ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp.42-47
doi: 10.18178/wcse.2017.06.007

Group Dispatch and Task Transfer in Cloud Computing

Bin Chen [1+], Yizhou Dong [2] and Mingrong Mao [3]

[1,2,3] Nanjing Normal University, Jiangsu, China

Abstract. This paper aims to group dispatch application results in a study of the cloud, the effect is reflected in the composite indicator of its performance and cost price. Study of the model is to analyze performance through virtualization and integrated consideration of a group dispatch task transfer and demand-based control as a means to architecture based on commercial compute cloud conducted experiments to assess the efficiency of the experimental background for supporting data. The results show that the dispatch configuration can effectively arranged in the environments, and the platform which as a true high-performance computing environment or other high-performance applications areas, has a high-performance cost ratio.

Keywords: Cloud computing, Group Dispatch, High Performance Computing, Task Transfer

1. Introduction

The ultimate goal of cloud computing is to use of computing resources like the use of conventional resources, users can obtain the computing resources in accordance with their needs. The importance of cloud computing is to provide access to computing infrastructure for small companies or organizations without prior investment. Small organizations and individual research organizations can now access large computing resources, utilize the cluster based on virtual unlimited resources [1]. These clusters require only a fraction of the maintenance cost, and operate in a paid - off mode.

The core of any distributed system is the task dispatcher, it's role is assign tasks to the server or virtual machine. Usually, the goal of the dispatcher is to achieve the dispatch strategy to achieve fault response time and lower the rate faster, this is achieved by minimizing the redundancy postpone. In our model, the virtual machine dispatcher must take strategy to minimum time spent in order to obtain the best cost efficiency ratio. The modelling system implements a special instance of parallel task dispatch called group dispatch, the tasks in this instance must be executed simultaneously and dispatched to meet the requirements of their continuous communication [2]. It is necessary to establish a point-to-point mapping between the task and the virtual machine, it is also necessary to avoid the bottleneck problem and deadlock caused by the input of the other non-running task as a task. In the field of distributed and cluster systems, group dispatch has been widely studied in recent years. However, the current application group dispatch is a static system designed to pre-arrange the total number of servers and the size of the single task.

The dispatch strategy of cloud computing platform has been studied before. There are researchers Used fuzzy virtual machine management architecture to study the utilization of virtual machine in batch dispatch of parallel tasks. In these models, the task does not require interaction and can be executed independently [3]. We study the methods and effectiveness of applying group dispatch strategies in distributed cloud computing systems that provide virtual machines dynamically. We make use of two task dispatch algorithms, adaptive first come first service(AFCFS) and Maximum priority task first service(MPTFS) to control the scope of virtual diversification and multi-tasking. In this paper, we integrate the transfer mechanism and the demand control system in our model, and compare the effectiveness of these methods.

+ Corresponding author. Tel.: +86-13813834627
E-mail address: njnuchenbin@njnu.edu.cn

2. System and Working Mode

The development of virtual modeling including dynamic virtual machine(DVM) and virtual machine cluster dispatcher [4]. There is no need for a virtual machine lease when the system is initialized, the number of virtual machines can be dynamically increased and reduced. When the virtual machine dispatcher has the ability of dispatch the distributed parallel task, each virtual machine will achieve its own waiting queue. The virtual machine dispatcher also has a task queue, they cannot be dispatched according to the time of the virtual machine and the system load. For the simplify purpose, the virtual machine dispatcher itself is not included in the total number of virtual machine. Similarly, the task transfer and demand control mechanism will be managed by the virtual machine controller. Virtual machine communication can be considered as a free competitive relationship. However, any potential communication includes the task execution time. So, we must take into account the case of the task dispatcher queue exists task dispatch postpone.

Group dispatch requires the task to run in parallel, so each task requires a series of idle virtual machines that are equivalent to the needs of the execution. In our model, the degree of parallelization is random, it can be divided into the following two categories:1) Low simultaneous task, the sum of tasks is maintained between 1 and 64 by probability of q. 2) High simultaneous task, the sum of tasks is maintained between 64 and 128 by probability of $1-q$. q is the number of tasks that determine the total number of tasks, the coefficients can belong to the first or second class. Therefore, the average number of tasks can be calculated in accordance with the following publicity:

$$ANT = qE([1...64]) + (1-q)E([65...128]) \quad (1)$$

Here E represents the discrete uniform distribution of the value span. At the same time, the arrival cycle of the task is the exponential distribution of $1/\lambda$, and the average service time is exponential distribution with a as the average. There is no relationship between the sum of tasks, thus a low simultaneous task cost a longer service time. The context switching of group dispatch requires a high cost, so the task is often not implemented in a preemptive manner.

3. Architecture Introduction

Entrance point of the system is virtual machine dispatcher as shown in Figure 1. Task parallelism is less than or equal to the available immediately distributed virtual machine. Concern the distribution of virtual machine tasks, the virtual machine dispatcher using the shortest queue priority algorithm, according to the shortest queue principle, the algorithm distributes the tasks to the virtual machine.

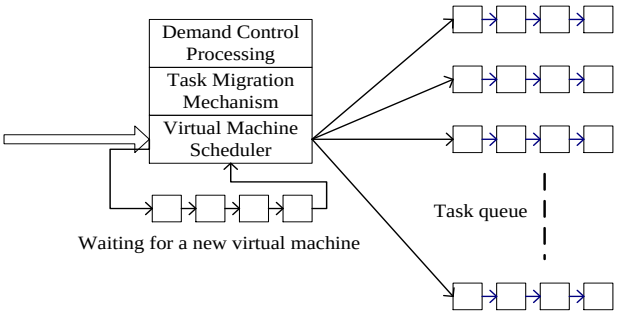

Fig.1: system mode

Our model make use of two group dispatch algorithms. Both AFCFS and MPTFS relate to a certain degree of research. Adaptive first come first service try to dispatch tasks when every virtual machine becomes idle. If there is no such task, AFCFS will attempt to dispatch the task by minimizing the queue length. AFCFS dispatch tends to work against smaller tasks that are easier to dispatch but will increase the waiting time for the maximum task. Maximum priority task first service provides a more priority dispatch policy for large task. In each dispatch cycle, MPTFS will attempt to dispatch the maximum priority task that is distributed in the idle virtual machine. This method greatly increases the response time of large tasks by authorization. Because a large number of virtual machine resources often be released in the dispatch process, smaller tasks will be reduced in waiting time relative to AFCFS.

There is a common problem in group dispatch, that is, the processor will always be idle when the queue is waiting for the queue [5]. In order to avoid this time fragmentation, the implementation of transfer is necessary. The transfer process involves migrating a task to a free virtual machine from a virtual machine queue. Although this method solves the problem mentioned above, it also brings a lot of overhead to the system.

Our system reduces the system overhead by considering the task transfer security. The specific approach is: when the system cannot dispatch tasks in common way so allow the task transfer. When this occurs, the transfer attempts to find out if the task in the task queue that is suitable for running in the current idle virtual machine. If there is such a task, the system will use one of the following two strategies to migrate the task, select which configuration ultimately depends on the virtual machine. These two strategies are: primacy adaptive and optimum adaptive. the former select the task in the first place of the appropriate task queue, the method is simple to implement, and the average cost is little. The latter choose the most suitable task, this task has the highest level of parallel adaptation for idle virtual machines. Due to the necessary of implement the optimal task adaptation algorithm, this method will bring more overhead. In order to achieve the purpose of transfer task efficiently, we also introduce a transfer monitoring mechanism to reduce the total number of transfer. This monitoring ensures that the task dispatch occurs only when the number of tasks is less than the specified number. When the end of transfer, the task is dispatched to be executed immediately in the next clock cycle. The way to deal with this problem is to prevent the occurrence of multiple repeated transfer events on the same task, and to generate more load overhead.

In our system, we also design a special subsystem which includes queue priority, the purpose of this subsystem is to control the demand. When the queue contains a demand control task, regular task dispatch and transfer will be suspended. When the expansion coefficient of the a reaches the set threshold, the task will automatically trigger. The calculation method of extension coefficient is:

$$P_{unit} = \frac{(ISW_i + t_i)}{t_i} \tag{2}$$

Here ISW_i and t_i are instance ready time, the length of time depends on the actual execution time i. The choice of the extension coefficient plays a very important role in the whole task control.

Cloud supply users with the capacity of add or reduce the number of computing resources in accordance with the actual number of virtual machines request. The process includes a postpone, which is generated by the virtual machine manager and the time required to create and set up a new VM. Every time the postpone is typically less than one hundred seconds. The postpone in our virtual model is designed as average 0.25, random numbers satisfying the normal distribution of $\mathrm{N}(0, 0.5)$.

A complex subsystem is realized for adding and removing virtual machines from the system. The virtual machine rental occurs when the following conditions are encountered: 1) Virtual machine deficiency, this happens when a task contains more than one subtask, and the total number exceeds the number of available virtual machines. The task is arranged in the virtual machine manager until a new virtual machine can be supplied. 2) Virtual machine overriding, in each distribution, the detector checks the situation of the virtual machine waiting queue. The formula for calculating the average extension coefficient is as follows: Here J_k is the current waiting task number in virtual machine k, P_k is the number of virtual machines currently leased on the system.

$$AEC = \frac{\sum_{k=1}^{P_k} J_k}{P_k} \tag{3}$$

AEC generally outperforms the previously defined load threshold. The system provides a virtual machine controller capability equivalent to the number of tasks to achieve the new virtual machine, the task queue will wait until the new virtual machine is available.

When the system does not need to use some virtual machines, virtual machine may be released. This function is particularly key because it influence the overall efficacy of the system. How to evaluate whether a virtual machine can meet the release conditions depends on the following three points: The virtual machine is free and its waiting alignment is empty; There are no tasks waiting to be re dispatched in the virtual

machine manager queue; There is no task in the planned transfer active state. When all the above three points are met, we think that the virtual machine is in a state of release.

4. Experiment and Analysis

The study is pay close attention to the evaluation of system performance and the overhead assessment. The following indicators are used in performance evaluation.

Response cycle RT_k, Represents the interval of the arrival, including feedback of the request of the task k. Its mean value can be defined as follows, here S represents the total number of tasks:

$$RC = \frac{\sum_{k=1}^{S} RC_k}{S} \tag{4}$$

Response postpone ratio S_k, It is defined as $S_k = RC_k / d_k$. This indicator shows the postpone tolerance ratio of a task relative to its service time. Because the index can easily be influence by the service time, so we use the following restrictions, λ act as the tune parameter in the experiment:

$$CLB = \max\left\{\frac{RC_k}{\max\{d_k, \lambda\}}, 1\right\} \tag{5}$$

General amount of transfer $Trans_{amount}$, We use the total number of clear effects to evaluate the system performance. Since the operation of the cloud is associated with a cost, the system maintains a good balance between the response cycle and the required cost. Therefore, we need to integrate in accordance with the following index as the cost of performance efficiency indicators. Virtual machine complete lease cycle (LC) and average response cycle (RC). Here DIF_{LC} is the difference between LC and two virtual experiments, DIF_{RC} is the correlation between response time.

$$PEI = DIF_{LC} + DIF_{RC} \tag{6}$$

Here DIF_{LC} is the difference between LC and two virtual experiments, DIF_{RC} is the correlation between response time. The model described above is implemented by a discontinuous simulation event. Each of the results presented here is the result of an average variance instance for each arrival ratio λ, dispatch algorithm, transfer algorithm, and a collaborative virtual experiment for each task size. Figure 2 and Figure 3 depicts the above performance metrics, including RC and CLB, and the various states of the $p = [0.25, 0.50, 0.75]$. Each graph shows the synergy effect between the optimum adaptive first service and the maximum priority task first service to the task arrival ratio, response cycle, response postpone ratio, and extension coefficient. Figure 4 shows the number of transfer tasks in each experiment, thus completing the transfer estimate. The final table 1 shows the performance and cost efficiency data for all parameter cases. In our previous work, there was no transfer and demand control, the conclusion is that for all tasks with high arrival ratio, the first priority task has a higher cost than the adaptive first come first served. Table 1 shows the basic different effects of transfer and demand control. Different task size and extension parameter seem to have a minor impact. When the minus parameter is used to describe the adaptive first come first service, the difference is indistinctness.

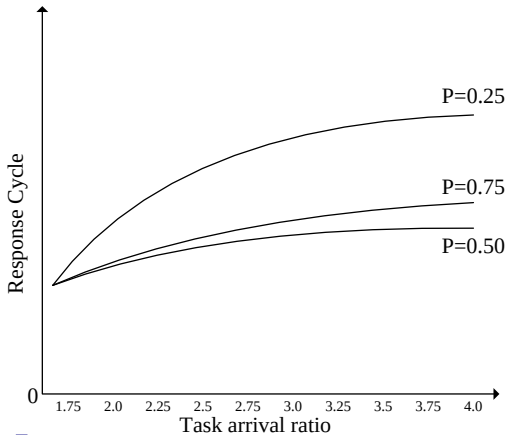

Fig. 2: Performance indices of different response times under the same arrival ratio

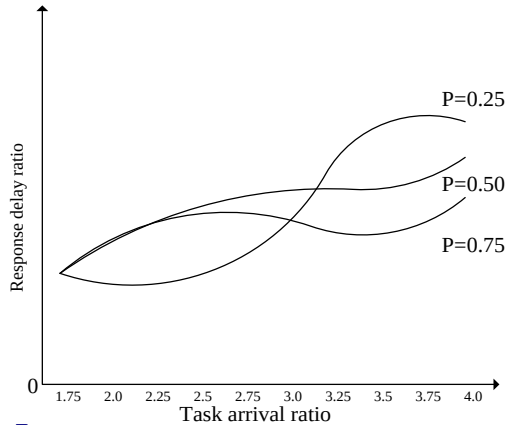

Fig. 3: Performance index of different response postpone ratio at the same arrival ratio.

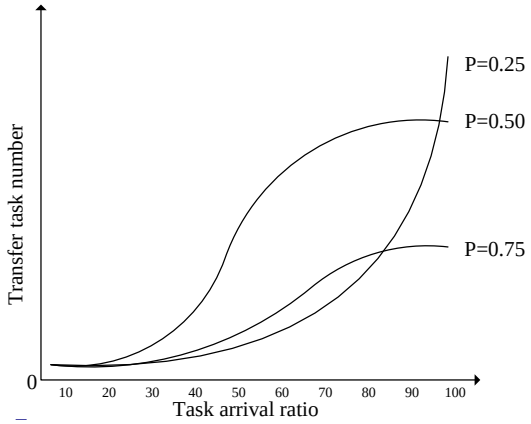

Fig. 4: Task transfer number index at the same arrival ratio.

Table 1: Cost and Performance Coefficients Result

p	λ	FCFS(10)	$AFS(10)$	FCFS(20)	$AFS(20)$
$p=0.25$	1.75	-0.01632	0.01279	-0.05801	-0.05102
	2.50	-0.05649	-0.02913	0.01514	-0.02927
	3.25	-0.06112	-0.03313	-0.13715	-0.20318
	4.00	-0.04145	-0.04213	-0.02531	-0.05199
$p=0.50$	1.75	-0.01207	-0.00280	0.00413	-0.03812
	2.50	-0.04132	0.00877	-0.04232	-0.04541
	3.25	-0.05769	0.00749	-0.01832	-0.07681
	4.00	-0.06142	-0.02193	-0.12147	-0.04802
$p=0.75$	1.75	-0.03158	-0.03311	-0.00691	-0.03185
	2.50	-0.04815	-0.02188	-0.05101	-0.07903
	3.25	-0.02110	-0.01898	-0.06613	-0.03573
	4.00	-0.02741	-0.04726	-0.08801	-0.07515

5. Summary

In this paper, two important concepts of task transfer and demand control are applied into the model. The application of them has a profound impact on the experimental model. In the future, we want to test the actual environment pressure in order to better adapt to the surroundings.

6. Acknowledgment

The Paper is Supported by the fund: Special subject of China Association of Higher Education - Large data integration platform for the campus information intelligence user operation and maintenance services(2016XXYB02).

7. References

[1] Li Dawei, Zhao Fengyu. Memory scheduling strategy for virtual machine in private cloud platform[J]. Journal of Computer Applications, 2014, 34(9): 2523-2526.

[2] Ioannis A. Moschakis, Helen D. Karatza. Performance and Cost evaluation of Gang Scheduling in a Cloud Computing System with Job Migrations and Starvation Handling[J]. IEEE Symposium on Computers and Communications, 2011:418-423.

[3] Wang Zhigang, Wang Xiaolin, Jin Xinxin, et al. Mbalancer: predictive dynamic memory balancing for virtual machines[J]. Journal of Software, 2014, 25(10): 2206-2219.

[4] HaiLong, Sun. Research of multidimensional indexing in a cloud computing system[J]. Associacao Iberica de Sistemas e Tecnologias de Informacao. 2016.1(17): 324-335.

[5] Adam, Sara Alfatih. Multilevel authentication scheme for cloud computing[J]. International Journal of Grid and Distributed Computing. 2016.9(9): 205-212.