High-priority and high-response job scheduling algorithm

Changwang Liu

School of software, Nanyang Normal University, Nanyang 473061, China

Jianfang Wang*, Anfeng Xu, Yihua Lan

School of Computer and Information Technology, Nanyang Normal University, Nanyang 473061, China

Chao Yin

School of information science and technology, Jiujiang University, Jiujiang 332005, China

**Correspondence should be addressed to Jianfang Wang (jianfangwang 2015@163.com)*

Abstract

In this paper, we have researched a lot of scheduling algorithm in different cloud environment and proposed a high-priority and high response scheduling algorithm. This algorithm has created its special constraints according to their quality of service (Qos in short) preference. In accordance with the classification of the task, a generally look forward to the task is added in its main element part. That gives a connected method between generally expected tasks added and resources. Then, we give the evaluation function evolved from Berg model. The model's parameters are refined according to the evaluation results in order to adapt the system. For tasks with multiple Qos properties, this paper gives multiple generally expect in processing tasks by applying people element analysis theory. Based on the model, we have proposed job scheduling algorithm based on high-priority response ratio. The algorithm takes both the request service time and waiting time into account. Job scheduling simulation program has been achieved on the extend CloudSim platform, which is recompiled and generated. And the algorithm simulation validation and comparative analysis of the experiments showed that the algorithm can effectively perform user tasks meanwhile improve the performance of the system.

Key words: SCHEDULING ALGORITHM, QUALITY OF SERVICE, HIGH-PRIORITY RESPONSE, CLOUDSIM

Introduction

Cloud computing is a development of grid computing, parallel computing and distributed computing [1]. It is an emerging commercial computing model, and has a good developing prospect. Grid computing, which genes accompanied with large-scale computing needs, is a produce a computing model to integrate the scattered resources and enable resource sharing and collaborative work, it's appearance solves a lot of complex issues in many field.

In the end of 2007, cloud computing began to develop. It distributes the calculating tasks in a resource pool of a large number of computers. That gives a way for various application systems to get computing power, storage space and a variety of software services according to the needed. Cloud computing is a business model that package various resources from data center into an outside services through the Internet.

The purpose of the job scheduling [2] in cloud environments is to achieve optimal scheduling of jobs submitted by users, and to maximize the throughput of the system. The main objectives are the optimal span, quality of service, load balancing and economic principles:

(1) Optimal span

Span, which is the most important and common targets, refers to the length of scheduling, which is the running time from the start of the first task to the finish of the last one. Optimal span is the common goal of the user and the grid system.

(2) Quality of Service(QoS)

While the grid system provide users the computing and storage services, user's demand for resources reflect from QoS. During task scheduler, the task management and scheduling system must protect the grid application's QoS.

(3) Load balancing

Load balancing [3] is a critical issue in the development of parallel and distributed computing applications. Grid task scheduling involves cross-domain and large-scale applications, load balancing direct impact on the utilization of system resources.

(4) Economy principles

Resources are widely distributed geographically in a grid environment, and each resource belongs to a different organization, which has its own resource management mechanisms and policies. According to the principles of market economy in real life, the use of different resources is not the same cost. Resource management and task scheduling driven by market economy must make the consumer both sides (resource users and resource providers) mutual benefit, that's the principle to maintain the stability of the grid system.

By using simulation tools CloudSim and combining the basic mechanism of scheduling algorithm, the paper conducts in-depth discussion and analysis of key technologies in building scheduling mechanism based on cloud platform. The contributions of this paper are described as follows:

•By introduction of the sociology wealth distribution theory - Berg model of distributive justice, two fairness constraints are appended in the cloud environment. First, the selection process of resources using generally expect constrained. Second, the fairness of the allocation judgment and model self-correcting using allocation results fairness evaluation function. By using those constraints, resource allocation and job scheduling are achieved in cloud computing environment.

•The idea that job scheduling algorithm derive from the fairness perspective of using resources are introduced, which makes up traditional job scheduling algorithm that characteristically focus on efficiency.

• Taking into account the operational requirements of the service time issue, we have combined with the operating system scheduling idea and proposed a job scheduling algorithm called HRP algorithm based on higher response priority. For a relatively short service job requirements, this job scheduling algorithm can make it wait for a short time. It will reduce the average waiting time jobs and improve the system throughput.

•Analysis and expand the simulation platform CloudSim. Comply job scheduling algorithm simulation on an expanded and recompile platform.

The rest of this paper is organized as follows. Section 2 describes related work in cloud environment and scheduling algorithm. The architecture and the design of our system are introduced in Section 3. Section 4 is the experimental result and evaluation. Section 5 is the conclusion.

Related Work

A. Job Scheduling

The so-called job scheduling is a dispatch method that assigns suitable task of suitable jobs to the appropriate tasks server. There contains a two-step process. Firstly, we select the job and the task in this job. Same with other task allocation work, job scheduling is also a complex task. Bad task allocation may lead to the increase in network traffic, the overload of some tasks server and lead to an efficiency decline, so on. Moreover, the distribution of tasks or a consistent pattern of

different business backgrounds, may require different algorithms in order to meet the demand. Therefore, after the Hadoop 0.19 release, job scheduling becomes independent as a plugin component, allowing the user to provide different implementations according to their needs.

Hadoop [4] is the Apache Software Foundation's open source distributed computing platform, which mimics and Google File System implements (GFS), parallel programming model (MapReduce) and other major technologies. In Hadoop Distributed File System and MapReduce as the core of Hadoop provides users with a system-level details Transparent distributed infrastructure, HDFS high fault tolerance, high scalability, etc. It allows users to deploy to form a distributed system. MapReduce programming model allows the users to develop the parallel applications without understanding the detail of the details of distributed systems. So users can take advantage of Hadoop easily to build their own distributed computing platform. They can also take advantage of set cluster computing and storage capacity and complete the processing of massive data. Because of Hadoop's MapReduce programming box Frame efficiency and HDFS data processing job management capabilities, it has been widely applied in the field of scientific research and large-scale data processing.

In computing power scheduling algorithm(Capacity Scheduler), you can define multiple job queue. When the job is submitted, it will be directly put into a queue. Each queue can be obtained by configuring a certain number of the TaskTracker resources user to process Map operation and Reduce operations, scheduling algorithm will allocate an appropriate amount of computing resources for the queue according to the configuration file.

Approximate with fair scheduling algorithm, resources that has been allocated but still in the idle state will be share by every other queue to improve the resource utilization. When the queue that didn't obtain a sufficient amount of resources according to its set value begin to increase the operating pressure, the resources that have been assigned to it but was occupied by another queue will ration back to the queue it should be immediately after the completion of current task. It can be seen, the idea of computing power scheduling algorithm is to simulate an independent Hadoop cluster resource with specified computing power for each job in queue, rather than fair scheduling algorithm as trying to achieve a fair sharing of resources among all jobs [5].

B. How to Use Job Schedule

Zaharia et al. have proposed to the ring in a heterogeneous to improve the performance of MapReduce under the border. His main work is in the analysis of the Hadoop job scheduler. After considering the advantages and disadvantages heterogeneous cluster environments, they have proposed a LATE (Longest Approximate Time to End) job scheduler [6]. According to the progress of the job and remaining run time, deadline scheduling algorithm can complete the job within the deadline time [7]. Kc et. al. proposed scheduling algorithm is based on the job and the current deadline of real-time operating systems operation [8]. It can predict whether the job submission can be completed within the deadline time or not.

In a multi-user Environment for fair scheduling in dynamic reallocation of resources, fair scheduling algorithm can select and kill recently started tasks without considering the characteristics of the job as well as data. Tao et al proposed an improved the fair scheduling algorithm when re-allocation of resources, the choice of killing a special operations task considering local sexual and data [9]. Literature [10] also starts from the local data and takes into account the scheduling of public leveling and data localization; we propose a lsap-fair-sched job scheduling algorithms. Rasooli et al. developed a cluster task scheduling algorithm which takes execution frequency and the average execution time as an important measure of task scheduling factors [11].

The Load Balancing Algorithm

A. The Design of the System

The Berg model is a theory that individuals obtain their distribution fairness through the comparison with common people’s distribution in social system structure, where reference to the common distribution is a kind of common justice (IE, fairness) universally acknowledged. The justice theory of social distribution is mapped to the resource allocation model in the cloud environment. That provides a new idea to job scheduling in the cloud environment. The entities of cloud system are users, resource providers and scheduling systems. Their corresponding main

bodies are user tasks, resources and scheduling policy. Therefore, cloud computing needs definition of the following concepts: the meaning of user tasks' fairness, task classification, general expectations, parameterization of tasks and resources, mapping of resources, Etc.

The first definition is the fair judging constraints of the resources allocating results. In Berg model, social individual's justice (fairness) is defined as a distribution relationship generating from the comparison with common people's justice distribution. That process is quantified as the ratio of the actual and expected distribution. In cloud computing, firstly, we should define the meaning of fair task. Cloud computing provides users on-demand services through Internet accompanied by on-demand payments. According to Berg model's principle, the fairness of resources used in the cloud system is embodied in the ability that cloud system can provide reasonable and available resources to different users according to their needs and task characteristics. Different users can get resources and satisfy their needs. We give following definitions in the cloud environment:
Definition 1: (Actual Resource, AR): Resource amount actually allocated by user task.

Definition 2: (Expected Resource, ER): Resource amount reasonably expected by user task according to the executing characteristics
Definition 3: (Justice, J): Looking forward to the maximum consistent between the actual resources allocated and the expected amount. That is called the fair execution of tasks. A judging function about task T is set as follows:

$$J_i = \theta \ln\left(AR_i / ER_i\right) \quad (1)$$

θ is an equilibrium constant, and $0<\theta<1$, ARi is the actual allocated resource amount for task Ti, ERi is the expected resource amount for task Ti. When ARi and ERi are equal, that is, the actual allocated and expected resource amount consistent. The function value is zero, which means justice (fair); The time when $Ji>0$, which means task Ti get too much allocation, is called upper-injustice; Similarly, the time when $Ji<0$ represent that task receive too little allocation is called under-injustice. Both the latter two condition show unfair, however as for job scheduling in the cloud environment, the higher resource allocating amount than expected can provide better service quality to user task. So the much injustice is allowed. This function is used to judge whether the resource allocating results are fair. This function is used to judge the resource allocating results are fair.

Definition 3-4. The Justice of the system (system Justice SJ): Assuming that system task sets are T= {T1, T2... Tn}; Corresponding fair judging function sets are J= {J1, J2... Jn}. Then the system's fair judging function is defined as:

$$J = \sum_{i=1}^{n} \left| J_i \right| \quad (2)$$

It can be inferred that the justice of system constraint can be implemented as follows:

$$\min J = \min \sum_{i=1}^{n} \left| J_i \right| \quad (3)$$

The system performs best when J obtain the minimum value, which represent the maximum fairest achieved to every system user. Parameter J can be used to optimize the overall justice of the system.

B. Algorithm Description

Algorithm flow is shown in Figure 1, the main program is divided into several major subroutine instructions

(1) Task sorting algorithm

Functions: Using a fast sorting algorithm to prioritize the task list.

Input: Task List

Output: Sequenced priority task list

(2) Task parameterization and classification

This algorithm expands the Cloudlet class in simulation tools by adding main task element descriptive variables and set their relative setting and access methods. ClassType variable is also added so the tasks' classification parameters can be assigned when users submit their task.

(3) VM parameter normalization algorithm

Functions: VM parameters normalized

Input: VM parameter sets

Output: Normalized parameter vector

(4) Matching algorithm

Functions: According to task classification, the general expectations of various task categories correspond to parametric general resource vector. That matches a binding between task and VM.

Input: Sets of tasks, sets of virtual machines
Output: The VM mapping sets' sub-modules: Calculating of the Euclidean distance

(5) Calculating of fair evaluation function

Functions: Fairness judge of resource allocating

Input: List of user tasks

Output: J sets.

C. High response priority scheduling algorithm

The scheduling algorithms which we used have their own features. The advantage of FIFO scheduling algorithm is simple and straightforward. It has lighter workload, but it ignores the differences among different jobs. Computing power scheduling algorithm contains multiple job queues which can be set to share cluster resources. Each queue uses FIFO scheduling policy with a priority. In the case of multiple users share a cluster, fair scheduling algorithm can get better results because the job can respond to each user in a relatively short period of time.

The three job scheduling algorithms only consider the waiting time and priority of jobs, without considering the operational requirements of service time. For a relatively short service job requirement, it is unreasonable to wait a long time to perform. We should increase the average waiting time jobs and reduce system throughput.

In order to solve the above problems, we propose a job scheduling algorithm based on high-priority response ratio. However, when we calculate the response ratio jobs, we should use the required service time of the job. It is difficult to determine the require service time in general, because the long jobs may reduce the service time intentionally or unintentionally in order to finish the implementation earlier.

In the fair scheduling algorithm, there are three methods to calculate the weight of the job. It shows as follows:

(1) By default, the weight of the job is calculated by the priority of jobs. It also can be calculated based on the size of the job and the job submission time. Weight is calculated as formula 4:

$$JW = \begin{cases} \log_2(TaskNum+1) & based\ \ weight\ \ size \\ 1 & otherwise \end{cases} \quad (4)$$

In formula 4, JW means the weight of the jobs while tasknum means the number of tasks in the jobs.

(2) The priority is calculated based on the weight.

$$JW = JW * PF \quad (5)$$

In formula 5, PF means the priority factor of the jobs.

(3) Depending on the users' needs, we adjust the weight of the job through a custom weight Adjuster class. Fair scheduling algorithm formula for each running job right weight will be updated as follows:

$$JW = JW * \frac{PW}{PRJW} \quad (6)$$

In formula 5, PF means the priority factor of the jobs and the default value is 1. PRJW indicates that the job pool is running all the weight of the job and the right.

In this paper, we propose the method to calculate weights based on high-priority response job scheduling algorithm which will be showed as formula 7:

$$JW = \frac{JW + RST}{RST} \quad (7)$$

In formula 7, JW means the weight of the job, JWT means waiting jobs. The value of JWT is the result of system current time subtracts commit time jobs;

RST indicates that the service time of the job requirements.

Experimental results

A. Experimental Setup

Data processing simulated by CloudSim [12] is shown in Figure 2. When datacenter resources are created, they register to CIS; the process of information exchange is managed by Datacenter-Broker.

B. Test Result

Due to the limitations of basic classes in CloudSim simulation platform, the failure rate of resources cannot be directly read; Also communication constraints between platform and emulator program makes it complex to calculate the cost of virtual machines, so we use the simulation scheduling of first class (completion time) and second class (bandwidth) tasks to validate the algorithm model.

General expected vector is set in accordance with task classification:

The first class tasks: W1 = [0.7, 0.1, 0.2]

The second class tasks: W2 = [0.3, 0.2, 0.5].

Wherein the first dimension is the weights of CPU amount, the second dimension is the weights of memory, and the third is the weights of bandwidth.

Create a set of tasks, whose parameter list is shown in Table I. Create a group of virtual machine with performance differences and preferences. The parameter list of virtual machines is shown in Table I.

Analysis of the experimental data results, as shown Figure 3. The comparison shows that overall efficiency of the job scheduling algorithm based on the Berg model is slightly worse than the optimal completion time algorithm. However task 0 to task3 are high computing power preference tasks which completion time is better than using a job scheduler aiming at an optimal completion time.

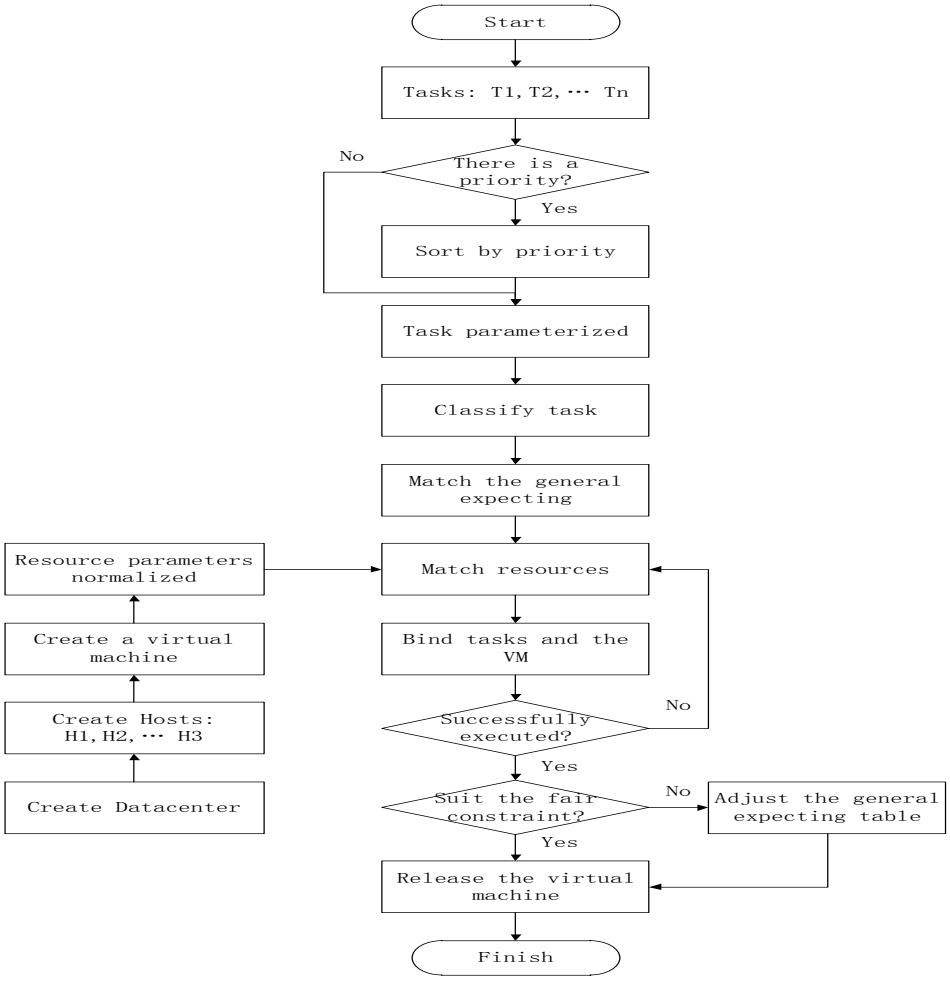

Figure 1. The algorithm description

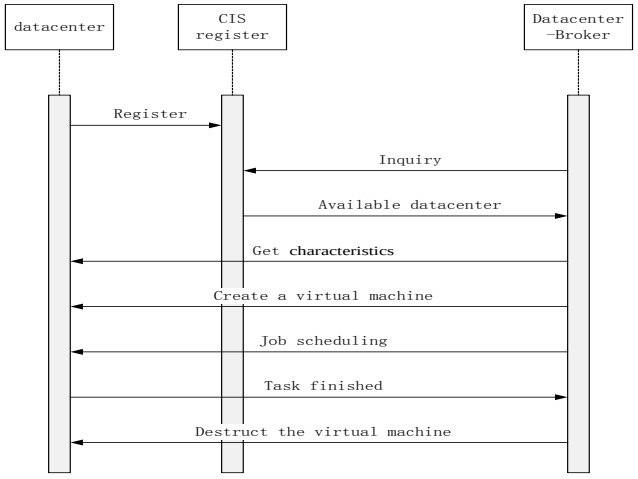

Figure 2. The experiment simulated by CloudSim

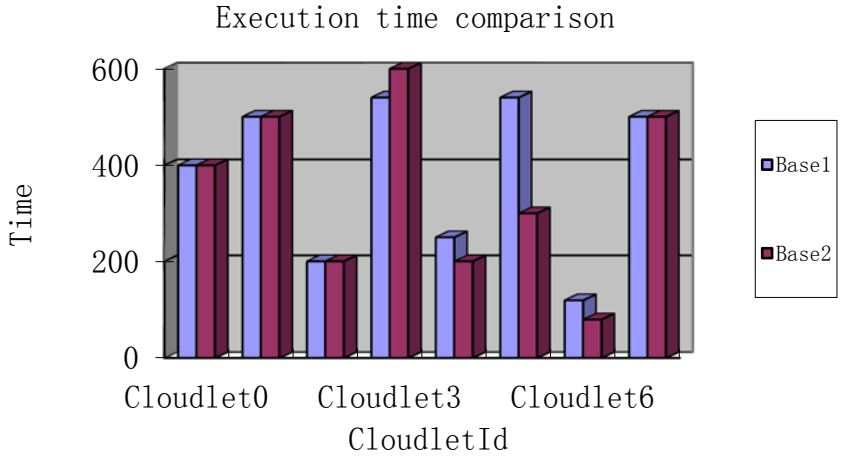

Figure 3. Execution time of different algorithm

Table 1. Selection among Parallel Options

Cloudlet Id	Class type	length	file_size	output_size	expectationtiome	expectation BW
0	1	2000	1200	250	200	-
1	1	1500	1000	200	100	-
2	1	1000	400	150	70	-
3	1	2500	2500	1000	250	-
4	2	1000	400	150	-	1000
5	2	1500	1000	200	-	1500
6	2	400	150	150	-	600

Conclusions

This paper researches the Berg model theory of justice distribution in the social field, imports that theory structure to job scheduling in cloud computing, where the dual fair constraint is established. The main element of User tasks is classified according to QoS preference. And justice in allocating sources is constrained according to the general expectance recorded in the main element of the task description. The QoS classification optimized the performance of virtual machine resources, and established a mapping between resources and tasks.

The evaluation function of resource allocating fairness is defined in the cloud

environment to judge the fairness of the allocating result, and furthermore to correct parameters of the model and adjust system's fairness. Task's custom description part concentrate on efficiency, the main description part concentrates on equity. Job scheduling algorithm and verification process is achieved on CloudSim cloud computing simulation platform, the experimental results show that the algorithm has reliable performance.

There are lots of works to do in the future which are described in the following directions. Firstly is learning of general expected vector in job scheduling algorithms. According to a nonlinear relationship between QoS and resources, fuzzy neural networks of the tasks QoS characteristic vector and the resource parameter vector can be built to learn and get a more accurate general expected vector. Secondly, Job scheduling algorithm based on the Berg model can be continually studied in terms of performance and perfection. There is also a vast space in refinement of the task classification and full support of various tasks classifications.

References

1. Foster I., Zhao Y., Raicu I., Lu S. (2008) Cloud computing and grid computing 360-degree compared. *Proc. Conf. on Grid Computing Environments Workshop*, Texas, U.S., p.p. 1-10.
2. Negi A., Kishore K. (2005) Applying machine learning techniques to improve linux process scheduling. *Proc. IEEE Region Conf. on TENCON*, VIC, Australia, p.p.1-6.
3. Belabbas Y., Yahya S., 2006. *Dynamic Load Balancing Strategy for Grid Computing*, World Academy of Science, Engineering and Technology.
4. Dean J. (2006) Experiences with MapReduce, an abstraction for large-scale computation. In *Proc. Conf. on* Parallel Architecture and Compilation Techniques. Seattle, Washington, p.p.1-1.
5. Matei Z., Dhruba B., Joydeep S. S, Khaled E, Scott S, and Ion S. (2009) Job Scheduling for Multi-User MapReduce Clusters. *Technical Report UCB/EECS-2009-55*, EECS Department, University of California, Berkeley.
6. Zaharia M., Konwinski A., Joseph A., et al.(2008) Improving Map Reduce performance in heterogeneous environments. *Proc. Conf. on Operating Systems Design and Implementation*, San Diego, CA, USA, p.p. 29-42.
7. Polo J., Carrera D., Becerra Y., et al. (2010) Performance-driven task co-scheduling for Map Reduce environments. *Proc. IEEE Conf. on Network Operations and Management Symposium*, Osaka, Japan, p.p.373-380.
8. Kc K., Anyanwu K. (2012) Scheduling Hadoop jobs to meet deadlines. *Proc. Conf. on Cloud Computing Technology and Science*, Hangzhou, China, p.p.388-392.
9. Tao Y., Zhang Q., L. Sh, et al. (2011) Job scheduling optimization for multi-user Map Reduce clusters. *Proc. Conf. on Parallel Architectures, Algorithms and Programming*, Tianjin, China, p.p.213-217.
10. Guo Z., Fox G., Zhou M.(2012) Investigation of data locality in MapReduce. *Proc. Conf. on Cluster, Cloud and Grid Computing*, Ottawa, Canada, p.p.419-426.
11. Rasooli A., Down D. G..(2011) An adaptive scheduling algorithm for dynamic heterogeneous Hadoop systems. *Proc. Conf. on the Center for Advanced Studies on Collaborative Research*, p.p.30-44.
12. Rodrigo N. C., Rajiv R., Cesar A.F. De R., and Rajkumar B. (2009) CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. *Technical Report.*