AN IMPROVED K-SUBSET ALGORITHM FOR LOAD BALANCE PROBLEMS IN CLOUD COMPUTING

Linlin Tang, Pingfei Ren, Jengshyang Pan

School of Computer Science and Technology,
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
hittang@126.com

Abstract: Cloud Computing has earned more and more attention for its important role in the modern network development process. It can be seen as a new distributed computing mode based on virtualization process. One of the problems for Cloud Computing is load balance. An improved k-subset algorithm is proposed in this paper for this problem. Good performance in experiments on CloudSim platform shows its efficiency

Keywords: Cloud Computing; Load balance; k-subset algorithms; CloudSim.

1 Introduction

Cloud Computing has become more and more popular in network. Generally speaking, it can be seen developed from grid computing with some characteristics of utility computing. It also has basic feature of virtualization [1].

Cloud Computing could can both benefit its users and providers[2]: For those users, a large number of money buying IT resources could be saved by using services provided by the cloud services providers[3]. For those cloud services providers, money could be earned by using its machines' abilities fully. Here the word "users" could be an end user, an organization or even another cloud services provider [4].

There are two main opinions on cloud computing [5]: concreted or distributed. From user's view their tasks are submitted to a "cloud", and then the "cloud" returns the result, just like days in middle 20thcentury. On the other hand, tasks may be divided into several parts and then processed by one or more real machines, just like distributed computing. In fact, tasks users submitted is processed by many virtual machines installed in one or more real machines, in another word, in a distributed method through virtualization.

Both the way cloud serve users and the types of services affect the way to make load balance decisions. Here we give some basic type of services [6]:

Infrastructure as a service (IaaS)

IaaS providers provide Process ability, Storage, network and other fundamental computing ability as services, hidden details of real machines [7]. Users could deploy and control all the things in virtual machines except the real hardware where virtual machines are deployed in.

IaaS providers are: EC2, HP Cloud, Azure Service as a service, etc.

Platform as s service (PaaS)

PaaS providers provide a kind of platform where OS, DB, Runtime Environment, web server are already given. Users could only deploy and control their own software and the runtime needed.

PaaS providers are: Google App Engine, Windows Azure Services, AWS Elastic Beanstalk, etc.

Software as a service (SaaS)

SaaS providers provide software for users. Here users are often referred to as end users, in another word, people. Users relief themselves from installing software, resulting in more hard disk space and less load for their CPU. By using SaaS, device inventors could make devices smaller faster while have more functionality.

People usually give particular names for some kind of SaaS by its function: Desktop as a service, Business Process as a Service, Communication as a service, etc.

SaaS providers are: Google Apps, Microsoft Office 365, onlive, GT Nexus, etc.

Others

The aforementioned three types of cloud services are defined by NIST, Nov, 2011. Other organizations have several different opinions about it. In Feb, 2012, ITU-T raised network as a service (NaaS), which could be seen as the same level where IaaS, PaaS and SaaS at. NaaS is determined by network provider, network user and what kind of service it provides; take VPN, BoD and MVNOs for example.

Also ITU-T separates CaaS from SaaS from the view of its real time attribute.

In this paper we mainly focus on the CPU-bound tasks, ignoring other attributes of the task such as I/O and bandwidth needed.

2 Related work

Load balancing problem is studied deeply in traditional distributed systems. Aim of load balancing is to avoid some particular processing nodes have heavy load while others have almost zero load, in another word, to avoid imbalance. Load balancing try to optimize system performance within its ability while overload control mainly focus when and how to determine refusing new requests, where in cloud computing context it mainly

978-1-4799-4719-5 /14/$31.00 ©2014 IEEE

focus on when, where and how to start more virtual machines.

In traditional distributed systems load balancing action could be performed at several levels: In network level, we can use DNS redirection, NAT, etc. In OS level, we can use load sharing, cluster, process migration, etc. In application level, load balancing action is performed by application themselves.

2.1 Load Balancing and Its Algorithms

Load balancing problem has been studied deeply in traditional distributed systems. The aim is to avoid some particular processing nodes having heavy load while others have almost zero, in another words, it is used to avoid imbalance. Load balance techniques try to optimize system performance within its ability while overload control mainly focus when and how to determine refusing new requests, where in cloud computing context it mainly focus on when, where and how to start more virtual machines.

In distributed systems load balance can be achieved vairously: In network level, one can use DNS redirection, NAT, etc. In OS level, one can use load sharing, cluster, process migration, etc. In application level, load balancing action is performed by application themselves.

Basically a whole load balancing algorithm contains several aspects, such as job assignment, job immigration and overload detection [8]. In This paper we mainly focus on the job assignment part, thus task assignment method in this paper is equivalent to the load balance algorithm in concept. Based on the time when the assignment action is decided, a load balance algorithm could be static or dynamic. Static load balancing algorithm gains all the information at beginning [9], while dynamic algorithm gains the information needed at runtime. Shown as before, cloud computing system is a special kind of distributed system and is full of indeterminacy [10]. Dynamic load balancing algorithms are preferred in cloud computing context.

Some important results about load balancing has been got: Proved mathematically, finding an optimal load balancing algorithm in various conditions is an NP-Complete problem [11], meaning that we cannot find the best solution in polynomial time. Thus in reality it is acceptable to find a sub optimal algorithm at a reasonable price.

In fact, a good load balancing algorithm should also be simple, robust, stable and flexible.

2.2 k Subset Algorithm

K subset algorithm was first proposed by M. Mitzenmacher in 1996, writing in his dissertation called "the Power of Two Choices in Randomized Load Balancing"[12] for the degree of Ph.D. in UNVERSITY of CALIFORNIA at BERKELEY.

In this paper a supermarket model was introduced as shown in the following Figure 1, where he takes k=2 for example.

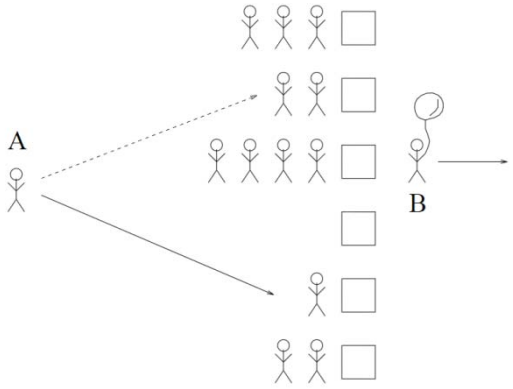

Figure 1 Supermarket Model

Generally speaking, K subset algorithm is a randomized load balance algorithm, in which:

1. *The consumer selects two queues randomly.*
2. *The consumer chooses the smaller queue and then waiting to be served.*
3. *Go to step 1, until there are no consumers making decision.*

In supermarket model, all the processing nodes are supposed equal. Thus the node chosen to be the candidates will maintain their mathematic relations (say<, >, and =) before and after an attempt to send the task to them all [13].

3 Proposed Method

In order to load balance among processing nodes, some elementary things should be considered: what we want to achieve, how we balance between the resolution and the complexity to get it[14].

In this paper aim is to minimize the total execution time. Thus we have following definitions:

TSet= $\{t_1, t_2, t_3 \ldots t_n\}$ where TSet denotes the total task to be executed by the cloud system and each of $t_1, t_2, t_3 \ldots t_n$ denotes a single task to be proceed by a virtual machine (VM) in the cloud system.

VSet= $\{VM_1, VM_2, VM_3 \ldots VM_k\}$ where VSet denotes all the virtual machines a cloud computing system while each of $VM_1, VM_2, VM_3 \ldots VM_n$ denotes a virtual machine exists in the cloud computing system.

$T_i = T_{i.assign} + T_{i.tranport} + T_{i.queue} + T_{i.proc} + T_{i.etc}$ Where T_i represents total time t_i spends in its life cycle. $T_{i.assign}$ is the time load balancing algorithm spend to decide where to put t_i. $T_{i.tranport}$ is the transportation delay to transport t_i to a VM.

$T_{i.queue}$ is the time spend in the waiting queue when the VM processing t_i.

$T_{i.proc}$ is the time VM spend to execute the process actually.

$T_{i.etc}$ is the time spend in other conditions, such as system crash, transportation miss, I/O delay.

To make things simple enough, here we let $T_i = T_{i.assign} + T_{i.queue} + T_{i.proc}$, that is, to ignore any failure condition and things about network.

$\mathrm{TotalTime} = \min(\sum T_i)$ where TotalTime is our target function.

Thus our goal could be described as: to find a mapping between TSet and VSet while achieving $\min(\sum T_i)$. Here TSet and VSet are undetermined.

Of course we can infer:

$$\mathrm{TotalTime} = \min(\sum T_i) = \min \sum (T_{i.assign} + T_{i.queue} + T_{i.proc})$$
$$= \min \sum (T_{i.assign}) + \min \sum (T_{i.queue}) + \min \sum (T_{i.proc})$$

The ideal situation is that we can find a mapping making $T_{i.assign}, T_{i.queue}, T_{i.proc}$ to its minimal value at all. Limited VMs make it impossible to meet the ideal situation. Actually making $T_{i.proc}$ smaller result in the increasing of $T_{i.assign}$ and $T_{i.queue}$. Same situation occurs for $T_{i.assign}$ and $T_{i.queue}$.

Thus actually the aim is to make a trade off among $T_{i.assign}$, $T_{i.queue}$ and $T_{i.proc}$.

Here we can describe the k subset algorithm in this way:

For t_i in TSet, choose a k subset of VSet randomly.

Choose the best performed VM_i among the k subset at that time and assign tasks to VM_i

If all tasks are assigned, exit. Or go to the next step.

Change i, repeat the above mentioned steps.

4 Improved K Subset Algorithm and Experimental Results

In cloud computing context, the abilities of processing nodes (VMs) could be quite different [3] while we can get more information needed due to the virtualization.

Thus K subset algorithm using directly in cloud computing contexts directly will lead to this problem: mathematic relations may be damaged, that is:

1. Suppose the two processing nodes chosen to be candidates for task Ti, named Pi and Pj, have load Wi and Wj respectively.
2. To make things simple, we make Wi<Wj before making any assignment decision.
3. Ti observes Wi<Wj and then was assigned to Pi while Wi>Wj may happen if the same task is assigned to both Pi and Pj;

If Wi>Wj happens in the attempting assignment, the system will step into a more imbalance status compared to assign the task to Pj.

Here we'll show a small improvement for k subset algorithm, that is: when making assignment decisions, assign the task to Pi where Pi is the smaller loaded processing node after.

The main idea is that we should always avoid the system from further imbalance.

The K subset algorithm has many advantages, such as: simplicity, stability and flexibility. we can also make the improved algorithm own these advantages. In fact, it is difficult to calculate the $T_{i.queue}$ and $T_{i.proc}$ directly. By using CloudSim [4] we can get $T_{i.queue} + T_{i.proc}$ for every task submitted to VMs. Then, we can compare the result between several basic algorithms in a centralized way and then we can get the following Figure2 which shows the comparison results with other methods. Here we use the static load balance algorithm max-min as a principle and make k=2.First we use 5 VMs.

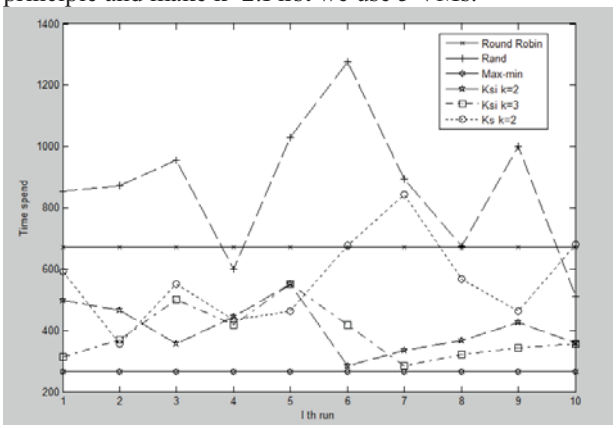

Figure 2 Comparison using 10 random generated tasks.

From Figure 2 we can observe that though ignoring $T_{i.assign}$ (means in the centralized way), the improved k subset algorithm still shows its high efficiency. The improved k subset algorithm even has a great chance to be close to the max-min method.

Figure 2 also shows that k=2 is enough when the total number of VMs is small.

To make it more clearly, we did the experiment several times. Here we use different tasks with the same VM configurations. Results are shown in Figure 3 and Figure 4.

Compared to Figure 2, Figure 3 and Figure 4 give us a more clear view about how the improvement contributes. Here tasks vary in a random generated method:

Shown as in Figure 3, with the number of tasks doubled, each method costs more time. However the improved k subset algorithm still maintains the ability to reach the bottom line. The simple random selection gets much worse with more tasks added in the job list. The reason is that dozens of tasks may be assigned to the same VM, making the whole system in a low efficiency condition. Here, getting enough candidates for selection is an ideal way to handle such situation. Just as shown in Figure 4, the improved k subset algorithm (k=2, 3) has its advantages.

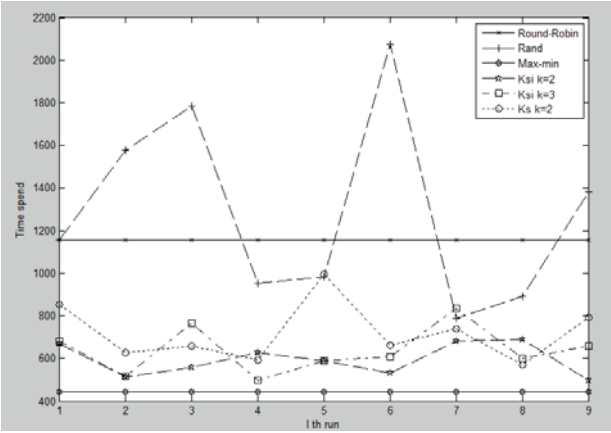

Figure 3 Comparison using 20 random generated tasks

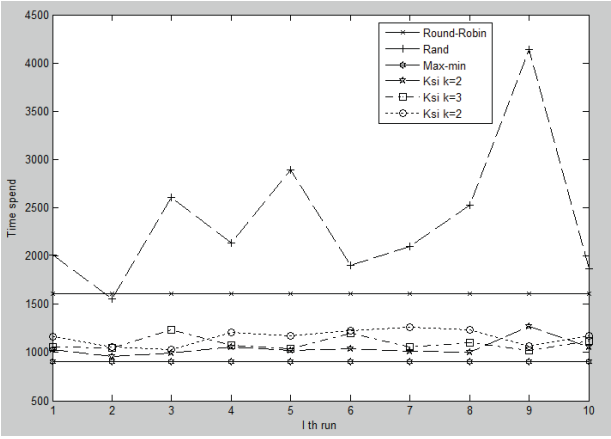

Figure 4 Comparison using 45 random generated tasks

Experiments above show that the improved k-subset algorithm is efficient when the number of tasks is large. The following experiments are done while VMs' capability are random generated. Tasks remain the same with those used in Figure 4. Results are illustrated in Figure 5 and 6.

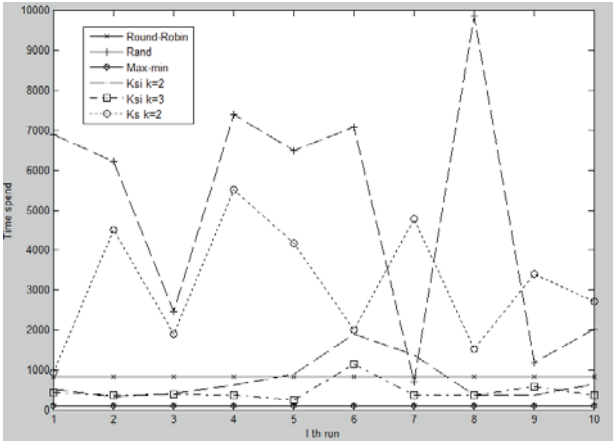

Figure 5 Comparison using 57 random generated VMs

Although we can always make VMs busy by making use of the characteristics of elastic in cloud, in extreme conditions, the number of VMs can be bigger than that of the tasks. Figure 5 illustrates such condition.

In Figure 5, both the Round-Robin and the Max-min methods work well because of the empty waiting queues. In contrast, other algorithms have a chance to make the queues much bigger than the above mentioned ones. Here the reason why k subset algorithm at k=3 works better relatively is that it has a bigger chance to get an good, empty VMs when k=3.

Figure 6 illustrates the most common situation with both VMs and tasks are generated randomly. The improved k subset algorithm shows its power clearly.

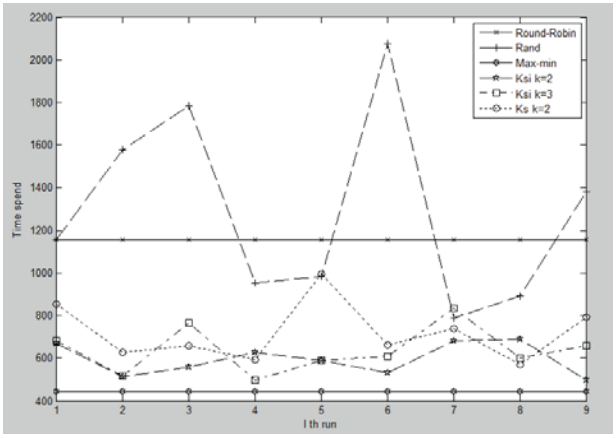

Figure 6 Comparison using 17 random generated VMs

Here we will also make some experiments to show its time and memory consumptions. We calculated the time and memory used several times under the same condition. Then we got their average value to make it more accurate. The configurations are the same with that of Figure6.

Firstly, we calculated the total time used with the number of tasks. Figure7 shows its result:

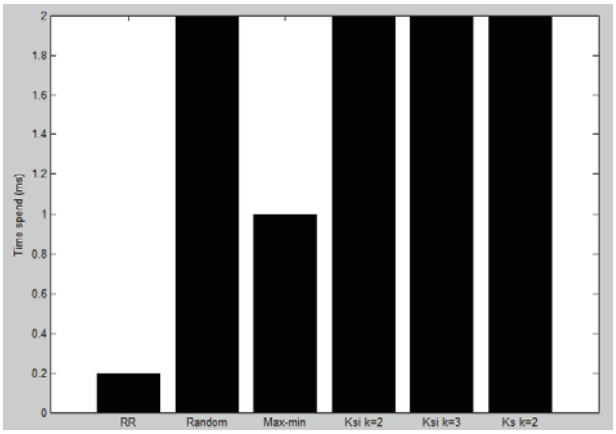

Figure 7 Time consumed for allocation

From Figure 7 we can infer that the improved k subset algorithm has its space in generating random number with less time consummation.

Secondly, we calculate the total memory used for each algorithm tested. We use jProfiler to monitor the java VM used in program. The results are shown in Figure 8:

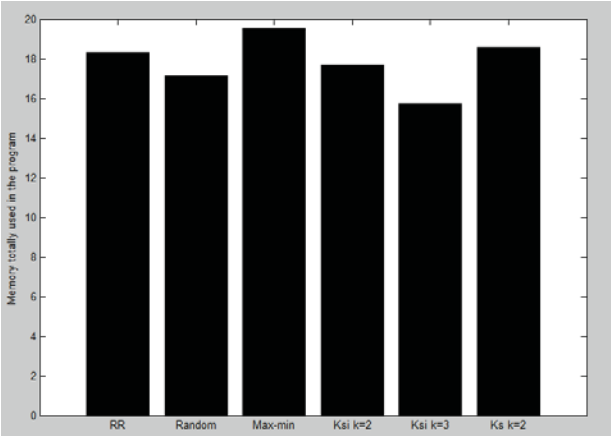

Figure 8 Memory consumed for the total program

Although not accurate, from Figure 8 we can get the information that the improved k subset algorithm works in a reasonable memory consummation mode.

5 Conclusions.

Generally speaking, a k subset algorithm and its improvement and the reason behind have been proposed in this paper. Experiment shows the effectiveness of both k subset algorithm and its improvement.

Acknowledgements

The authors would like to thank for the support from the project NSFC (National Natural Science Foundation of China) with the Grant number 61202456, the Harbin Institute of Technology Innovation Fund project and the NDRC's cloud computing project..

References

[1] Bao Rong Chang, Hsiu-Fen Tsai, and Chi-Ming Chen Evaluation of Virtual Machine Performance and Virtualized Consolidation Ratio in Cloud Computing System, Journal of Information Hiding and Multimedia Signal Processing (JIHMSP), Vol. 4, No. 3, pp. 192-200, July 2013.

[2] Azodolmolky S, Wieder P, Yahyapour R. Cloud computing networking: challenges and opportunities for innovations. Communications Magazine, IEEE, 2013, 51(7).

[3] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50-58.

[4] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 2011, 41(1): 23-50.

[5] Hwang K, Dungaree J J, Fox G C. Distributed and Cloud Computing. Elsevier/Morgan Kaufmann, 2012.

[6] Mell P, Grance T. The NIST definition of cloud computing. NIST special publication, 2011, 800: 145.

[7] Hongfeng Zhu, Tianhua Liu, Dan Zhu, and Haiyang Li Robust and Simple N-Party Entangled Authentication Cloud Storage Protocol Based on Secret Sharing Scheme, Journal of Information Hiding and Multimedia Signal Processing (JIHMSP), Vol. 4, No. 2, pp. 110-118, April 2013.

[8] Ray S, De Sarkar A. EXECUTION ANALYSIS OF LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING ENVIRONMENT. International Journal, 2012.

[9] Kathy Yelick. Load Balancing Part2: Static Load Balancing http://www.cs.berkeley.edu/~yelick/cs194f07/lectures/lect11-loadbalance.pdf.

[10] Xu G, Pang J, Fu X. A load balancing model based on cloud partitioning for the public cloud[J]. Tsinghua Science and Technology, 2013, 18(1): 34-39.

[11] Tayal S. Tasks scheduling optimization for the cloud computing systems. International Journal of Advanced Engineering Sciences And Technologies (IJAEST), 2011, 5(2): 111-115.

[12] Mitzenmacher M D. The power of two choices in randomized load balancing. University of California, 1996.

[13] Vvedenskaya N D, Dobrushin R L, Karpelevich F I. Queueing system with selection of the shortest of two queues: An asymptotic approach. Problemy Peredachi Informatsii, 1996, 32(1): 20-34

[14] Maguluri S T, Srikant R, Ying L. Heavy traffic optimal resource allocation algorithms for cloud computing clusters//Proceedings of the 24th International Teletraffic Congress. International Teletraffic Congress, 2012: 25