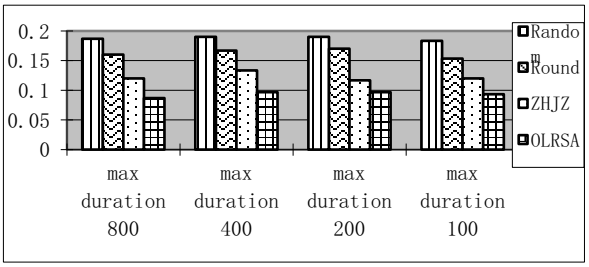

Figure 4 Imbalance value comparison when varying max duration of VMs

Figures 4 to 6 show the imbalance level, capacity-makespan and skew results respectively when fixing the total number of VM requests as 500 but varying the max duration of all VMs. From these figures, we can notice that OLRSA algorithm shows a better performance, as for imbalance level and makespan, Random>Round>Benchmark>OLRSA, as for skew, OLRSA is the larger than other three algorithms.

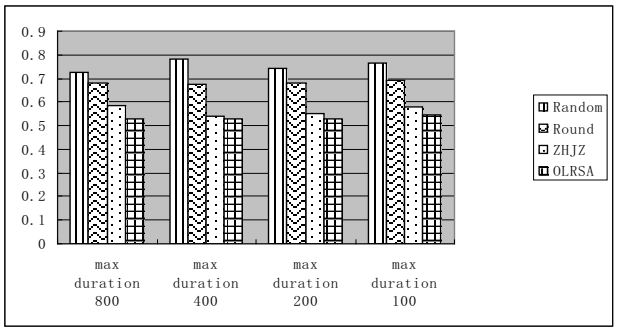

Figure 5 Capacity-makespan comparison when varying max duration of VMs (maximum is normalized to 1)

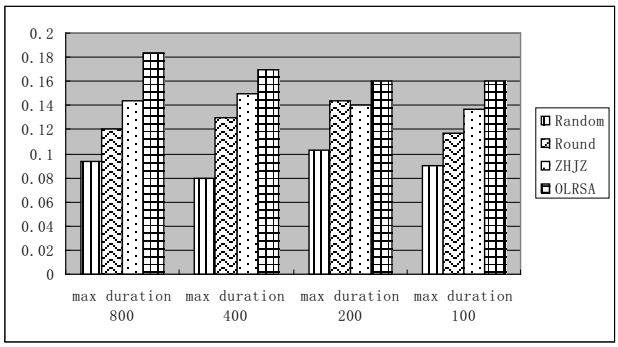

Figure 6 Skew comparison when varying max duration of VMs

2) Fixing max duration but varying the number of VM requests

Similar results are obtained. Because of page limit, we omit the details.

6 Discussion and conclusions

In this paper, we proposed an online load balance resource scheduling algorithm (OLRSA) to solve the real-time multi-dimensional resource scheduling problem in Cloud data centers. Simulations have shown that OLRSA has a better performance than a few existing algorithms at imbalance level, makespan and skew. We are comparing more algorithms with OLRSA and considering many scenarios. Besides, an approach is under study to combine load balance and energy-saving.

Acknowledgment

This research is sponsored by the National Natural Science Foundation of China (NSFC) Grant 61150110486.

References

[1] L. Andre, et al.,The Data center as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Ebook. 2009.

[2] M. Armbrust et al., Above the Coulds: A Berkeley View of Cloud Computing, technical report, 2009.

[3] I. Foster, Y. ZHAO, I. RAICU, S. Lu, Cloud Computing and Grid Computing 360-Degree Compared, IEEE International Workshop on Grid Computing Environments (GCE) 2008, co-located with IEEE/ACM Supercomputing 2008.

[4] R. Buyya., R. Ranjan, , R.N. Calhiros, Modeling and Simulation of Scalable Cloud computing environments and the CloudSim tookit: Challenges and opportunities, High Performance Computing & Simulation, 2009. International Conference on HPCS '09.

[5] B. Wickremasinghe et al., CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments, Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), Perth,Australia, April 20-23, 2010.

[6] T. Wood, et. al., Black-box and Gray-box Strategies for Virtual Machine Migration, in the proceedings of Symp. on Networked Systems Design and Implementation (NSDI), 2007.

[7] W. Zhang, Research and Implementation of Elastic Network Service, PhD dissertation, National University of Defense Technology, China (in Chinese) 2000.

[8] A. Singh, M. Korupolu, D. Mohapatra, Server-Storage Virtualization: Integration and Load balancing in Data Centers, International Conference for High Performance Computing, Networking, Storage and Analysis, 2008.

[9] E. Arzuaga, D. R. Kaeli, Quantifying load imbalance on virtualized enterprise servers, in the proceedings of WOSP/SIPEW'10, January 28-30, 2010, San Jose, California, USA.

[10] W. Tian, Adaptive Dimensioning of Cloud Data Centers: In the proceeding of the 8th IEEE International Conference on Dependable, Automatic and Secure Computing, DACS 2009.

[11] X. Sun, P. Xu, K. Shuang, et al., Multi-Dimensional Aware Scheduling for Co-optimizing Utilization in Data Center, China Communications 2011 8(6), 19-27.

[12] W. Tian, C.Jing, J.Hu, Analysis of resource allocation and scheduling policies in Cloud datacenter, in the proceedings of the IEEE 3rd International Conference on Networks Security Wireless Communications and Trusted Computing. March 2011.

[13] K. Li, G. Xu, G. Zhao,et al., Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization, chinagrid, pp.3-9, 2011 Sixth Annual ChinaGrid Conference, 2011.

[14] J. M. Galloway, K. L. Smith, S. S. Vrbsky, Power Aware Load Balancing for Cloud Computing, Proceedings of the World Congress on Engineering and Computer Science 2011 Vol I WCECS 2011, October 19-21, 2011.

[15] J. Hu; J.a Gu; G. Sun, et al., A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment, Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, vol., no., pp.89-96, 18-20 Dec.2010.

[16] A. Gulati, G. Shanmuganathan, A. Holler, I. Ahmad, Cloud-scale resource management: challenges and techniques, VMware Technical Journal, 2011.

[17] J. Kleinberg, E. Tardos, Algorithm Design, Pearson Education Inc., 2005.

[18] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson, Bin-Packing with Divisible Item Sizes, J. Complexity 3(1987), 406-428.

[19] H. Zheng, L. Zhou, J. Wu, Design and Implementation of Load Balancing in Web Server Cluster System, Journal of Nanjing University of Aeronautics & Astronautics, Vol.38 No. 3 Jun. 2006.

[20] R.L. Graham. Bounds for certain multiprocessing anomalies. SIAM J. Applied Mathematics 17 (1969), 263-269.

[21] Amazon, Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/, 2011.