2017 First IEEE International Conference on Robotic Computing

A Load-Based Scheduling to Improve Performance in Cloud Systems

Yi-Ju Chiang[1], Yen-Chieh Ouyang [1]*, Armin B. Cremers[2], Liangyu Xu[2]
[1]Department of Electrical Engineering
National Chung Hsing University
Taichung, Taiwan, ROC
{yjchiang0320@gmail.com}{ycouyang@nchu.edu.tw}
[2] Institute of Computer Science
University of Bonn
Bonn, Germany
{abc, xul}@ iai.uni-bonn.de

Abstract—**Cloud computing is a type of parallel and distributed system for dynamically provisioning on-demand services. Customers and enterprises can employ cloud services including software, platforms and infrastructure to improve the scalability of their services and handle stochastic demands from users. However, random arrival tasks from users have different load sizes and deadline requirements that need to be satisfied in a cloud system. How to schedule tasks to virtual machines under heavy traffic load still remains a challenge problem for cloud service providers. If high-load tasks occupy shared resources for a long time, it will cause higher waiting times for remaining tasks, which make other tasks fail to be completed within their deadline requirements. In this paper, a load-based task scheduling approach is proposed to improve the system performance when facing heavy-tailed traffic. The main purpose is to schedule arrival tasks effectively and prevent high-load tasks from occupying in the same VM group. A series of experiments and performance analyses are conducted by using the CloudSim simulation tool to evaluate the proposed approach. We also compare the experimental results with other approaches under various situations. Simulation results show that the proposed approach outperforms other approaches in terms of response times and profit values.**

Keywords—Task scheduling; Heavy-tailed distribution; Deadline requirement; Dynamic programming; CloudSim simulation

I. INTRODUCTION

Cloud computing is a new way to deliver services such as infrastructure, platforms, and software. These services are also respectively referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Cloud services enable customers to lease computational resources for temporarily needed. It eliminates the need to maintain expensive computing hardware, dedicated space, and software as compared to traditional "own-and-use" patterns. The gradually increasing big data including healthcare, science, engineering, and enterprise applications, etc. are being developed and stored in cloud computing systems to provide quality of services (QoS). Healthcare cloud service systems are also built to improve the security of medical data and personal information [1], [2].

*Supported by the German Academic Exchange Service (DAAD) Sandwich Model Fellowships and the Bonn-Aachen International Center for Information Technology (B-IT institute).

For example, the e-Healthcare [3] cloud plays the role of implementing various e-services among all participants such as general users, patients, doctors, secondary users, and health insurance companies. However, the variation in task load sizes makes task scheduling become a complicated work, especially under heavy traffic load. This is due to the fact that a majority of normal tasks will be severely delayed when a small part of high-load tasks occupy shared computing resources [4]. It may also cause these tasks that are queued at the back fail to be completed within their various deadline requirements. In other words, few high-load tasks would lead to starvation of other tasks, and result in fewer "successfully completed tasks". Furthermore, more and more evidences [5], [6], [7] have showed that a task size distribution in computer loads have heavy-tailed and long-tailed properties instead of traditional normal distributions.

Heavy-tailed distributions have been increasingly observed and studied in computer workloads which coincide with the task load size in supercomputers and internet traffic. Previous works had presented evidences that these phenomena were well modeled by using heavy-tail distributions [8]. In modern complex networks and computer systems, the heavy-tailed phenomenon is also a ubiquitous property. However, during a peak-load period, it is impossible for a cloud system to process all incoming heavy-tailed traffic. Scheduling tasks improperly will cause not only higher waiting times but also fail to meet deadline requirements of tasks in a cloud system. Besides, since task arrival pattern is not predictable and the workload capacities of virtual machine (VM) groups differ from each other, controlling workload still remains a crucial issue in a cloud system. The amount of tasks scheduled to a VM group directly depends on the workload limitation. In order to improve system performances and load balancing [9], tasks that are scheduled to VM groups with different workload constraints cannot be dispatched randomly.

To solve complex constrained optimization problems for a workload control in the cloud system, a sequential decision processes can be employed. For instance, the dynamic programming approach [10], [11] is often used to address constrained control problems by breaking a complex problem down into a collection of sub-problems. The remaining

978-1-5090-6724-4/17 $31.00 © 2017 IEEE
DOI 10.1109/IRC.2017.28

decisions can be resolved with regard to the state resulting from the previous decision.

To schedule incoming tasks under heavy-load traffic, task load sizes, deadline requirements, profit values and workload constraint of the target VM group are all required to be taken into account. To address this problem, this paper presents a load-based task scheduling approach combined with a workload control to improve the system performance when facing heavy-tailed traffic. The designed scheduling approach is implemented and tested by using the CloudSim 2.0 toolkit, so as to conduct an in-depth analysis under a series of experiments. The remainder of this paper is structured as follows: Section 2 gives a brief overview of existing works on the task blocking and rejection approaches. Section 3 introduces the cloud service system and heavy-tailed distribution model. Section 4 presents the designed task scheduling approach. Section 5 compares the proposed approach with existing algorithms and evaluates the simulation results. Our findings and conclusion are summarized in Section 6.

II. Related Work

To prevent heavy-traffic load from overloading cloud service systems, several proposals [12], [13], [14] such as rejection policies, blocking methods, etc. had been put forward. In [12], Atdelzater, Atkins, Shin presented a novel scheme for QoS negotiation in real-time applications. An admission control schemes (which do not support negotiated QoS degradation) which would always incur the request rejection penalty was designed whenever an arrived task made current tasks un-schedulable. A designed negotiation model was centered on three simple abstractions: QoS levels, rewards, and rejection penalty. The rejection penalty of a client's request was the penalty incurred to the application if the request was rejected.

In [13], Ghosh, et al. used a stochastic reward net based model for provisioning and servicing requests in an IaaS cloud. If none of these servers were available, the request was rejected. There were two components of calculating the net job rejection rate. The first component was admission control resulted from rejecting jobs when the buffer was full. Second component of the rejection rate was that after job was admitted in the queue, they assumed that the job was dropped if all machines (hot, warm and cold) were fully occupied.

In [14], Calheiros, Ranjany, and Buyya developed a provisioning technique that automatically adapted to workload changes related to applications for facilitating the adaptive management of system. High task arrival rates would lead to service rejections. The provisioning technique tried to meet QoS targets such as response time and service requests rejection, while preventing over-provisioning of IT resources. In [15], Stankovic, et al. developed software control algorithms based on a theory of feedback control to distributed systems. Task rejection was deemed the same as missing the task's deadline. Admission control was based on estimated CPU utilization and could admit or reject tasks from the outside. The service level actuator modified the service level of tasks in the system to the appropriate level and the reject control actuator would abort tasks by using the reject control if necessary.

In [16], Anandharajan and Bhagyaveni calculated the completion time of job in the designed job selection algorithm when the resource was overloaded. The Dynamic Scheduling algorithm using Boundary Value approach was applied if the resource was overloaded. In [17], Shen, et al. formulated the resource provisioning and job allocation policies as integer programming problems. The designed algorithm sorted all the un-allocated jobs by using job selection criteria including the first-come-first served (FCFS), round robin (RR), Smallest job first (SJF), Largest Job Fist Served (LJFS), Shortest run-Time job first (STJF) for job selection criteria. However, previous works did not take the variability of task load size into consideration, which will cause urgent tasks to be severely delayed by large tasks and lost profit from those disrupted tasks, especially under heavy-tailed traffic.

III. System Model

A. A Cloud Service System

Recently, more and more Cloud-based business models and enterprise models are presented to help organizations and companies save operational costs, meet enterprise demands [18] and provide scalable resources provisioning for their customers when employing virtualized application services. The increasing healthcare services also adopt cloud systems to improve mobility, economical savings and efficiency.

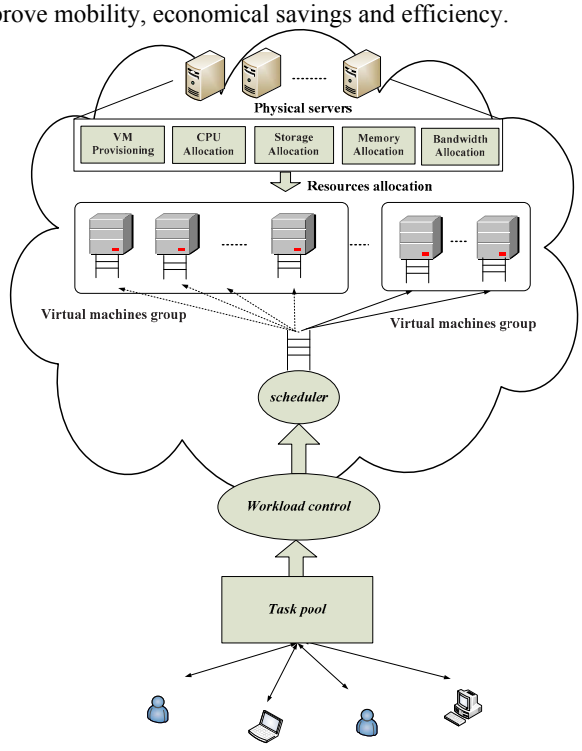

Figure 1. A private cloud service system with a workload control mechanism.

Figure 1 illustrates a simple model of a cloud service system and shows the main components including the physical

servers, virtual machines, CPU allocation, memory, storage and bandwidth allocation. The physical servers will produce different groups of VMs with various workload constraints to process these arrival tasks. There are various customers on the client side [1], [2], the involved parties may include urgent users, high-demand tasks, general customers, etc. with different deadline requirements. Therefore, the cloud task scheduler is responsible to schedule arrival tasks efficiently in order to meet users' performance requirements and prevent the cloud system from overloading during peak-load periods.

B. Pareto Distribution Model

Heavy-tailed task size distributions have the property that the vast majority of tasks are small, while very high-load demands are requested by a tiny minority of tasks. In our service model, we focus on low-frequency tasks with high-load demands because they tend to have higher impact than high-frequency tasks with smaller-load sizes. A random variable X follows a heavy-tailed distribution [19] if

$$P[X > x] \sim x^{-k} \quad for\ 0 < k < 2, as\ \ x \to \infty \qquad (1)$$

In our proposed model, task instances are assumed to be independent and identically distributed random variables. One of the most commonly used methods to model heavy-tailed distributions is the *Pareto* distribution whose probability density function is given by:

$$p(x) = \begin{cases} 0, & if\ \ x < x_b \\ \dfrac{k x_b^{\,k}}{x^{k+1}}, & if\ \ x > x_b \end{cases} \qquad (2)$$

Where k is the shape parameter (also known as the tail index) and x_b is the scale parameter of the *Pareto* distribution.

The cumulative distribution function (CDF) of the *Pareto* distribution model can be obtained as

$$F(x) = \begin{cases} 1 - \left(\dfrac{x_b}{x}\right)^k & x \geq x_b \\ 0 & x < x_b \end{cases}. \qquad (3)$$

The probability that X is greater than the number x is given by

$$P(X > x) = \begin{cases} \left(\dfrac{x_b}{x}\right)^k & x \geq x_b \\ 1 & x < x_b \end{cases}. \qquad (4)$$

which is also called a tail function.

IV. Analytical Approaches

To design an effective task scheduler in the cloud service system, the load variations, deadline requirements and profit values must be taken into account before scheduling tasks. In the proposed model, all arrival tasks are collected in the task pool and tasks that are featured by different properties will all be recorded at per planning period. Here we let the notation L_i and R_i denote the task load size and deadline requirements of a task i, where $i \in \{1, 2, \ldots, m\}$ respectively. The m denotes the total amount of tasks in the task pool and the notation V_i denotes the profit value of a task i.

The deadline requirement R_i is assumed to have a proportional relationship with the load size L_i in a general situation. To prevent the cloud system from overloading caused by a heavy traffic load coming into the VM group, the cloud scheduler needs to control an arrival workload that can be scheduled to a target VM group at per planning period. Therefore, an objective function subjected to a workload constraint is defined as below.

$$\begin{aligned} &Maximize && \textstyle\sum_{i=0}^{i=m} V_i \times t_i \\ &subject\ \text{to} && \textstyle\sum_{i=0}^{i=m} L_i \times t_i \leq workload\ constraint \qquad (5) \\ &and && t_i \in \{0, 1\} \end{aligned}$$

The main purpose is to increase profit as much as possible from these chosen tasks, while the total workload cannot exceed the predefined constraint at per planning period. However, this problem is not the same as the linear programming since it has a new integer constraint.

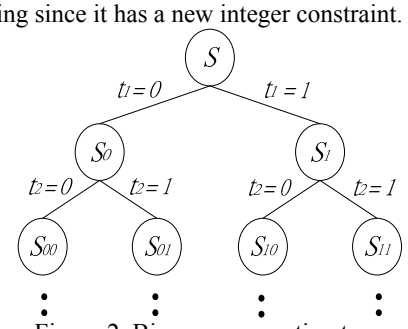

Figure 2. Binary enumeration tree.

Brute-force or exhaustive search method is a very general problem-solving technique to search all possible candidates for the solution. As shown in figure 2, by assuming that each variable is only 0 or 1 valued, it would generate the whole solution tree, where every path from the root to any leaf is a solution set, denoted as *S*. However, enumerating all possible candidates for the solution is computationally unfeasible. To speed up the search process, the branch and bound approach is applied.

A. Branch and Bound Approach

The Branch and Bound (B&B) [20] is one of the most widely used tools for solving large scale NP-hard combinatorial optimization problems. The branch-and-bound algorithm can be viewed as a tree search and proceed by branching from all possible solutions into smaller subclasses. A bounding operation is used to find out the feasibility of current configuration. If the subclass of solutions from a node is infeasible, then the node will be seen as an "unpromising" node and further branching from that node is bounded.

Figure 3 shows an example of the branch and bound search tree by assuming that each variable is only 0 or 1. In this case, after selecting the task1 $\{t_1=1\}$, we need to decide whether the current solution is feasible or not. If the result of going to select the task2 $\{t_2=1\}$ will exceed the predefined bound, then we are not going to make the decision of expanding the node, so as to reduce the state space by pruning the tree.

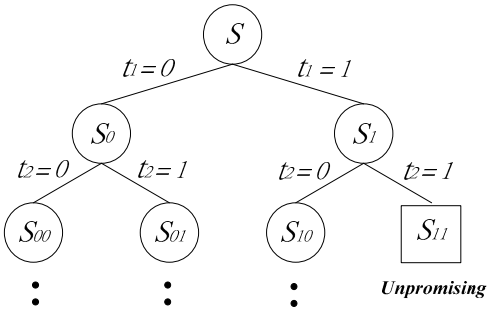

Figure 3. Branch-and-bound search tree

B. Dynamic Programming

Dynamic programming [21] is a mathematical optimization method and also a computer programming method. It can be used to solve a variety of problems analytically that cannot readily be treated by either method alone. Dynamic programming has attracted considerable research attention. In [22], Shahzad and Szymanski proposed an innovative dynamic programming algorithm to find a nearly optimal offloading solution quickly. The algorithm would offload as many tasks as possible to the cloud when the network transmission bandwidth was high. In [23], Liu and Lee presented a Dynamic Programming based Offloading Algorithm (DPOA) to quickly find the optimal partitioning between executing subcomponents of a mobile application at the mobile device and the cloud server. In [24], the optimization method used to solve the spatial domain optimal control problem, and run on the cloud computer, was also based on dynamic programming, which gave the global optimal solution over a given driving route.

Dynamic programming which transforms a complex problem into simpler problems has two ways to compute the global optimal solution. One is top-down approach which solves an optimization problem by starting from the top-level problem and breaking it down into a small portion of sub-problems. The other is bottom-up, which recursively divides problem into a set of sub-problems, and then solutions can be solved trivially until it hits the smallest sub- problem.

C. Load-based Scheduing

In designing the task scheduling approach in our service system, one of the main purposes is to avoid high-load task occupying the shared resources. Therefore, we not only take the goal of profit optimization, but also take the task size into consideration. The flowchart of the proposed load-based scheduling approach is shown in figure 4. First, we can obtain the information of load size, deadline requirement and profit value of each arrival task in the task pool. Then, we start to examine tasks that have higher profit values with smaller load sizes than others. That is, at each stage of W_j, where $j \in \{1, 2, \ldots, t\}$ in the decision-making process, tasks with a higher V_i/L_i value than others will be chosen as the optimal solution when the sum of workload is no larger than the workload constraint at that time. This process will continue until the workload constraint has been added up to the maximum threshold value W_t.

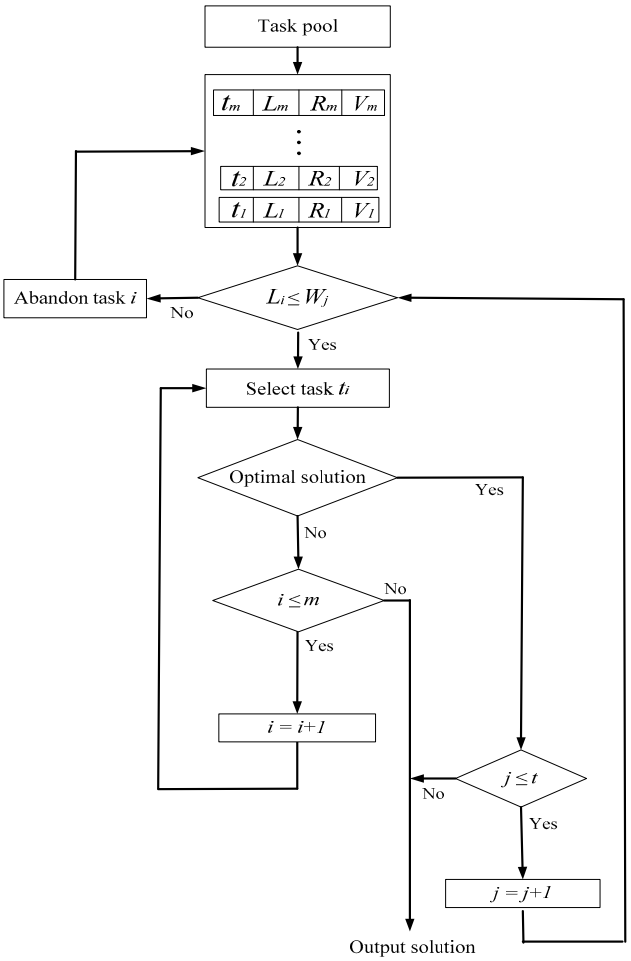

Figure 4. The flowchart of the load-based scheduling approach.

The problem-solving steps are shown as follows:

Step 1: Decomposes the problem into smaller problems.

Step2: Starts from the beginning problem and solves the smaller problems iteratively.

Step3: Computes the optimal solution in a bottom-up way.

Step4: After obtaining these chosen tasks, the queue discipline of EDF (Earliest Deadline First served) is used in order to give priorities to tasks in the queue according to their deadline requirements.

V. SIMULATION RESULTS

In the following simulation, the system modeling, experiments and performance testing are conducted by using the CloudSim simulation tool [25], so as to examine the feasibility of the proposed scheduling approach. The CloudSim provides simulation of modeling a virtualization engine with multiple virtualized services and supports dynamic creation of different cloud scenarios. The major benefit is that the system bottlenecks can be fine-tuned before these designed approaches are actually deployed in a real Cloud system. In this experiment, a data center was created and the simulation details of the environment and parameters setting are described in Table 1.

TABLE 1
Simulation parameters

Architecture	x86
Operating System	Linux
Physical machine Memory	4G
Physical machine Storage	1TB
Physical machine core	8 cores
Number of VMs	5
Virtual Machine Monitor (VMM)	Xen
VM workload size	10000 MB (image workload size)
VM memory	512 MB
Processor speed	1000 MIPS
Bandwidth	10000 Bytes per second
VM allocation policy	Space-Shared

The load size of tasks L_i are generated from a random Pareto distribution with a location parameter 500 and shape parameters 1.9 and 2.1. The dynamic inter-arrival time of task is exponentially distributed with a mean value of 0.5. In a general situation, the deadline requirement and profit value may have a positive, proportional relationship with its task load size; hence, the deadline requirements are set as $R_i=L_i/60$ and the profit value are set as V_i= (random$(0,5)+L_i)/100$ for the task i, respectively in this study case. Here 150 tasks characterized by different properties are generated randomly and then submitted to the cloud task pool.

We create 5 VMs in the VM group to execute these tasks which are scheduled by the dynamic programming (denoted as DYP), the Branch and Bound algorithm (denoted as B&B), and the designed load-based scheduling approach (denoted as LS). The workload constraint of w=75000 in the VM group are tested. Figure 5 shows the comparison result of the response time by applying the DYP, the B&B and the LS approaches when the shape parameter is 1.9. The comparison results show that the response times increase significantly when the numbers of tasks increase in the system by adopting the DYP and the B&B approaches.

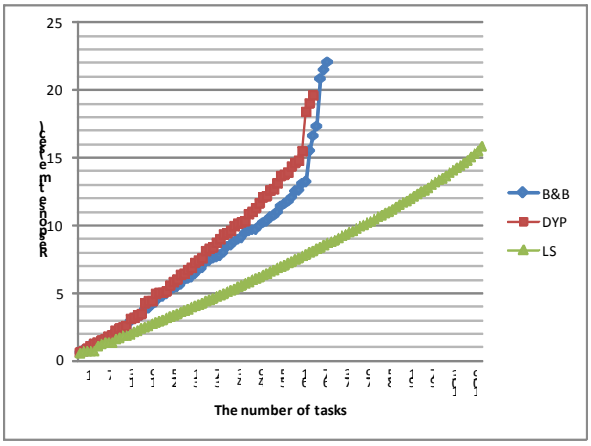

Figure 5. The comparison of response times by using different scheduling approaches when the shape parameter is 1.9.

It also can be noted that the designed LS approach accepts more tasks than the DYP and B&B methods. This is due to the fact that the LS approach will examine tasks and select tasks that have higher profit values with smaller load sizes than others instead of using their pure profit values. Therefore, a task that has a higher profit value with a larger load size is not necessarily to be selected even when the amount of workload is under its constraint at that time. The number of tasks selected by the LS approach is 113, while others are 70 and 66, respectively by using the DYP and B&B approaches. The value of deadline satisfation evaluated by subtracting the response time from the deadline requirement of each task is shown in figure 6. As can be seen, the LS approach can schedule more tasks and complete these selected tasks within their deadline requirements. However, even if the amount of incoming workload that can be scheduled into the VM group has been controlled, few tasks fail to meet their deadline requirements by adopting the DYP and the B&B approaches. In this case, there have 8 and 11 tasks that fail to be completed in the VM group; that is, the failure rates are 11.42% and 16.66 % by adopting the DYP and the B&B approaches, respectively.

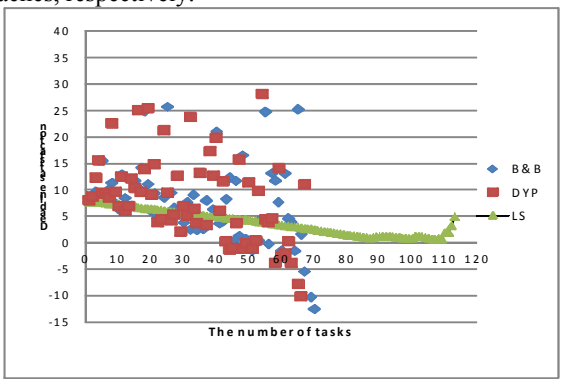

Figure 6. The comparison of deadline satisfation by using different scheduling approaches when the shape parameter is 1.9.

In a general situaiton, the cloud system will lost profit from those unsucessful tasks that VMs can not complete them within their deadline requirements. Figure 7 shows the comparison of profit values including the predicted value as well as the final value obtained by using different scheduling approaches. The predicted value is the total profit obtained from these scheduled tasks, while the final profit is those successful tasks after being excuted. It can be noted that the DP and the B&B approaches can obtain higher predicted profit values of 7489 and 7520, respectively than the LS approach by choosing more tasks with higher profit in the beginning. However, after excuting their selected tasks, the final profit values are 6928 and 6807, respectively by adopting the DP and the B&B approaches, which are lower than the LS approach.

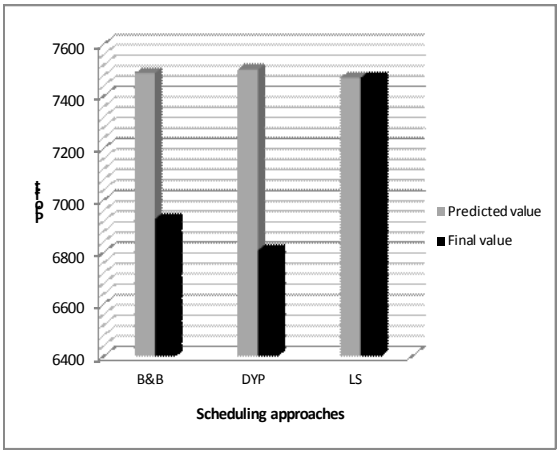

Figure 7. The comparison of predicted and final profit values when the shape parameter is 1.9.

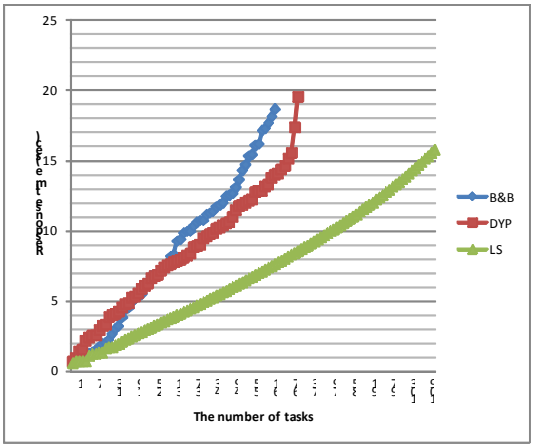

Figure 8. The comparison of response times by using different scheduling approaches when the shape parameter is 2.1.

Figure 8 shows the response times of tasks by adopting the DP, the B&B and the LS approaches when the shape parameter is 2.1. The number of tasks scheduled by the DP and the B&B are 63 and 70, while the LS approach selects 112 tasks to process in the VM group and it also results in lower response time than others. It can be noted that the DP approach and the B&B approach can obtain higher estimated profit values of 7497 and 7503, respectively than the LS approach, as shown in figure 9. However, the real profit values of 6502 and 7021 obtained by the B&B and the DP approaches, respectively, are lower than the LS approach.

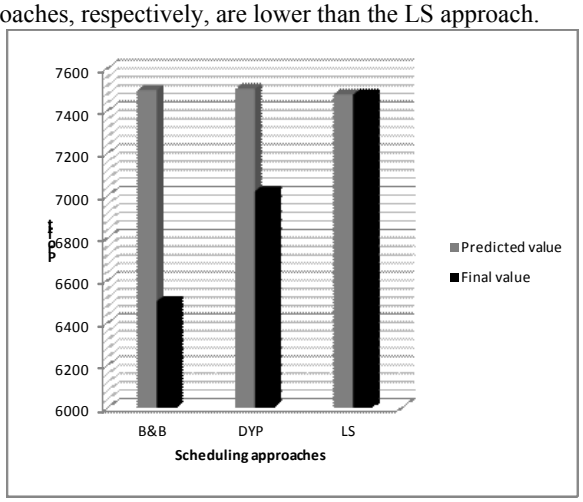

Figure 9. The comparison of predicted and final profit values when the shape parameter is 2.1.

Next, we try to test how the proposed approach performs on a single VM. A workload constraint of w=30000 in a single VM is considered in our experiments. Here we provide a comparison study by using the Rejection Policy (denoted as RP) to control the amount of workload that can be scheduled to a single VM. The RP approach will schedule tasks in the order of their arrivals and start to reject tasks when all space and resources in the VM have been occupied. The comparison of response time by using the RP, DP, B&B and LS approaches when w=30000 are shown in figure 10.

The performance testing results show that the LS approach also can obtain lower response time even though it schedules more tasks than others. Tasks scheduled by the DP and the B&B approaches fail to meet their deadline requirements even though they select fewer tasks to a VM within a restricted workload. Besides, the RP approach results in lower predicted profit than others since these tasks are selected only based on the workload constraint, as shown in figure 11.

For the DP and the B&B approaches, they mainly select tasks with higher profit to meet the objective of maximizing profit, so both can obtain higher predicted profit values. However, few selected tasks fail to complete within their deadline requirements, so both result in lower final profit values. Conversely, the LS can prevent high-load tasks from occupying in the same VM under heavy traffic condition. Although it obtains a lower predicted profit value than the B&B and the DP approaches, these scheduled tasks are all been completed successfully within their deadline requirements, so as to obtain a higher profit value.

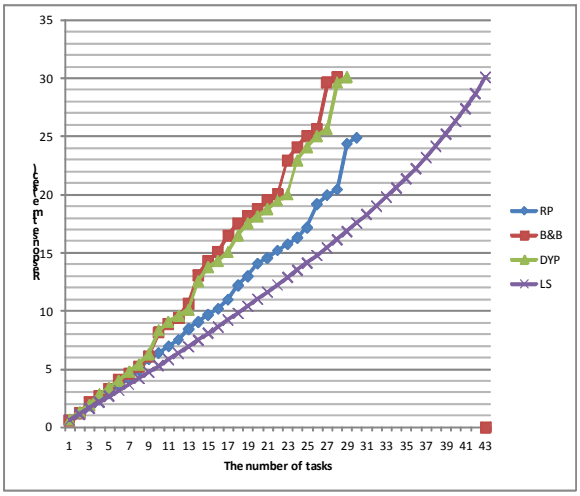

Figure 10. The comparison of response time by adopting the RP, DP, B&B and LS approaches when w=30000.

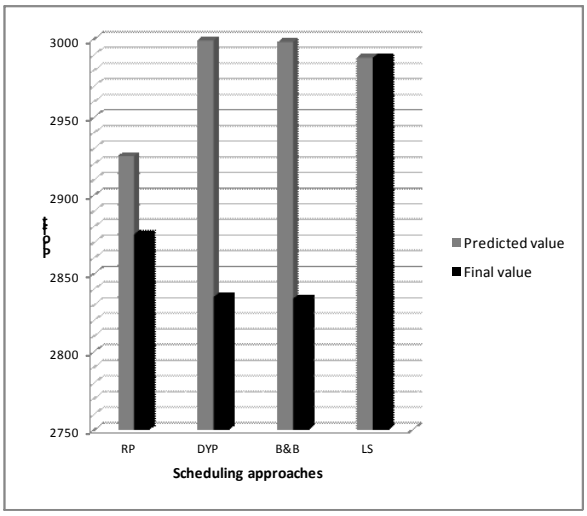

Figure 11. The comparison of profit value when w=30000.

VI. CONCLUSION

Scheduling tasks plays a significant role in improving performances and profit in a cloud computing system, especially when the system is under a heavy traffic load. In this paper, the LS approach is proposed to schedule tasks by taking load sizes, profit values and VM workload constraints into consideration. The main goal is to maximize profit and improve performances when the system is under heavy-tailed traffic. The challenge of how to schedule arrival tasks with various load sizes and deadline constraints is solved. Numerical experiments and simulation are conducted to validate our proposed approach. The CloudSim simulator is used to build the simulation model and implement task scheduling approaches. Simulation results show that the proposed LS approach outperforms other approaches in terms of reducing response times and enhancing profit values.

REFERENCES

[1] L. Xu, and A. B. Cremers, "A Decentralized Pseudonym Scheme for Cloud-based eHealth Systems", Proc. International Conference on Health Informatics, pp. 230-237, 2014.

[2] L. Xu, and A. B. Cremers, "Patients' Privacy Protection against Insurance Companies in eHealth Systems", in: K. Bernsmed, S. Fischer-Hübner (Eds.) Secure IT Systems, Springer International Publishing, pp. 247-260, 2014.

[3] Patara F., and Vicario E. "An adoptable patient-centric Electronic Health Record system for personalized home care," In Medical Information and Communication Technology (ISMICT), 8th International Symposium on, pp. 1-5, 2014.

[4] M. E. Crovella, M. S. Taqqu, and A. Bestavros, "Heavy-tailed probability distributions in the World Wide Web," A practical guide to heavy tails, 1, pp. 3-26, 1998.

[5] I. Taboada, J. O. Fajardo, F. Liberal, and B. Blanco, "Size-based and channel-aware scheduling algorithm proposal for mean delay optimization in wireless networks," In Communications (ICC), IEEE International Conference on, pp. 6596-6600, 2012.

[6] D. G. Feitelson, "Workload modeling for performance evaluation," In Performance Evaluation of Complex Systems: Techniques and Tools , pp. 114-141. Springer Berlin Heidelberg, 2002.

[7] B. Fu, J. Broberg, and Z. Tari, "Task assignment strategy for overloaded systems," In Computers and Communication, 2003.(ISCC 2003). Proceedings. Eighth IEEE International Symposium on, pp. 1119-1125, 2003.

[8] A. Nahir, A. Orda, and D. Raz, "Distributed oblivious load balancing using prioritized job replication," In Proceedings of the 8th International Conference on Network and Service Management, pp. 55-63. International Federation for Information Processing, 2012.

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

[10] R. C. Chen, and G. L. Blankenship, "Dynamic programming equations for discounted constrained stochastic control," Automatic Control, IEEE Transactions on, 49(5), pp. 699-709, 2004.

[11] X. Lin, H. Zhang,L. Wei, and H. Liu, " Optimal control for industrial sucrose crystallization with action dependent heuristic dynamic programming," In Intelligent Control and Automation (WCICA), 2010 8th World Congress on, pp. 656-661, 2010.

[12] T. F. Atdelzater, E. M. Atkins, and K. G. Shin, "QoS negotiation in real-time systems and its application to automated flight control," IEEE Transactions on Computers, 49(11), pp. 1170-1183, 2000.

[13] R. Ghosh, F. Longo,V. K. Naik, and K. S. Trivedi, "Quantifying resiliency of iaas cloud," In Reliable Distributed Systems, 29th IEEE Symposium on, pp. 343-347, 2010.

[14] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," In Parallel Processing (ICPP), 2011 International Conference on, pp. 295-304, 2011.

[15] J. A. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu, "Feedback control scheduling in distributed real-time systems," In Real-Time Systems Symposium, (RTSS 2001). Proceedings. 22nd IEEE, pp. 59-70. IEEE, 2001.

[16] T. R. V. Anandharajan, and M. A. Bhagyaveni, "Co-operative scheduled energy aware load-balancing technique for an efficient computational cloud," International Journal of Computer Science, (8), pp. 571-576, 2011.

[17] S. Shen, K. Deng, A. Iosup, and D. Epema, "Scheduling jobs in the cloud using on-demand and reserved instances," In Euro-Par 2013 Parallel Processing, Springer Berlin Heidelberg, pp. 242-254, 2013.

[18] V. Chang, and D. Bacigalupo, G. Wills, and D. De Roure, " A categorisation of cloud computing business models," In Proceedings of the 2010 10th ieee/acm international conference on cluster, cloud and grid computing, pp. 509-512, 2010.

[19] M. E. Crovella, A. Bestavros, "Self-similarity in World Wide Web traffic: evidence and possible causes." IEEE/ACM Transactions on networking 5.6, pp. 835-846, 1997.

[20] J. Clausen, "Branch and bound algorithms-principles and examples," Department of Computer Science, University of Copenhagen, pp. 1-30, 1999.

[21] D. P. Bertsekas, and D. P. Bertsekas, "Dynamic programming and optimal control," Belmont, MA: Athena Scientific, 1995.

[22] H. Shahzad, and T. H. Szymanski, "A dynamic programming offloading algorithm for mobile cloud computing," IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1-5, 2016.

[23] Y. Liu, and M. J. Lee, "An effective dynamic programming offloading algorithm in mobile cloud computing system," In 2014 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1868-1873, 2014.

[24] J. Wollaeger, and S. A. Kumar, S. Onori, D. Filev, Ü. Özgüner, G. Rizzoni, and S. Di Cairano, "Cloud-computing based velocity profile generation for minimum fuel consumption: A dynamic programming based solution," IEEE American Control Conference (ACC), pp. 2108-2113, 2012.

[25] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, " CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, 41(1), pp. 23-50, 2011.