A DYNAMIC AND INTEGRATED LOAD-BALANCING SCHEDULING ALGORITHM FOR CLOUD DATACENTERS

Wenhong Tian, Yong Zhao, Yuanliang Zhong, Minxian Xu, Chen Jing

University of Electronic Science and Technology of China, Chengdu, China
tian_wenhong@uestc.edu.cn

Abstract

One of the challenging scheduling problems in Cloud datacenters is to take the allocation and migration of reconfigurable virtual machines into consideration as well as the integrated features of hosting physical machines. We introduce a dynamic and integrated resource scheduling algorithm (DAIRS) for Cloud datacenters. Unlike traditional load-balance scheduling algorithms which consider only one factor such as the CPU load in physical servers, DAIRS treats CPU, memory and network bandwidth integrated for both physical machines and virtual machines. We develop integrated measurement for the total imbalance level of a Cloud datacenter as well as the average imbalance level of each server. Simulation results show that DAIRS has good performance with regard to total imbalance level, average imbalance level of each server, as well as overall running time.

Keywords: Cloud computing; dynamic and integrated resource scheduling algorithms; Cloud datacenter

1 Introduction

A Cloud datacenter can be a distributed network in structure, which is composed of many compute nodes (such as servers), storage nodes, and network devices. Each node is formed by a series of resources such as CPU, memory and so on. Each resource has its corresponding properties. There are many different types of resources for Cloud providers. This paper focuses on the Infrastructure as a Service (IaaS) level. The definition and model defined in this paper aim to be general enough to be applicable to a variety of Cloud providers.

In a traditional data center, applications are tied to specific physical servers that are often over-provisioned to deal with workload surges and unexpected failures [2]. Such configuration rigidity makes data centers expensive to maintain with wasted energy and floor space, low resource utilization and significant management overheads.

With virtualization technology, today's Cloud datacenters become more flexible, secure and on-demand allocating. With virtualization, Cloud datacenters should have ability to migrate an application from one set of resources to another in a non-disruptive manner. Such agility becomes a key in modern cloud computing infrastructures that aim to efficiently share and manage extremely large data centers. One key technology plays an important role in Cloud datacenters is resource scheduling.

There are quite many research conducted in the area of scheduling algorithms. Most of them focus on load balancing for traditional Web servers or server farms. One of the challenging scheduling problems in Cloud datacenters is to take into consideration both the allocation and migration of reconfigurable virtual machines, and the integrated features of hosting physical machines. Unlike traditional load-balancing scheduling algorithms which consider only physical servers with one factor such as CPU load, DAIRS treats CPU, memory and network bandwidth integrated for both physical machines (PMs) and virtual machines (VMs). We develop integrated measurement for the total imbalance level of a Cloud datacenter as well as the average imbalance level of each server. Buyya et al. [1] introduced a way to model and simulate Cloud computing environments, in which a few simple scheduling algorithms such as time-shared and space-shared were discussed and compared. [5] introduced three general scheduling algorithms for Cloud computing and some results were provided. 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. [2] proposed a novel load balancing algorithm called VectorDot for handling the hierarchical and multi-dimensional resource constraints by considering both servers and storage in a Cloud. Tian et al. [3] provided a comparative study of major existing scheduling strategies and algorithms for Cloud datacenters.

978-1-61284-204-2/11/$26.00 ©2011 IEEE

In this paper, we introduce a dynamic and integrated resource scheduling algorithm (DAIRS), which treats CPU, memory and network bandwidth integrated for both physical machines and virtual machines. We develop integrated measurement of the total imbalance level of a Cloud datacenter as well as the average imbalance level of each server for performance evaluation. The organization of this paper is as follows: section 2 ntroduces measurements of integrated load-balancing scheduling algorithms, section 3 esents DAIRS algorithm in details, section 4 discusses simulation results by comparing a few different scheduling algorithms, and finally a conclusion is provided in section 5.

2 Measurement of integrated load-balancing scheduling algorithm

Wood et al. [6] introduced a few virtual machine migration techniques. One integrated load balance measurement is applied as follows:

$$V = \frac{1}{(1-CPU_u)(1-MEM_u)(1-NET_u)} \tag{1}$$

where CPU_u, MEM_u, $NEYT_u$ are average utilization of CPU, memory, network bandwidth during each observed period, respectively. The larger the value V is, the higher the integrated utilization. This actually is a strategy of minimizing integrated resource utilization. Allocation and migration algorithms therefore can be based on this measurement.

Zheng et al. [8] proposed another integrated load-balancing measurement as following:

$$B = \frac{aN1_i C_i}{N1_m C_m} + \frac{bN2_i M_i}{N2_m M_m} + \frac{cN3_i D_i}{N3_m D_m} + \frac{dN1_i Net_i}{Net_m} \tag{2}$$

The referred physical server m is selected firstly. Then each of the other physical servers i is compared to server m. $N1_i$ is the CPU capability, $N2_i$ is for memory capability, $N3_i$ is for hard disk. C_i, M_i is for average utilization of CPU and memory respectively, D_i is for transferring rate of hard disk, Net_i is for network throughput. a, b, c d are for weighting factor of CPU, memory, hard disk and network bandwidth respectively. The major idea of this algorithm is to choose the smallest value B among all physical servers to allocate virtual machines.

Singh et al. [2] introduced a novel VectorDot algorithm to consider integrating factors of load balance for flow paths in datacenters. For a server node, node fraction vector $< \frac{cpuU}{cpuCap}, \frac{memU}{memCap}, \frac{netU}{netCap}, \frac{ioU}{ioCap} >$ is defined, where $cpuU$, $memU$, $netU$ is average utilization of CPU, memory and network bandwidth of a server respectively, $cpuCap$, $memCap$, $netCap$ is the total capacity of CPU, memory and network bandwidth of a server respectively. And node utilization threshold vector is given by $< cpuT, memT, netT, ioT >$, where $cpuT$, $memT$, $netT$, ioT represents utilization threshold of CPU, memory, network bandwidth and IO respectively. To measure the degree of overload of a node, and of the system, the notion of an imbalance score is used. The imbalance score for a node is given by

$$IBscore(f,t) = \begin{cases} 0, if \ \ f < T, \\ e^{(f-T)/T}, otherwise \end{cases} \tag{3}$$

By summing up imbalance scores of all nodes, the total imbalance score of the system is obtained. This nonlinear measurement has advantage of distinguishing a pair of nodes at $3T$ and T from a pair of nodes both at $2T$. The imbalance score is a good measurement for comparing average utilization to its threshold.

For DAIRS algorithm, the following parameters are considered:

(1). average CPU utilization CPU_i^U of a single server i : is averaged CPU utilization during observed period. For example, if the observing period is one minute and CPU utilization is recorded every 10 seconds, then CPU_i^u is the average of six recorded values of server i.

(2). average utilization of all CPUs in a Cloud datacenter. Let CPU_i^n be the total number of CPUs of server i,

$$CPU_u^A = \frac{\sum_i^N (CPU_i^U) CPU_i^n}{\sum_i^N CPU_i^n} \tag{4}$$

where N is the total number of physical servers in a Cloud datacenter. Similarly, average utilization of memory, network bandwidth of server i, all memories and all network bandwidth in a Cloud datacenter can be defined as MEM_i^U, NET_i^U, MEM_u^A, NET_u^A respectively.

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

$$\frac{(Avg_i - CPU_u^A)^2 + (Avg_i - MEM_u^A)^2 + (Avg_i - NET_u^A)^2}{3} \tag{5}$$

where

$$Avg_i = (CPU_i^U + MEM_i^U + NET_i^U)/3 \tag{6}$$

(ILB_i) is applied to indicate load imbalance level comparing utilization of CPU, memory and network bandwidth of a single server itself.

(4). the imbalance value of all CPUs, memories and network bandwidth. Using absolute deviation, the imbalance value of all CPUs in a data center is defined as

$$IBL_{cpu} = \sum \left| CPU_i^U - CPU_u^A \right| \tag{7}$$

Similarly, imbalance values of memory and network bandwidth can be calculated. Then total imbalance values of all servers in a Cloud datacenter are given by

$$IBL_{tot} = \sum_{i}^{N} ILB_i \tag{8}$$

(5). average imbalance value of a physical server i. The average imbalance value of a physical server i is defined as

$$IBL_{avg}^{PM} = \frac{IBL_{tot}}{N} \tag{9}$$

where N is the total number of servers. As its name suggests, this value is used to measure imbalance level of all physical servers.

(6). average imbalance value of a Cloud datacenter (CDC). The average imbalance value of a Cloud datacenter (CDC) is defined as

$$IBL_{avg}^{CDC} = \frac{IBL_{cpu} + IBL_{mem} + IBL_{net}}{N} \tag{10}$$

3 DAIRS algorithm

3.1 Resources considered in this paper

Resource considered in this paper includes:

(1) Physical server: physical computing devices which forms a datacenter, each physical server can provide multiple virtual servers; each physical server can be multiple composition of CPU, memory, hard drives, network cards, etc.).

(2) Physical clusters: consists of a number of physical servers, necessary network and storage infrastructure.

(3) Virtual server: is a virtual computing platform on the physical server by using virtualization software, it has a number of virtual devices (CPU, hard drives, network cards, etc).

(4) Virtual cluster: consists of a number of virtual servers and necessary network and storage infrastructure.

3.2 Scheduling process in Cloud datacenter

Figure 1 provides a referred architecture of Cloud datacenters and major operations of resource scheduling:

(1) User requests: the user initiates the request through the Internet (such as login Cloud service provider's Web portal);

(2) Scheduling management: scheduler center makes decisions based on the user's identity (such as geographic location, etc.) and the operational characteristics of the request (quantity and quality requirements), the request is submitted to the appropriate data center, then the data center management program submit it to scheduler center, the scheduler center allocates the request based on scheduling algorithms applied in Cloud datacenter;

(3) Feedback: scheduling algorithm provide available resources to the user;

(4) Execute scheduling: scheduling results (such as deploying steps) are sent to next stage;

(5) Updating and optimization: scheduler updates resource information, optimizes resources among different data centers according to the objective functions.

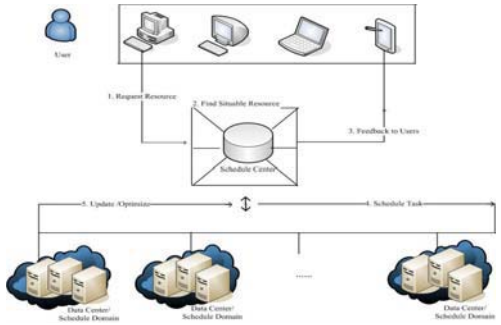

Figure 1 Referred architecture of Cloud Datacenters

3.3. Steps of DAIRS algorithm

Fig.2 shows the flow diagram of DAIRS algorithm. Four kinds of queues are defined: waiting queues using for those requests are not allocated immediately but have to wait; requesting queue is for new requests, optimizing queue is for tasks which need reallocation while deleting queue keeps tasks if its ending time is due.

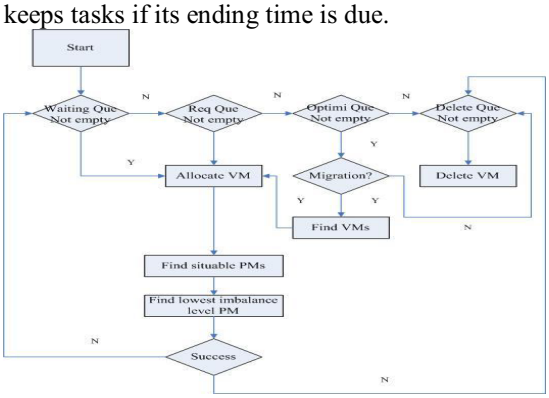

Figure 2 Flow chart of DAIRS algorithm

Major steps of DAIRS algorithm are as follows:

Step 1: obtain utilization information of current observation period, including [cpu utilization, memory utilization, network utilization].

Step 2: accept new task request (from highest to lowest priority queue are waiting, requesting, optimizing, deleting queues).

Step 3: check the waiting queue, if it is not empty, remove element from waiting queue, following specific allocation process step 7 and 8. If the allocation fails, then go to step 9. If the allocation is successful, return to step 1.

Step 4: check requesting queue, if it is not empty and start time of the task is due, then begin allocation by following allocation algorithm for the

specific steps 7-8. If successful, put the request to deleting queue.
Step 5: check optimizing queue, if it is not empty, go to step 10.
Step 6: check deleting queue, delete the task if its ending time is due.
Step 7: allocation algorithm (can be other algorithms): first of all, based on the user request type of virtual machines (cpu, memory, network bandwidth), algorithm sorts physical servers in ascending order of utilization of this type. Then divides utilization of physical servers into multiple intervals, the size of each interval can be set (for example 0.10) and finds all physical servers within that interval.
Step 8: select the lowest utilization interval (for example (0,0.10)) of physical servers to start allocating virtual machine, as long as the allocation of the virtual machine does not exceed the maximum capacity of the physical server. if the interval can not be assigned for all physical servers, then try next interval until allocation is carried out.
Step 9: if all the physical servers cannot allocate, then the task is added to the waiting queue in a first-come-first-service fashion.
Step 10: migration process: according to preset utilization thresholds of cpu, memory, network bandwidth, algorithm finds physical server with highest load, and virtual machine with lowest load to migrate using allocation algorithm. Continues migration until utilization of the physical server is under threshold, or there is a repeat of migration of the same virtual machine.
Step 11: after checking all the queues, go back to step 1.
Figure 3 shows main class diagram of DAIRS algorithms.

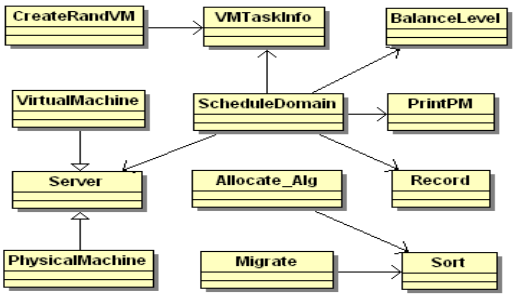

Figure 3 Main class diagram of DAIRS

4 Simulation results

In this section, we provide simulation results for comparing four different scheduling algorithms introduced in this paper. Simulation results are obtained using simulation tools designed in [4]. All results are collected using a Pentum Windows PC with 1.0 GHz CPU, 2G memory. For convenience, short name is given for each algorithm as follows:

1). ZHCJ algorithm: as introduced in [6], the algorithm always chooses physical machines with lowest V value (as defined in equation (1)) and available resource to allocate virtual machines.
2). ZHJZ algorithm: selects a referring physical machine [8], and calculates the value and chooses physical machines with lowest B value (as defined in equation (2)) and available resource to allocate virtual machines.
3). MZXQ algorithm (DAIRS algorithm): based on demands characteristics (for example, CPU intensive, high memory, high bandwidth requirements etc.), always selects physical machines with lowest integrated imbalance value (as defined in equation (5)) and available resource to allocate virtual machines.
4). Rand algorithm: randomly assigns requests (virtual machines) to physical machines which have available resource.
For simulation, three types of heterogeneous physical machines are considered (can be dynamic configured and extended, mem for memory, bd for bandwidth):
type1:cpu=64GHz,mem=120G, bd= 200M
type2:cpu=96GHz, mem=180G, bd= 300 M
type 3 cpu128GHz, mem=240G, bd =400 M.
And eight types of virtual machines with equal probability of requests are generated as follows (can be dynamic configured):
Type1:cpu=2.0GHz, mem=1.0G, bd=2.0M
Type2:cpu=10.0GHz, mem=4.0G, bd=8.0M
Type3:cpu=16.0GHz,mem=12.0G, bd=15.0M
Type4:cpu=3.0GHz, mem=9.0G, bd=5.0M
Type5:cpu=6.0GHz, mem=20.0G, bd=15.0M
Type6:cpu=13.0GHz, mem=36.0G, bd=25.0M
type7:cpu=1.0GHz, mem=1.0G, bd=25.0M
type8:cpu=2.0GHz, mem=4.0G, bd=50.0M.
For all the simulation, the number of physical servers is ranging from 100 to 800, the number of requests of virtual machines is varying from 1000 to 8000, a Pentium PC with 2 GHz CPU, 2G memory is used for all simulation.

Fig.4 shows average imbalance level (defined in equation (10)) of a Cloud datacenter. It can be seen that DAIRS algorithm (MZXQ) has lowest average imbalance level of a Cloud datacenter.

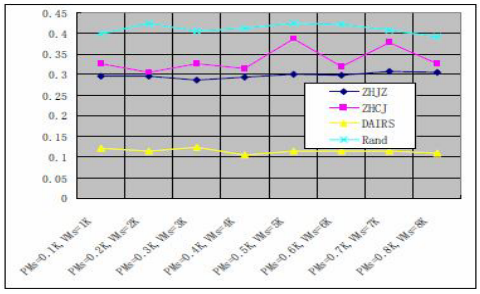

Figure 4 Average imbalance values of a Cloud datacenter

Figure 5 shows average imbalance level of all physical servers (defined in equation (5)). DAIRS algorithm has lowest average imbalance level

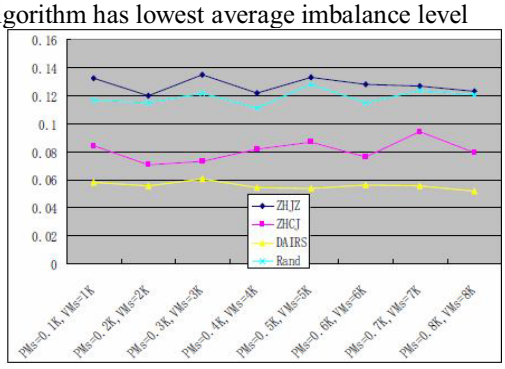

Figure 5 Average imbalance values of each physical server

Figure 6 shows overall running time of four different algorithms, DAIRS algorithm has running time just next to the shortest running time Rand algorithm. But Rand algorithm has higher average imbalance level of all physical servers and average imbalance level of each physical server.

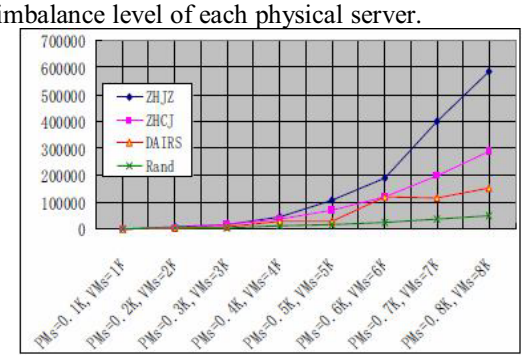

Figure 6 Running time comparison of four algorithms (in milliseconds)

Through extensive simulation, similar results are observed. Because of page limitation, some more simulation results such as varying the probability of each virtual machine request and fixing total number of physical servers but varying number of virtual machines are not provided here.

5 Conclusions

In this paper, we introduce a dynamic and integrated resource scheduling algorithm (DAIRS) for Cloud datacenters. Simulation results show that DAIRS algorithm has good features regarding of total imbalance level of all servers and average imbalance level of each server as well as of running time. In the near future, we will develop more indices to measure the quality of DAIRS and related algorithms. Also some new algorithms such as considering energy-efficiency for Cloud datacenters are under research.

Acknowledgements

The first author Dr. Tian’s research is jointly sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry（2010-2011），and Chinese Post-doc Research Foundation, State Education Ministry（2011-2012）.

References

[1] R. BUYYA, R. RANJAN AND R. N. CALHEIROS, Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities, Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS 2009, ISBN: 978-1-4244-4907-1, IEEE Press, New York, USA), Leipzig, Germany, June 21 - 24, 2009.

[2] A. SINGH, M. KORUPOLU, D. MOHAPATRA, Server-Storage Virtualization: Integration and Load Balancing in Data Centers, in the proceedings of the 2008 ACM/IEEE conference on Supercomputing (2008), pp. 1-12.

[3] 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.

[4] W. TIAN, ZHONG, Y., DONG, X., A lightweight simulation system for resource scheduling in Cloud datacenter, in the proceedings of the IEEE 3rd International Conference on Networks Security Wireless Communications and Trusted Computing. 2011.

[5] BWICKREMASINGHE 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) 2000102353.

[8] 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.