Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2015, Article ID 246420, 8 pages
http://dx.doi.org/10.1155/2015/246420

Research Article

User Utility Oriented Queuing Model for Resource Allocation in Cloud Environment

Zhe Zhang and Ying Li

Institute of Software, Nanyang Normal University, Nanyang, Henan 473061, China

Correspondence should be addressed to Ying Li; lying1024@163.com

Received 20 August 2015; Revised 29 September 2015; Accepted 8 October 2015

Academic Editor: James Nightingale

Resource allocation is one of the most important research topics in servers. In the cloud environment, there are massive hardware resources of different kinds, and many kinds of services are usually run on virtual machines of the cloud server. In addition, cloud environment is commercialized, and economical factor should also be considered. In order to deal with commercialization and virtualization of cloud environment, we proposed a user utility oriented queuing model for task scheduling. Firstly, we modeled task scheduling in cloud environment as an $M/M/1$ queuing system. Secondly, we classified the utility into time utility and cost utility and built a linear programming model to maximize total utility for both of them. Finally, we proposed a utility oriented algorithm to maximize the total utility. Massive experiments validate the effectiveness of our proposed model.

1. Introduction

Providers of cloud services usually provide different computing resources with different performances and different prices, and the requirements of users for performance and cost of resources differ greatly too. So how to allocate available resources for users to maximize the total system utilization is one of the most important objectives for allocating resources and scheduling tasks [1] and is also a research focus in cloud computing.

Traditional resource allocation models mainly focus on the response or running time, saving energy of the whole system, and fairness of task scheduling and do not take user utility into consideration [2]. However, the utility of a user in cloud environment is the usage value of services or resources, and it describes how the user is satisfied with the proposed services or resources while occupying and using them [3, 4]. In order to maximize the total utility of all users in cloud environment, it is necessary to analyze and model user utility first and then optimize it to get a maximum [5]. The modeling of user utility is very complex, as it needs a formal description considering many factors, such as the processing time that tasks have passed by [6], the ratio of finished tasks [7], the costs of finished and unfinished tasks [8], and the parallel speedup [9].

In a cloud server, requests of users, called tasks, come randomly, and a good description of these tasks is the Poisson distribution assumption. At the same time, under the commercialization constraint of the cloud environment, the utility of cloud server becomes much more important. In this paper, we formalized and quantified the problem of task scheduling based on queuing theory, divided the utility into time utility and cost utility, and proposed a linear programming method to maximize the total utility. The contributions of the paper are as follows:

(i) We modeled task scheduling as $M/M/1$ queuing model and analyzed related features in this queuing model.

(ii) We classified utility into time utility and cost utility and built a linear programming method to maximize total utility for each of them.

(iii) We proposed a utility oriented and cost based scheduling algorithm to get the maximum utility.

(iv) We validated the effectiveness of the proposed model with massive experiments.

The rest of the paper is organized as follows. In Section 2, we review related works about resource allocation and task scheduling in cloud computing. In Section 3, we formalize the tasks in cloud environment based on the queuing theory, define a random task model for random tasks, describe our proposed user utility model, and design a utility oriented time-cost scheduling algorithm. Experiments and conclusion are given in Sections 4 and 5, respectively.

2. Related Works

In cluster systems that provide cloud services, there is a common agreement in researchers that the moments, when tasks come into the system, conform to the Poisson distribution, and both the intervals between two coming tasks and the serviced time of tasks are exponentially distributed. In this situation, heuristic task scheduling algorithms, such as genetic algorithm and ant colony algorithm, have better adaptability than traditional scheduling algorithms. However, the deficit of heuristic algorithms is that they have complex problem-solving process, so they can only be applied in small cluster systems. The monstrous infrastructure of cloud systems usually has many types of tasks, a huge amount of tasks, and many kinds of hardware resources, which makes heuristic algorithms unsuitable.

There are a lot of researches about resource allocation or task scheduling in cloud environment, especially for the MapReduce programming schema [10]. Cheng et al. [11] proposed an approximate algorithm to estimate the remaining time (time to end) of tasks in MapReduce environment and the algorithm scheduled tasks with their remaining time. Chen et al. [12] proposed a self-adaptive task scheduling algorithm, and this algorithm computed the running progress (ratio of time from beginning to total running time) of the current task on a node based on its historical data. The advantage of [12] is that it can compute remaining time of tasks dynamically and is more suitable to heterogeneous cloud environment than [11]. In addition, Moise et al. [13] designed a middleware data storage system to improve the performance and ability of fault tolerance.

Traditional task scheduling algorithms mainly focus on efficiency of the whole system. However, some researchers introduce economic models into task scheduling, and the basic idea is optimizing resource allocation by adjusting users' requirements and allocating resources upon price mechanism [14]. Xu et al. [15] proposed a Berger model based task scheduling algorithm. Considering actual commercialization and virtualization of cloud computing, the algorithm is based on the Berger social allocation model and adds additional cost constraints in optimization objective. According to experiments on the CloudSim platform, their algorithm is efficient and fair when running tasks of different users. In addition, with respect to the diversity of resources in cloud environment, more researchers believe that the diversity will increase as time goes on with update of hardware resources. In order to alleviate this phenomenon and ensure quality of services, Yeo and Lee [16] found that while the resources were independently identically distributed, dropping resources that needed three times the number of minimal response time could make the whole system use less total response time and thus less energy.

The study of random scheduling began in 1966, and Rothkopf [17] proposed a greedy optimal algorithm based on the weights of tasks and expected ratios of finished time to total time. If all tasks had the same weights, then this algorithm became the shortest expected processing time algorithm. Möhring et al. [18] proved the optimal approximation for scheduling tasks with random finished time. They began with the relaxation of linear programming, studied the problem of integer linear programming for systems with homogeneous tasks, and got an approximate solution with the lower limit of the linear programming. Based on the above research, Megow et al. [19] proposed a better optimal solution with better approximation. In addition, Scharbrodt et al. [20] studied how to schedule independent tasks randomly. They analyzed the problem of scheduling n tasks on m machines randomly and gave the worst performance of random scheduling under homogeneous environment theoretically, and their result was the best among related works.

All of the above algorithms focus on the response or running time of users' requirements, saving energy of the whole system and fairness of tasks, and do not take user utility into consideration. However, utility of users is very important in cloud service systems. In order to maximize the total utility of all users in cloud environment, we analyze and model user utility first and then optimize it to get a maximal solution.

In addition, Nan et al. [21] studied how to optimize resource allocation for multimedia cloud based on queuing model, and their aim is to minimize the response time and the resource cost. However, in this paper, we deal with commercialization and virtualization of cloud environment, and our aim is maximizing utility. Xiao et al. [22] presented a system that used virtualization technology to allocate data center resources dynamically. Their aim is to minimize the number of servers in use considering the application demands and utility, whereas in this paper we aim to maximize the system's total utility under a certain cloud environment.

3. Proposed Model

3.1. Queuing Model of Tasks. In this paper, we describe randomness of tasks with the $M/M/1$ model of queuing theory, and the model is illustrated in Figure 1. The model consists of one server, several schedulers, and several computing resources. When user tasks are submitted, the server analyzes and schedules them to different schedulers and adds them to local task queue of the corresponding scheduler. Finally, each scheduler schedules its local tasks to available computing resources. In Figure 1, $t(d)$ is the waiting time of a task in the queue and $t(e)$ is the running time.

3.2. Modeling Random Tasks. In the following, we will analyze the waiting time, running time, and queue length of the proposed $M/M/1$ model.

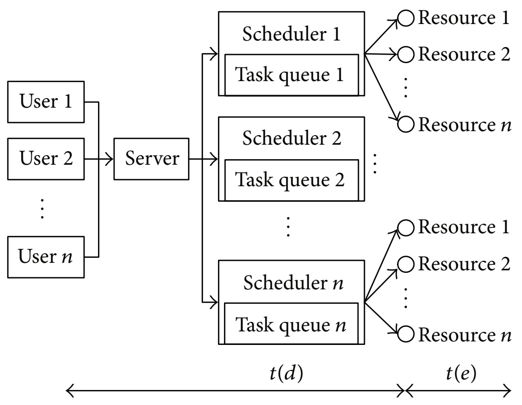


Figure 1: Scheduling model of user tasks in cloud environment.

Definition 1. If the average arrival rate of tasks in a scheduler is λ, the average service rate of tasks in a scheduler is μ, and then the service intensity ρ is

$$\rho = \frac{\lambda}{\mu}. \tag{1}$$

The service intensity describes the busyness of the scheduler. When ρ approaches zero, the waiting time of tasks is short, and the scheduler has much idle time; when ρ approaches one, the scheduler has less idle time, and thus tasks would have long waiting time. Generally speaking, the average arrival rate should be equal to or smaller than the average service rate, otherwise there will be more and more waiting tasks in the scheduler.

Definition 2. If we denote the expected length of tasks in a scheduler as L, the expected length of tasks in queuing as L_q, the expected total time (including both waiting time and running time) of a task as W, and the expected waiting time of a task in queuing as W_q, then we have the following equations according to queuing theory [17]:

$$\begin{aligned} L &= \frac{\lambda}{(\mu - \lambda)} = \rho(1-\rho) \\ L_q &= \frac{\lambda^2}{\mu(\mu-\lambda)} = \rho^2(1-\rho) = L\cdot\rho \\ W &= \frac{1}{(\mu-\lambda)} \\ W_q &= \frac{\lambda}{\mu(\mu-\lambda)} = W\cdot\rho. \end{aligned} \tag{2}$$

In addition, let $P_n = P\{N = n\}$ be the possibility of number of tasks in a scheduler at any moment; then, we have the following equation:

$$P_n = \rho(1-\rho). \tag{3}$$

If $n = 0$, then P_0 is the possibility that all virtual machines are idle.

3.3. Model of User Utility

3.3.1. Time Utility of Tasks. As we can see from Figure 1, the total time that a user takes from submitting a request to getting the result includes both waiting time $t(d)$ and running time $t(e)$. Here, the computing resources are virtual resources managed by virtual machines. Let T be total time; then, we have

$$T = t(d) + t(e). \tag{4}$$

In (4), the running time $t(e)$ is the sum of used time $t(f)$ and remaining time $t(u)$; that is,

$$t(e) = t(f) + t(u). \tag{5}$$

In order to calculate the time requirement of a task, the system needs to calculate the remaining time $t(u)$ and schedules $t(u)$ for different tasks to different virtual machines.

For analyzing the remaining time, we classified tasks into set $P = \{p_i \mid 1 \le i \le m\}$ and nodes into set $V = \{p_j \mid 1 \le j \le n\}$. According to statistical computing, we can get the average executing rate of task p_i on node v_j; that is, $R = \{r_{i,j} \mid 1 \le i \le m,\ 1 \le j \le n\}$, and then the remaining time of p_i on v_j is

$$t(u)_{i,j} = \frac{(w - w_u)}{r_{i,j}}, \tag{6}$$

where w is the number of total tasks and w_u is the number of finished tasks. For computing intensive tasks, w is the total input data and w_u is the already processed input data. Schedulers schedule tasks on virtual machine resources according to their remaining time and assure all tasks are finished on time.

A task can be executed either on one virtual machine or on n virtual machines in parallel, while being divided into m subtasks. We denoted the subtask set as $D = \{d_k \mid 1 \le k \le m\}$. While these subtasks are executed on different virtual machines, especially different physical nodes, the communication cost increases, and we use speedup to measure the parallel performance

$$s = \frac{T_1}{T_p}, \tag{7}$$

where T_1 is the time of a task in one node and T_p is the time of a task in p nodes. In order to make sure $s \le S_0$, all subtasks run in parallel, and total time of the task is

$$T = t(d) + \max\left\{t(e)_{k,j}\right\}, \tag{8}$$

where $t(e)_{k,j}$ is the time of subtask d_k on v_j and $\max\{t(e)_{k,j}\}$ is the maximal time of all subtasks.

3.3.2. Cost Utility of Tasks. In this paper, we assume that the cost rate of nodes is proportional to CPU and I/O speed, and tasks of different types consume different energy, different bandwidth, and different resource usage. So different tasks will have different cost rates.

Definition 3. Let $C = (c_{i,j} \mid 1 \le i \le m,\ 1 \le j \le n)$ be the cost matrix of task p_i on node c_j, and then total cost of a task is the product of node cost and running time; that is,

$$M = C \times T = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(c_{i,j} \times t(e)_{i,k,j}\right), \tag{9}$$

where $c_{i,j}$ is the unit cost of task p_i on node v_j and $t(e)_{i,k,j}$ is the time of subtask d_k on node v_j.

3.3.3. Formalization and Optimization of User Utility

Definition 4. Let the time utility function be U_t and let the cost utility function be U_c; then, the total utility is

$$U = a \times U_t + b \times U_c, \tag{10}$$

where $a + b = 1$, $0 \le a \le 1$ and $0 \le b \le 1$.

In (10), both time utility and cost utility are between 0 and 1 and a and b are the weights of time utility and cost utility, respectively.

The aim of utility oriented task scheduling is to maximize the total utility, and the constraints are expected time of tasks, expected cost, finished rate, speedup, and so on. In this paper, we classify user tasks into time sensitive and cost sensitive.

For time sensitive user tasks, change of running time for a task will affect the time utility a lot, and its definition is as follows.

Definition 5. The utility model of time sensitive user tasks is defined by the following equations:

$$\begin{aligned} U &= a \times U_t + b \times U_c, \\ U_t &= \frac{k}{(\ln(t-a) \times b)}, \\ U_c &= a \times c + b. \end{aligned} \tag{11}$$

The constrains are

$$F(D) = 1, \tag{12}$$

$$0 \le UT < U_t \le 1, \tag{13}$$

$$0 \le UC < U_c \le 1, \tag{14}$$

$$t(d) + t(e) < T_0, \tag{15}$$

$$\frac{L_q}{\lambda} < T_1, \tag{16}$$

$$\max\left\{t(e)_{k,j}\right\} < T_2, \tag{17}$$

$$C \times T < M_0, \tag{18}$$

$$s > S_0, \tag{19}$$

where D is the set of subtasks for all tasks, and the aim is to maximize total utility U.

For cost sensitive user tasks, change of running cost for a task will affect the cost utility a lot, and its definition is as follows.

Definition 6. The utility model of cost sensitive user tasks is defined by the following equations:

$$\begin{aligned} U &= a \times U_t + b \times U_c, \\ U_c &= \frac{k}{(\ln(c-a) \times b)}, \\ U_t &= a \times t + b. \end{aligned} \tag{20}$$

The constrains are

$$F(D) = 1, \tag{21}$$

$$0 \le UT < U_t \le 1, \tag{22}$$

$$0 \le UC < U_c \le 1, \tag{23}$$

$$t(d) + t(e) < T_0, \tag{24}$$

$$\frac{L_q}{\lambda} < T_1, \tag{25}$$

$$\max\left\{t(e)_{k,j}\right\} < T_2, \tag{26}$$

$$C \times T < M_0, \tag{27}$$

$$s > S_0. \tag{28}$$

In both Definitions 5 and 6, their aims are maximizing the total utility U, but the differences are the computation of U_c and U_t. Based on the above definitions, we propose a utility oriented and cost based scheduling algorithm. The details of the algorithm are as follows:

(1) Analyze user type for each user and select computing equations for U_c and U_t.

(2) Initialize parameters in constraints for $UT, UC, T_0, T_1, T_2, M_0$, and S_0.

(3) Compute L_q and W_q for each scheduler according to (1) to (3).

(4) With the results of step (3), tag X schedulers with least waiting time.

(5) Input some data into the X schedulers and set the highest priority for these tasks.

(6) Execute the above tasks, and record the running time and cost (see Pseudocode 1).

(7) Predict running time, cost, and corresponding utility of all tasks with time and cost of results from step 6, and tag the scheduler with the maximal utility.

(8) Schedule tasks in the scheduler with maximal utility, and optimize user utility (see Pseudocode 2).

(9) Wait until all tasks finish, and record the running time, cost, and corresponding utility.

```
if (user task is time sensitive) {
    select nodes with quickest speed, execute the above tasks, such that $s < S_0$;
} else {
    Select nodes with lowest cost, execute the above tasks, such that $s < S_0$;
}
```

Pseudocode 1

```
initialize upgrade = 1;
while (task is time sensitive and upgrade = 1) {
    let previous user of current be current user;
    unit time cost of current user = unit time cost × (1 + $v$%);
    unit time cost of previous user = unit time cost × (1 − $w$%);
    if (both cost of current user and previous user do not decrease) {
        upgrade = 1;
    else {
        upgrade = 0;
        rescore current user bo be current user;
    }
}
```

Pseudocode 2

Table 1: Hardware configuration parameters.

Number	CPU	Amount	Memory (GB)
1	4-core, 3.07 GHz	10	4
2	4-core, 2.7 GHz	10	4

4. Experiments

4.1. Experimental Setup. We do experiments on two hardware configurations and the configurations are in Table 1. Both of the two hardware configurations run on CentOS5.8 and Hadoop-1.0.1.

There are total 20 computing nodes in our experimental environment, and each computing node starts up a virtual computing node. We start 10 schedulers, and each scheduler manages 2 virtual nodes (computing nodes). The application that we use in the experiments is WordCount.

According to (1) to (3), we computed the service intensity ρ, the expected number of tasks in a scheduler L, the expected length of queuing L_q, the expected finishing time of tasks W, and the expected waiting time of queuing W_q. Figure 2 describes the expected waiting time $T(w)$ on each scheduler. As we can see from the figure, the waiting time from schedulers 1, 3, 5, and 7 satisfied (14) and (23), and thus we can copy and execute some subtasks (data with size 1 KB) on them. If the user task is time sensitive, then we run the task on node with faster speed; and if the user is cost sensitive, then we run the task on node with lower cost.

4.2. Experiments for Time Sensitive User Utility Model. In order to select the parameters for time utility and cost utility functions, we normalize them first and get the following equations. Figure 3 describes the curves of the following two equations:

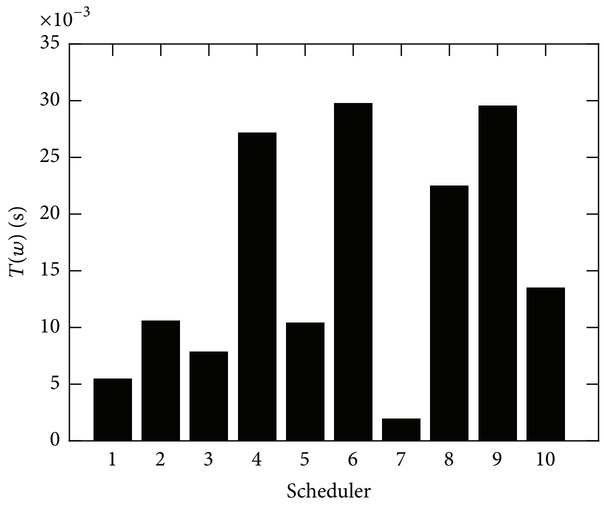


Figure 2: Expected waiting time for each scheduler.

$$U_t = \frac{8}{(\ln (t - 20) \times 5)}, \qquad U_c = -\left(\frac{1}{63}\right) \times c + \frac{61}{63}. \tag{29}$$

Based on running time and rate, total time, cost, and utility from schedulers 1, 3, 5, and 7, we set $a = 0.7$ and $b = 0.3$ in (10). Under constrains from (12) to (19), we compute the total utility. In Figure 4, U_t is the predicted time utility, U_c is the predicted cost utility, U' is the predicted total utility, U is

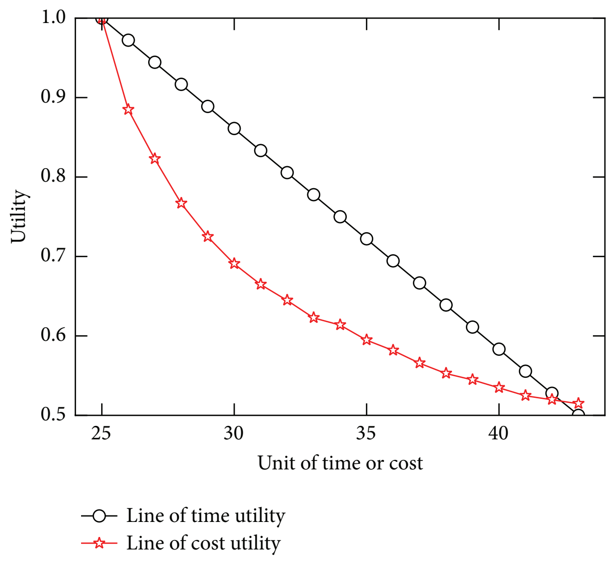


FIGURE 3: Time and cost utility lines for time sensitive user tasks.

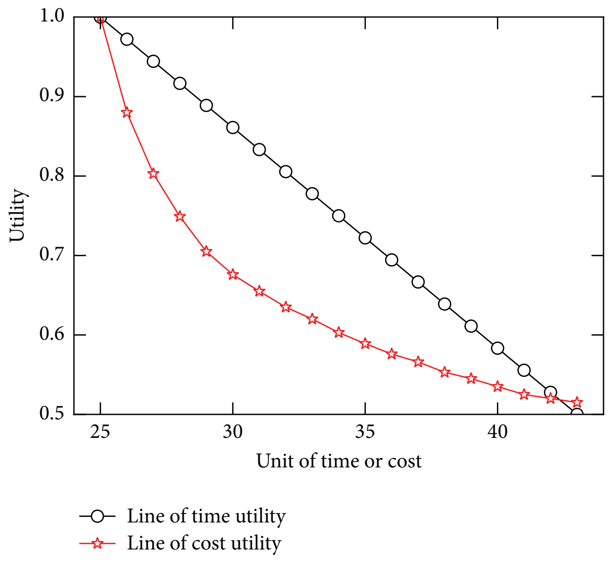


FIGURE 5: Time and cost utility lines for cost sensitive user tasks.

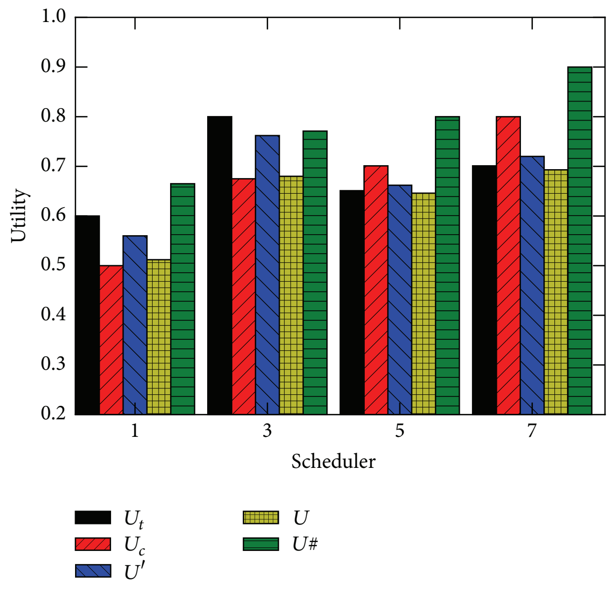


FIGURE 4: Utility distribution of time sensitive user tasks.

the actual total utility, and $U\#$ is the total utility that we get by rescheduling tasks on the above 1, 3, 5, and 7 schedulers.

In Figure 4, for scheduler 1, U_t, U_c, U', and U are all the lowest; for scheduler 3, U_t is the highest, U_c is much lower, and U' is the highest too; for schedulers 5 and 7, although their U_c is higher than scheduler, their U' is lower than scheduler 3. According to the rule of maximizing utility, we should choose scheduler 3 as the scheduler. However, in order to further improve the total utility, we applied the proposed algorithm in Section 3.3.3. By rescheduling the tasks in queuing, we get the actual total utility $U\#$ for each scheduler. In schedulers 5 and 7, $U\#$ is much higher than U' of scheduler 3.

4.3. Experiments for Cost Sensitive User Utility Model. In order to select the parameters of time utility and cost utility functions for cost sensitive user tasks, we also normalize them and get the following two equations. Figure 5 describes the curves of the following two equations:

$$U_t = -\left(\frac{1}{63}\right) \times t + \frac{61}{63}, \qquad U_c = \frac{8}{(\ln (c - 20) \times 5)}. \tag{30}$$

From Figure 6 we can see that the predicted total utility U' in scheduler 5 is the highest, and if we schedule tasks on scheduler 5, we would have the highest actual total utility $U\#$. So if user tasks have different time and cost requirements, we can choose different computing nodes to execute them and make the total utility maximal. In addition, after rescheduling the tasks, all tasks have higher actual total utility $U\#$ than predicted utility U' and actual total utility U, which validates the effectiveness of our proposed algorithm.

4.4. Comparison Experiments. In this experiment, we selected 10 simulating tasks and compared our algorithm with both Min-Min and Max-Min algorithms. The Min-Min algorithm schedules minimum task to the quickest computing node every time, and the Max-Min algorithm schedules maximum task to the quickest computing node every time. We implemented two algorithms for both time sensitive and cost sensitive user tasks and denoted them as MaxUtility-Time and MaxUtility-Cost. The experimental result is in Figure 7.

In Figure 7, the total utilities of MaxUtility-Time and MaxUtility-Cost algorithms are higher than the other two algorithms and are also stable; both Min-Min and Max-Min algorithms have lower total utilities, and their values fluctuate very much. Both Min-Min and Max-Min algorithms only consider the running time of tasks and ignore requirements of

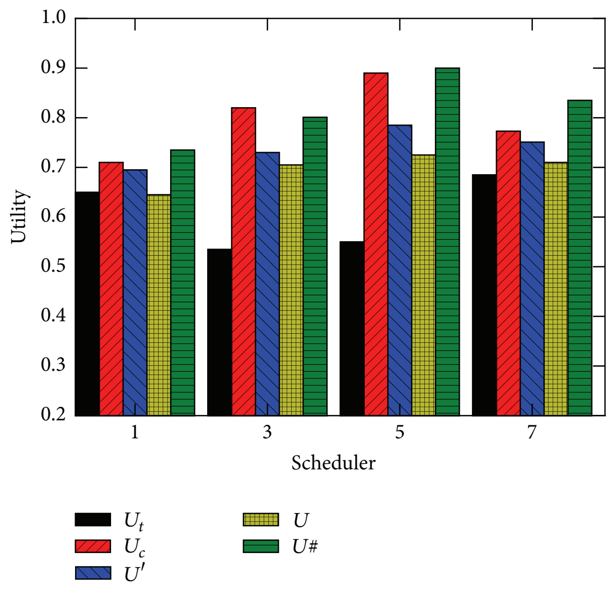


FIGURE 6: Utility distribution of cost sensitive user tasks.

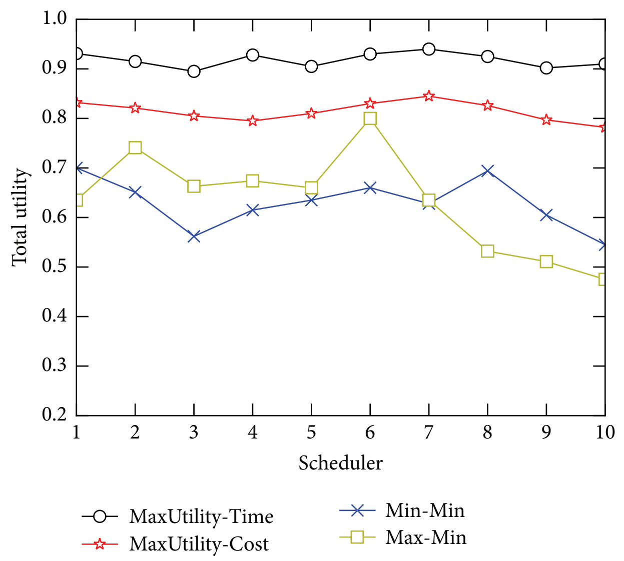


FIGURE 7: Comparison result for different algorithms under simulating tasks.

both time and cost, which makes them get lower total utilities and fluctuate very much. In particular, when running tasks 8, 9, and 10, total utility of the Max-Min algorithm drops quickly. The reason is that it schedules long-running tasks to computing nodes with high performance, which makes the utility very low.

5. Conclusion

In this paper, we introduced utility into the cloud environment, quantified the satisfaction of users to services as utility, and proposed utility oriented queuing model for task scheduling. We classified utility into time and cost utility, rescheduled tasks according to their remaining time, and minimized the total utility by constraints. With the proposed model, we can reschedule remaining tasks dynamically to get the maximum utility. We validated our proposed model by lots of experiments.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 826–831, IEEE, Melbourne, Australia, May 2010.

[2] C. S. Yeo and R. Buyya, "Service level agreement based allocation of cluster resources: handling penalty to enhance utility," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '05)*, pp. 1–10, Burlington, Mass, USA, September 2005.

[3] J. N. Silva, L. Veiga, and P. Ferreira, "Heuristic for resources allocation on utility computing infrastructures," in *Proceedings of the 6th International Workshop on Middleware for Grid Computing (MGC '08)*, pp. 93–100, ACM, Leuven, Belgium, December 2008.

[4] G. Song and Y. Li, "Utility-based resource allocation and scheduling in OFDM-based wireless broadband networks," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 127–134, 2005.

[5] T. T. Huu and J. Montagnat, "Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 612–617, IEEE, Melbourne, Australia, May 2010.

[6] Y. Yakov, "Dynamic resource allocation platform and method for time related resources," U.S. Patent Application 10/314,198[P], 2002.

[7] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.

[8] D. López-Pérez, X. Chu, A. V. Vasilakos, and H. Claussen, "Power minimization based resource allocation for interference mitigation in OFDMA femtocell networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 2, pp. 333–344, 2014.

[9] X. Wang and J. F. Martínez, "XChange: a market-based approach to scalable dynamic multi-resource allocation in multicore architectures," in *Proceedings of the 21st IEEE International Symposium on High Performance Computer Architecture (HPCA '15)*, pp. 113–125, IEEE, Burlingame, Calif, USA, February 2015.

[10] L. Thomas and R. Syama, "Survey on MapReduce scheduling algorithms," *International Journal of Computer Applications*, vol. 95, no. 23, pp. 9–13, 2014.

[11] D. Cheng, J. Rao, Y. Guo, and X. Zhou, "Improving MapReduce performance in heterogeneous environments with adaptive task tuning," in *Proceedings of the 15th International Middleware*

Conference (Middleware '14), pp. 97–108, ACM, Bordeaux, France, December 2014.

[12] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "SAMR: a self-adaptive mapreduce scheduling algorithm in heterogeneous environment," in *Proceedings of the 10th IEEE International Conference on Computer and Information Technology (CIT '10)*, pp. 2736–2743, IEEE, Bradford, UK, July 2010.

[13] D. Moise, T.-T.-L. Trieu, L. Bougé, and G. Antoniu, "Optimizing intermediate data management in MapReduce computations," in *Proceedings of the 1st International Workshop on Cloud Computing Platforms (CloudCP '11)*, pp. 37–50, ACM, Salzburg, Austria, April 2011.

[14] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency Computation Practice and Experience*, vol. 14, no. 13–15, pp. 1507–1542, 2002.

[15] B. Xu, C. Zhao, E. Hu, and B. Hu, "Job scheduling algorithm based on Berger model in cloud environment," *Advances in Engineering Software*, vol. 42, no. 7, pp. 419–425, 2011.

[16] S. Yeo and H.-H. S. Lee, "Using mathematical modeling in provisioning a heterogeneous cloud computing environment," *Computer*, vol. 44, no. 8, Article ID 5740825, pp. 55–62, 2011.

[17] M. H. Rothkopf, "Scheduling with random service times," *Management Science*, vol. 12, no. 9, pp. 707–713, 1966.

[18] R. H. Möhring, A. S. Schulz, and M. Uetz, "Approximation in stochastic scheduling: the power of LP—based priority policies," *Journal of the ACM*, vol. 46, no. 6, pp. 924–942, 1999.

[19] N. Megow, M. Uetz, and T. Vredeveld, "Models and algorithms for stochastic online scheduling," *Mathematics of Operations Research*, vol. 31, no. 3, pp. 513–525, 2006.

[20] M. Scharbrodt, T. Schickinger, and A. Steger, "A new average case analysis for completion time scheduling," *Journal of the ACM*, vol. 53, no. 1, pp. 121–146, 2006.

[21] X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model," in *Proceedings of the 3rd IEEE International Workshop on Multimedia Signal Processing (MMSP '11)*, pp. 1–6, Hangzhou, China, November 2011.

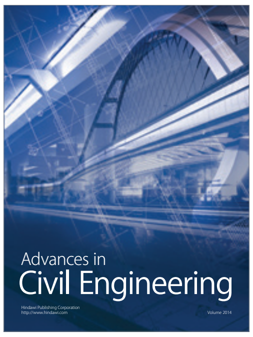
[22] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

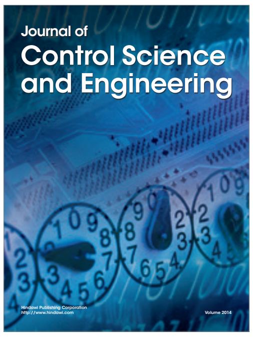
Journal of
Engineering

The Scientific World Journal

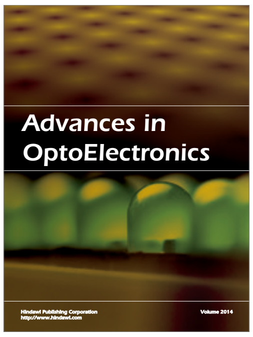
International Journal of
Rotating Machinery

Journal of
Sensors

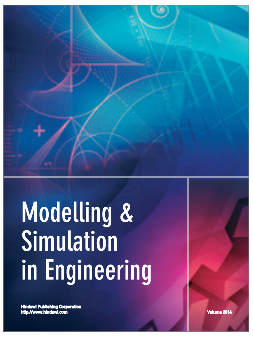
International Journal of
Distributed Sensor Networks

Advances in
Civil Engineering

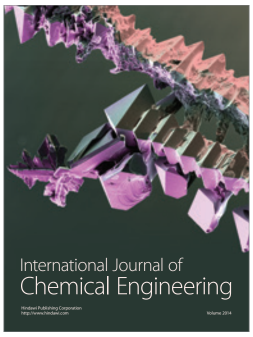
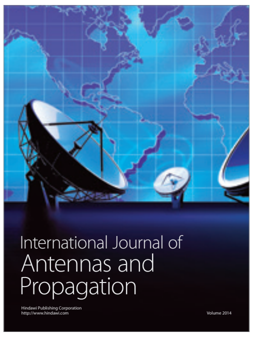

Journal of
Control Science and Engineering

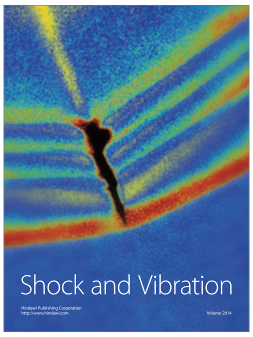
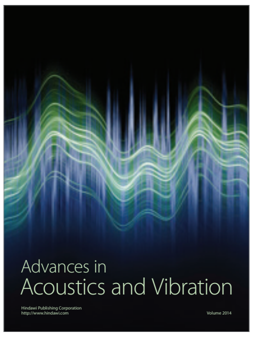
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration