Research on RMOLB Algorithms for Cloud Load Scheduling

Wei Guanghui

Chongqing College of Electronic Engineering, Chongqing, 401331 China

Keywords: Cloud computing, Load, RMOLB, Research

Abstract: An online load balancing resource scheduling algorithm (RMOLB) considering real-time and multi-dimensional resources is proposed. The algorithm is designed for real-time load balancing considering dynamic allocation of virtual machines in data centers. It is easy to operate and has good performance. The theoretical analysis proves that the algorithm has a competition ratio of 2-1/m, where m is the total number of physical machines. At the same time, through a large number of simulation analysis, it is found that RMOLB algorithm has certain advantages over other known algorithms in the key indicators of makespan (total completion time), capacity makespan (sum of capacity-completion time), imbalance degree and so on.

1. Introduction

Up to now, a lot of work has been done on scheduling algorithms and some achievements have been achieved. They can be roughly divided into two categories: online load balancing algorithm and offline load balancing algorithm. The main difference between them is that the online algorithm can only get the status of the current request, while the offline algorithm can get the status of all requests during the entire scheduling time[1]. Singh proposed a new load balancing algorithm called Vectordot, which can be used to handle hierarchical and multidimensional resources restricted by servers and storage[2]. Armbrust summarizes key issues and solutions in cloud computing. Foster provides a detailed comparison between cloud computing and grid computing. Andre discussed the design details of the data center[3]. Buyya introduces a way to model and simulate cloud computing environments. Wickremasinghe introduced the technology of virtual machine migration and proposed some migration algorithms. Zhan compares the main load balancing scheduling algorithms on traditional network servers[4]. Arzuaga proposes a method to measure the load imbalance of virtualized enterprise servers. Tian provides a comparative study of the existing main scheduling algorithms and strategies for data centers. Sun proposed a new empirical algorithm for obtaining approximate optimal solutions based on integrated resource scheduling[5]. Tian introduces a dynamic load scheduling algorithm that only considers the current allocation time and multi-dimensional resources, but does not consider the life cycle[6]. Li proposed a cloud task scheduling strategy based on ant colony optimization algorithm, which can be used to balance the load of the whole system and minimize makespan of a series of tasks[7]. Galloway introduced an online greedy algorithm to enable physical machines to dynamically open and close, but did not consider the life cycle of virtual machines[8].

2. Model establishment

Suppose there are six virtual machine requests (VM1 ~ VM6), each with different start time, end time and capacity requirements. For example, the start time, end time and capacity requirement of VM1 are 1, 6 and 0.25 respectively, which indicates that VM1 needs a fixed processing time from the beginning of the first slot to the end of the sixth slot, and the capacity requirement in this processing time is 0.25. Other virtual machine requests follow this definition. Using some traditional metrics such as makespan, load efficiency (load efficiency, also known as skew in some literatures), as well as the unbalance value proposed by ourselves (introduced in the next section), the measurement results in different scheduling algorithms reflect different performance. Our goal is to reduce makespan, unbalance values, and improve load efficiency.

We modeled the allocation of virtual machines as an improved interval scheduling problem (MISP), and assumed that the processing time of each virtual machine request was fixed. More explanations and analysis of the traditional interval scheduling problem with fixed processing time can be found in the literature. We propose an improved interval scheduling problem and compare it with some existing algorithms. If a series of requests are 1, 2,... N, then the start time corresponding to the first request is Si and the end time is fi; the capacity requirement is ci.

Since each request requires only a portion of the capacity on a physical machine, we redefine makespan as capacity-makespan. Capacity - makespan (CM): In the process of allocating VM requests to PM i, we use A (i) to represent the set of VM requests allocated to PM i, J is the ID of the virtual machine with the largest load in A (i), and the maximum load of the first virtual machine is represented by Li, as shown in Public (1).

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

Among them, C is the CPU request of VM, and t is the execution time of request J (the processing time of request f). The goal of load balancing is to minimize the maximum load of all PMs.

3. Resource scheduling algorithm

Zheng et al. introduced a comprehensive load balancing index and a load balancing algorithm, as shown in bulletin (2).

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

Among them, I is the number of a PM, m is the ID of the selected reference physical machine, N1 is the CPU performance, N2 is the memory parameter, N3 is the bandwidth parameter, C and M are the CPU and memory utilization, D is the transmission rate of hard disk, Net is the network throughput, a, b, c, D are the relative weighted values of CPU, memory, hard disk and network, initial value It is 1. The optimization goal is to find PM with the minimum B value to allocate requests.

The average CPU utilization of PM over a period of time is shown in formula (3):

$$PCPU_i^u = \frac{\sum_{k=0}^{n} (PCPU_i^{T_k} \times T_k)}{\sum_{k=0}^{n} T_k} \tag{3}$$

$PCPU_i^{T_k}$ is the average utilization rate of CPUi in slot T_k. PM memory utilization $Pmen_i^U$ and storage utilization $PStorage_i^T$ can be calculated in the same way. Similarly, the average CPU utilization of VM can be calculated in this way.

PMi's comprehensive load imbalance ILBi; in statistics, variance is widely used to assess the degree of data dispersion. Variance can be used to define ILBi, the comprehensive load imbalance of server i, as shown in formula (4):

$$LIB_i = \frac{(Avg_i - CPU_u^A)^2 + (Avg_i - Mem_u^A)^2 + (Avg_i - Storage_u^A)^2}{3} \tag{4}$$

CPU_u^A, Mem_u^A, $Storage_u^A$ are the average utilization rates of CPU, memory and storage in data center, respectively. ILBi represents the load imbalance of CPU, memory and network bandwidth utilization of a computer. This measure is very similar to VMware's DRS load balancing measure.

Skew refers to the ratio of the minimum average load to the maximum average load of all computers, as shown in formula (5):

$$skew(makespan) = \frac{\min_i(Load_i)}{\max_i(Load_i)} \tag{5}$$

Skew can represent the efficiency of load balancing to some extent.

The definition of CPU imbalance is shown in formula (6):

$$IBLcpu = \frac{\sum_{i=0}^{n}(PCPU_i^U - PCPU_{avg})^2}{n} \tag{6}$$

$PCPU_{avg}$ represents the average utilization of all CPU in a data center. Memory imbalance IBL_{mem} and storage imbalance $IBL_{storage}$ can be obtained in the same way.

Skew of capacity-makespan is defined as minimum capacity-makespan/maximum capacity-makespan:

$$\text{skew(capacity - makespan)} = \frac{\min \sum_{j \in A(i)} c_j t_j}{\max \sum_{j \in A(i)} c_j t_j} \tag{7}$$

Generally speaking, the larger the value, the better the load balancing effect. From the above formulas, we can see that life cycle and capacity sharing are two main differences from traditional measurement indicators, such as makespan and skew. Traditional List Scheduling is widely used in load balancing of online multiprocessor scheduling. Considering the fixed processing time interval and capacity sharing attributes of data centers, we propose a new online algorithm RMOLB. For each request, the algorithm first finds the PM with the minimum average capacity-makespan, if the PM does not exist. With sufficient resources, requests are allocated to PM with sub-small average capacity-makespan, and so on, and sufficient physical machines are provided to ensure that all requests are not rejected; using priority queue data structure, the computational complexity of RMOLB algorithm is O (nlog2m), where n is the number of VM requests, m is required. The number of PM.

Therefore, by using priority queues or related data structures, the algorithm can find the PM with the smallest average capacity - makespan in O (log2m) time. The upper limit of RMOLB algorithm is determined. For RMOLB, according to its allocation mode, the first (m-1)*g requests will be equally allocated to m computers (assuming (m-1)*g can divide m), and the last request will be allocated to the computer with the smallest capacity-makespan.

4. Comparison of RMOLB algorithm with other algorithms

The reason why RMOLB algorithm has better performance than random algorithm and rotation algorithm is easy to understand, because random algorithm and rotation algorithm do not take other metrics such as makespan, Skew and load balance into account. The performance of RMOLB algorithm is better than ZHJZ algorithm (benchmark) and LS algorithm because RMOLB algorithm constantly updates formula (2) in the lifecycle of virtual machine and physical machine to obtain more accurate state of virtual machine requests. However, ZHJZ algorithm (benchmark) and LS algorithm only consider the current allocation time to determine the allocation, but they do not. Consider the lifecycle of physical machines and virtual machines. The upper limit of approximation of RMOLB algorithm is also explained in the proof of algorithm approximation. In current applications, slot attributes related to requests should be measured by some more appropriate and accurate metrics. For example, the capacity-makespan index with time slot is better than the traditional index without time span.

5. Summary

In this paper, we propose a real-time and multi-dimensional online load balancing resource scheduling (RMOLB) algorithm to solve the real-time multi-dimensional resource scheduling problem in data centers. The simulation results show that RMOLB algorithm has better performance than some existing algorithms in terms of unbalance, makespan capacity makespan, skew of capacity-makespan and so on.

References

[1] L. Andre, et al. The Datacenter as a Computer: An Introduction to the Design ofwarehouse-scale Machines Ebook, 2009

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

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

[4] R. Buyya, R. Ranjan and R. N. Calheiros Modeling and Simulation of Scalable Cloud Computing Environments and the Cloud Sim Toolkit: Challenges and Opportunities Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS2009. ISBN: 978-1-4244-4907-1, IEEE Press, New York, USA), Leipzig, Germany, June1-24.20

[5] R L. Graham. Bounds on Multiprocessing Timing Anomalies. SIAM Journal on Applied Mathematics, Vol 17, No. 2.(Mar, 1969), pp 416-429

[6] A Gulati, G. Shanmuganathan, A Holler, I. Ahmad cloud-scale resource management challenges and techniques. Vmware Technical Journal, 2011

[7] 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, pp 89-96, 18-20 Dec2010

[8] K. Li, G. Xu, G. Zhao, et al. Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization chinagrid, pp3-9, 2011 Sixth Annual China Grid Con ference, 2011.