Resource virtualization under Batch Tasks Running in the Cloud Computing Environment*

Kuang Gui-juan[1,2,3] Zeng Guo-sun[1,2] Xiong huan-liang[1,2,4]

[1]Department of Computer Science and Technology, Tongji University, Shanghai, China
[2]Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai, China
[3]School of Science and Information Science, Qingdao Agricultural University, Qingdao, Shandong, China
[4]School of Software Jiangxi Agricultural University, Nanchang, China
1gjkuang@tongji.edu.cn, gszeng@tongji.edu.cn, xionghuanliang@126.com

Abstract - Nowadays, Virtualization technology is an important means in effective management of cloud computing resources. Focuses on dividable batch tasks running in the cloud computing environment, we study resource virtualization based on time-division multiplexing. Our methods make sure that users can use system resources equitably in cloud computing with multi-tenants, and allow limited computational resources to meet user requirements as more as possible, as well as improve the utilization of cloud resources.
Index Terms: cloud computing, batch tasks scheduling, resource management, time-division multiplexing

I. Introduction

Cloud computing has become a popular computing paradigm[0]. From a user's perspective, computing services cloud be used at anytime and at anywhere with SLA. From the system's perspective, it can maximize the throughput and utilization of the system, and be fair to all jobs regardless of their size or execution times by offering users on-demand with elastic, scalable high reliability and low cost computing resources [00]. To realize these advantages depends on effective management of the cloud resources.

Cloud computing resources management technology is based on traditional parallel and distributed computing technology among which the virtualization technology has become the key means for cloud computing system to realize on-demand resource management [000]. Previous researches related to Virtualization technology focus on the resource management in hardware virtualization[0], management in virtualized operation system, management in para-virtualization. Currently, cloud computing virtualization technology mainly combines the hardware virtualization with software virtualization to offer resources to different users in the form of VM (virtual machine) [0]. All these above virtualization technologies belongs the model of "large into small" which promote the utilization of physical resources, but increase the cost and complexity of resource management and scheduling. The other mode is "by the group", forthcoming multiple isolated physical resources into a more powerful server [0]. To some extent, the above virtualization technologies has achieved multi-user sharing and improved the utilization of resources. However, they are still far from meeting the requirements of on-demand services, there are still many problems: the waste of resources, the underutilization, multi-tenants can not use system resources equitably.

Noticed that recently in parallel computing system, there are a large number of dividable applications, such as Monte Carlo, Fractal calculation, parameter sweep, image processing, data mining and so on. This paper proposed new resource management methods based on time-division multiplexing in order to ensure users to use system resources equitably in cloud computing with multi-tenants, and allow limited computational resources to meet user requirements as more as possible, as well as improve the utilization of cloud resources.

2. Cloud Resource Virtualiztion

2.1 requirements from on-demand resource allocation in cloud computing

In cloud computing, the process of allocating resources is to arrange the computing resources to meet user's diverse requirements to achieve specific goals. For user, the goal is less computing time, less task waiting time etc. For provider, because Cloud computing is a computing model driven by scale economy, they will aim to improve the resource utilization and to offer more available resources to more users. What's more, they hope to provide users a fair experience in using resources. Figure 1 shows the process of on-demand resource allocation in cloud computing.

The physical computing resources $pr_1, pr_2, \ldots, pr_m$ most likely not be able to cope with a large number and variety of users' requirements $ur_1, ur_2, \ldots, ur_n$. Virtual technology are needed to organize the physical resources to form logic resources $lr_1, lr_2, \ldots, lr_m$ to meet users requirement.

* This work is partially supported by the National High-Tech Research and Development Plan of China under grant No. 2009AA012201; the National Natural Science Foundation of China under grant No. 61272107, No. 61202173, and No. 61103068; the joint of NSFC and Microsoft Asia Research under grant No. 60970155; the Program of Shanghai Subject Chief Scientist under grant No. 10XD1404400; the Ph.D. Programs Foundation of Ministry of Education under grant No. 20090072110035; the special Fund for Fast Sharing of Science Paper in Net Era by CSTD under grant No. 20110740001.

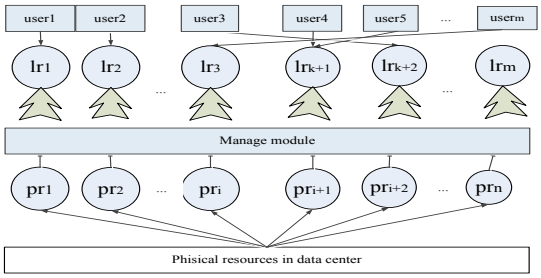

Fig. 1 On-demand resource allocation in cloud computing

2.2 The concepts of resource allocation

Definition 1 (physical resource)：Physical resource refers to the physical device which directly involved in computation process, also known as compute nodes. It is presented by $pr=(\lambda, \tau, v, c)$, Wherein, λ *is* the type of physical resource, τ is the clock interrupt cycle, which is the basic time unit for physical resource do a basic processing action, v is the processing speed, c is the context-switch time. The expression of all physical resource set in cloud system is $PRS=\{pr_1, pr_2, ..., pr_n\}$.

Definition 2 (user requirement)：User requirement is that user request cloud computing system to offer resources to perform the job according to SLA, also known as user task. It is presented by $ur=(\lambda, w, g, \tau)$, wherein, λ is the type of user requirement, w is workload, g is the dividable granularity of user's task, τ is the time required to complete the task. The expression of all user requirements set in cloud system is $URS=\{ur_1, ur_2, ..., ur_m\}$.

Definition 3 (logical resource)：Logical resource refers to the user's view of computational resource, which physical resources are reorganized with division, combination, reuse, also known as virtual resource. In this paper, it will be defined later in section 3.2.

3 Resource Time-division Multiplexing Methodology

3.1 Resource Time-division Multiplexing principle

For each computing resources, the entire running time axis is sliced into multiple small time slices, user tasks can use these time slices in turn. The process is as follows: a user task occupies only a time slice, and after all user tasks finish a time slice, a running cycle of the computing resource is end. That process will keep on until all user tasks are completed.

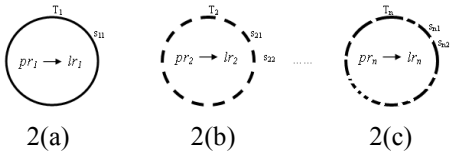

Fig. 2 Principle for physical resources time-division multiplexing

Figure 2(a) represents the resource is occupied by one task until it is finished. Figure 2(b) indicates that the resource is equally used by each task. Figure 2 (c) indicates that the each task use unequal time slice with in a running cycle. In this paper, the resource after be time-multiplexed is called logic resource.

3.2 Resource Time-division Multiplexing Model

Each computing node in the cloud computing environment has attributes running cycle T、time slice s_{ij} . T is also known as multiplexing cycle. T is divided into a plurality of small time slices i.e. s_{ij} denotes the running time that the ith computing nodes allocated to the jth user task. To reasonably effectively slice the running time axis of the computing nodes, many factors must be considered, including size of T, size of time slice s_{ij}, the number of time slices in a multiplexing-cycle, and the relationship between the user task ur and the slices s_{ij}.

Definition 3(multiplexed physical resource): multiplexed physical resource is a special kind of of virtual resources, also can be known as logic resource. After time-sharing multiplexed, logically, it can be used by a plurality of user tasks at the same time. It is represented by $lr=(\lambda, v, T, S)$, where λ is the type of physical resource, v is process speed of physical resource, T is running cycle of the multiplexed physical resource, $S=\{s_1, s_2, ..., s_m\}$ is the set of time slices, s_i is the i-th time slice, m is the number of time-slices. The set of the multiplexed physical resources in the system is $LRS=\{lr_1, lr_2, ..., lr_n\}$.

Based on the above analysis, formal description of resource time-multiplexing in cloud system is as follows:

$$\begin{cases} \exists T_i, s_{ij} : \{pr_1, pr_2, \ldots, pr_n\} \rightarrow \{lr_1, lr_2, \ldots, lr_n\} \\ T_i = \sum_{j=1}^{m} s_{ij} \\ ur_j.\tau = \sum_i s_{ij} \\ ur_j.\tau \le \theta_j, \theta_j \text{ is deadline of } ur_j \\ 1 \le i \le n, 1 \le j \le m \end{cases}$$

3.3 Appraisal indexes for Time-division Multiplexing

Definition 4 (makespan)：the makespan is $mst=\max_{1\le i\le m} ft_i - \min_{1\le i\le m} at_i$,where at_i is the moment that user task ur_i arrives the system, ft_i is the moment task finished.

Definition 5 (average response time)：is a average time to run a user task, presented by $art=\frac{1}{m}\sum_{i=1}^{m}(ft_i - at_i)$.

Definition 6 (average slowdown rate)：is a average value of the ratio of the task response time and the actual running time. presented by $asr=\frac{1}{m}\sum_{i=1}^{m}\frac{(ft_i - at_i)}{et_i}$,where et_i is the actual running time of i-th user task ur_i .

Definition 7 (Resource utilization)： It mean the ration of the effective work ability and total work ability , $ru=\frac{\sum_{i=1}^{m} et_i}{\sum_{i=1}^{m}(ft_i - at_i)}$ 。

4. batch task Time-division Multiplexing strategies

4.1 Conditions and assumptions

(1) We assumed that user requirements arrive in batches in cloud computing systems, denoted the moment when single batch tasks arrive as 0. Although users service requests may not be strictly batch arriving at the same time, we can still regard these tasks arrive at the same time by adjustment in the deadline of the task.

(2) We assume that the user requests is independent tasks which can be arbitrarily split.

(3) We assume that the number of batch tasks is *m*, the number of physical resource in cloud computing environment is *n* and all these resources are able to be time-multiplexed , any user task can be running on any physical resource , all the physical resources have been in running state until all user tasks are finished.

(4) We assume that the physical resources multiplexed switching overhead between users tasks are equal. Such assumption is In order to facilitate the analysis and discussion, and will not affect the validity of our methods. The task switching overhead is $c_0 = max_{1\le j\le n}\ pr_j.c$, *m* tasks switching overhead is c， $c=mc_0$.

4.2 Priority and exclusive strategy

Strategy 1(2): Minimum(Maximum) task is assigned firstly, and assigned to the resource with maximum speed: the idea of the strategy is that each time select a task and assign it to a certain resource, the resource is exclusive until the task is completed, shown in Figure 2 (a). The selection criteria are: 1) the task selected has the smallest(largest) workload (2) the selected resource has maximum process speed

The number of task assigned to pr_j is x_j ， x_j user requirements is represented with ur_{j1} ， ur_{j2} ， ... ， ur_{jxj} respectively. *We* use $mst^{(1)}$ represents the value of *mst* in strategy 1, and other variables are similar. Then We can get the performance indicators according to 3.3, we can get the performance indexes.

4.3 time-division multiplexing strategy

Strategy3: equal time slice and same amount of workload strategy

In this strategy, as shown in figure 2(b), the number of the time slices of each logic resource is the same, equal to the number of user tasks. Each logic resource allocate for each user task a same and fixed time slice, and handle the same size of workload in a time slice.

Let s_α be the fixed time slice. The size of s_α should meet: (1) $s_\alpha = k.lcm(\{pr_j.\tau | j=1,2,...,n\})$ ， where *k* is a const integer. (2) $\frac{s_\alpha}{c_0 + s_\alpha} \ge \rho$ ， where $c_0 = max_{1\le j\le n}\ pr_j.c$ ， ρ is a proportion of the time for the task runs in a time slice and the entire time slice. (3) $s_\alpha \le \frac{1}{n} \cdot \frac{min_{1\le i\le m}\ ur_i.w}{max_{1\le j\le n}\ pr_j.v}$.

Let w_α be the fixed size of workload. $w_\alpha = min_{1\le j\le n}(pr_j.v \times s_\alpha)$ ， thus each logical resource can have sufficient task workload to process within a time slice, and the workload handled for each task is equal.

According to the principle of time-division multiplexing, shown as in Figure 2(b) we can get the performance indexes.

In this strategy, small task time slice is wasted and faster resource will be idle in its time slice. In order to make full use of these time slices, need to further improve the time-division multiplexing strategy.

Strategy4: unequal time slice and different workload strategy

As shown in Figure 2 (c), the main idea of this strategy is: within each multiplexing cycle, give an unequal time slice to each user task and the time slice size is proportional to the workload of the task. The different tasks will be completed in almost the same time to obtain the fairness between users.

As for the same user task ur_i ， it's size of time slice on different resource is same, i.e, $s_{ij}=s_{ik}$. Let s_β be the time slice for the minimum user task. The method of determining s_β is exactly the same as the method of determining s_α in strategy 3, i.e. $s_\beta = s_\alpha$. In this strategy, the time slice for ur_i meets $s_{ij} = s_\beta \cdot ur_i.w / w_\beta, j = 1,2,...,n$ ， j=1, 2,…, n, where $w_\beta = \min_{1\le i\le m} ur_i.w$. The amount of workload performed in a time slice $w_{ij} = s_{ij} \cdot pr_j.v$. Thus, different computing nodes will perform different workload due to different speed.

Strategy5: type-matching-first integrated strategy

To make effective use of various multiplexing strategy in different cases, we proposed algorithm1.

Algorithm 1： type_matching_first_multiplexing()

Input: $URS=\{ur_1, ur_2, ..., ur_m\}$ $PRS=\{pr_1, pr_2, ..., pr_n\}$

Output: $LRS=\{lr_1, lr_2, ..., lr_n\}$

step 1: divide URS into subsets of tasks C1, C2, ... C_M according to $ur.\lambda$. M is the number of the type , then divide C_i into a exclusive subset of tasks C_{i0} and a few dividable tasks subsets $C_{i1},...,C_{ik}$. According to g.

Step 2: divide PRS according to $pr.\lambda$ into subsets of resources $P_1, P_2, ...$ ， P_M, then devide P_i into $P_{i0},P_{i1},...,P_{ik}$ according to the type in order to execute the corresponding subset of tasks $C_{i0},C_{i1},...,C_{ik}$。

Step 3: as for C_{i0} ，if m≤n or m≥n and the number of tasks is much smaller than the number of short tasks strategy 2 is employed, otherwise, strategy 1 .

Step 4: as for C_{ij} ，$j\ge1$ ，If the workload of the tasks in the task group is little difference, and P_{ij} calculation capacity is almost the same circumstances, in order to simplify administration, strategy 3 is employed, otherwise, strategy 4.

Step 5: When all group tasks are completed, next batch of task will be scheduled.

5. Analysis and Evaluation of Strategies

Conclusion 1：Strategy 4 assures the single batch tasks can be completed at the same time.

Proof：According to Strategy 4, any task ur_i completion time ft_i *satisfy*:

$$ft_i = T^{(4)} \cdot \frac{ur_i.w}{\sum_{j=1}^{n} w_{ij}} = T^{(4)} \cdot \frac{ur_i.w}{\sum_{j=1}^{n} (s_{ij} \cdot pr_j.v)} = T^{(4)} \cdot \frac{w_\beta}{s_\beta} \cdot \frac{1}{\sum_{j=1}^{n} pr_j.v}$$

Conclusion 2：Strategy 4 outperforms Strategy 3 :(1)$mst^{(4)} \le mst^{(3)}$，(2)$art^{(4)} \le art^{(3)}$，(3)$asr^{(4)} \le asr^{(3)}$，(4) $ru^{(4)} \ge ru^{(3)}$。

Proof:

(1) $$mst^{(3)} = \frac{m}{n} \cdot \frac{max_{1\le i\le m} ur_i.w}{min_{1\le j\le n} pr_i.v} + \frac{c}{n} \cdot \frac{max_{1\le i\le m} ur_i.w}{s_\alpha \cdot min_{1\le j\le n} pr_i.v}$$

$$mst^{(4)} = \frac{\sum_{i=1}^{m} ur_i.w}{\sum_{j=1}^{n} pr_i.v} + c \cdot \frac{min_{1\le i\le m} ur_i.w}{\sum_{j=1}^{n} pr_j.v}$$

Obviously, at the above formula, the first and second parts of $mst^{(3)}$ are respectively greater than that of $mst^{(4)}$, so $mst^{(4)} \le mst^{(3)}$. Similarly we can prove (2)art$^{(4)} \le art^{(3)}$，(3)$asr^{(4)} \le asr^{(3)}$.

(4) $ru^{(4)}$ is only affected by task switching overhead, however, In the strategy 3,both the time slice of the faster resource and the short task time slice are wasted partly, so easily to proof $ru^{(4)} \ge ru^{(3)}$。 □

Conclusion 3：(1) if m≤n，that is the number of the user tasks is less than that of resources，then $mst^{(1)} \ge mst^{(2)}$ (2)if m>n and ignoring time-division multiplexing switching overhead then $mst^{(2)} \ge mst^{(4)}$

Proof：(1) According to definition 5, we get:

$$mst^{(1)} = max\{\frac{min_{i\in[1,m]}(ur_i.w)}{max_{j\in[1,n]}(pr_j.v)}, ..., \frac{max_{i\in[1,m]}(ur_i.w)}{min_{j\in[1,n]}(pr_j.v)}\} = \frac{max_{i\in[1,m]}(ur_i.w)}{min_{j\in[1,n]}(pr_j.v)}$$

$$mst^{(2)} = max\{\frac{max_{i\in[1,m]}(ur_i.w)}{max_{j\in[1,n]}(pr_j.v)}, ..., \frac{min_{i\in[1,m]}(ur_i.w)}{min_{j\in[1,n]}(pr_j.v)}\}$$

So $mst^{(1)} \ge mst^{(2)}$

(2) $$mst^{(4)} = T^{(4)} \cdot \frac{w_\beta}{s_\beta} \cdot \frac{1}{\sum_{j=1}^{n} pr_j.v} = \frac{\sum_{i=1}^{m} ur_i.w}{\sum_{j=1}^{n} pr_j.v}$$

$$= (\frac{pr_k.v}{\sum_{j=1}^{n} pr_j.v} \cdot \sum_{i=1}^{m} ur_i.w) \cdot \frac{1}{pr_k.v}, k = 1,2,...,n$$

$$mst^{(2)} = max_{1\le j\le n} (\sum_{y=1}^{x_j} \frac{ur_{jy}.w}{pr_j.v})$$

so $mst^{(2)} \ge t \ge mst^{(4)}$ □

Conclusion 4：Following the above symbol, assumed the arrival interval of the batch task is G，the cloud system scheduling cycle is D, single batch task completion time is mst, if $mst \le D$，then says single batch task scheduling and resource management strategy is feasible and effective; if $mst \le D \le G$，then says Task scheduling and resource management strategy of the entire cloud system is feasible and effective;if $\lim_{m\to\infty} (mst \le min_{1\le i\le m} ur_i.\tau))$, then says cloud system PRS supply URS with infinite computing power.

Proof：obviously. □

Through discussed above can be seen : the nature of the time-division multiplexing virtualization is, according to the characteristics of user tasks, as well as the resources state of the environment, to set scientific and reasonable relationships between the following variables to strive to meet customer needs and strive to meet conclusion 5 .Those variables are arrival interval G of the batch task, cloud system scheduling cycle D, single batch task completion time mst_i, multiplexing cycle T_i and time slice s_{ij} and so on.

6. Conclusions

This paper focuses on dividable batch tasks running in the cloud computing environment, and studies resource virtualization based on time-division multiplexing. We first introduce the concepts used in the process of on-demand resource allocation. Then we give the principle of time-sharing multiplexing methods and propose priority and exclusive strategy, equal time slice and same amount of workload strategy, unequal time slice and different amount of workload strategy, and type-matching-first integrated multiplexing algorithm to deal with different requirements. Furthermore, detailed performance analysis for different strategies is performed and certain conclusions are presented, which is the guidance for resource time-multiplexing in special application scenarios. Our methods make sure that users can use system resources equitably in cloud computing with multi-tenants, and allow limited computational resources to meet user requirements as more as possible, as well as improve the utilization of cloud resources.

References

[1] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: vision, hype, and reality for delivering IT service as computing utility. Proc. Of the 10th IEEE Int Conf. on High Performance Computing and Communications, 2008, pp5-13.

[2] I. Foster, Y. Zhao, I Raicu, et al. Cloud computing and grid computing 360-degree compared. Proc. of Grid Computing Environments Workshop, 2008, pp1-10.

[3] M. Armbrust, A. Fox, R. Griffith,et al. Above the clouds: a berkeley view of cloud computing.Technical Report, UCB/EECS-2009-28, 2009.

[4] S. Govindan, J. Choi, A.R. Nath, et al. Xen and Co.: communication-aware cpu management in consolidated xen-based hosting platforms. IEEE Transactions. on computers, 2009, 58(8): 1111-1125.

[5] Qumranet, Linux kernel virtual machine, http://kvm.qumranet.com .

[6] Vmware white paper, virtualization overview, http://www.vmware.com/pdf/virtualization.pdf.

[7] P. Barham, B. Dragovic, K. Fraser, et al. Xen and the Art of Virtualization. Proc. of the 19th ACM symposium operating systems principles, 2003,pp164-177.

[8] E. Walker. Benchmarking Amazon EC2 for high-performance scientific computing. http://www.usenix.org/publications/login/2008-10/benchmark_results.tgz. 2008, 33(5):18-23.

[9] K. Kopper.The Linux enterprise cluster. No Starch Press San Francisco, CA, USA, 2004.