Utility-based Server Management Strategy in Cloud Networks

Eunhye Choi*, Suin Song*, Hyejin Kim*, Jiyeon Hong*, Hyunggon Park* and Mihaela van der Schaar†

*Multimedia Communications and Networking Laboratory
Department of Electronics Engineering, Ewha Womans University, Seoul, Korea
†Networks, Economics, Communication Systems, Informatics and Multimedia Research Laboratory
Electrical Engineering Department, University of California, Los Angeles (UCLA), Los Angeles, USA
{ceh1990, ssin87, 0915039, jiyeons77}@ewhain.net, hyunggon.park@ewha.ac.kr, mihaela@ee.ucla.edu

***Abstract*—Numerous real-time applications are currently deployed over mobile networks - ranging from multimedia streaming, to real-time online games, to video conferences/chat, to real-time stock exchanges, etc. Supporting such applications is challenging because they have very stringent Quality of Service (QoS) requirements in terms of both throughput and delay. To address this challenge, in this paper, we propose to assist such mobile applications by a cloud-based network environment, which consists of multiple servers and clients (users). In cloud networks, servers can create multiple replica of the popular content in order to provide the needed QoS for the various service requests of the users. However, in order to efficiently provide services to the potentially large amount of service requests, it is essential for servers to strategically respond to the requests from users. Since the strategic response involves the resource management of the servers, we design a strategy which explicitly considers the service requests (e.g., service types, data priorities, delay constraints) as well as the servers' resource usages (e.g., current loads of servers, contributions of servers). Since the servers are strategic, they will aim to maximize their own utilities. Simulation results verify that the proposed approach enables servers to manage their resources more efficiently compared to existing approaches, thereby providing prioritized data processing and operating in a desired range and leading also to an improved performance for the users.**

***Index Terms*—Cloud networks, server utility, data priority, server contribution**

I. Introduction

A plethora of emerging mobile devices has been recently exploding and various mobile applications such as multimedia streaming, real-time online games, video conferences/chat, real-time stock exchanges, etc. have been widely serviced over networks. This leads to an extremely large volume of data exchanges in networks. In order to guarantee Quality of Service (QoS) requirements for such services, one of key features is to efficiently store and deliver data in timely manner [1]. Moreover, cloud networks have been considered as a promising solution that provide a variety of services in forms of software, storage, servers, etc.

Cloud networks consist of multiple servers that are designed to provide services to users. Servers can create several replica of content and exchange them to support various service requests from users. Practical examples include Google File System (GFS) [2] and Hadoop [3]. In these systems, if arbitrary data is entered as an input, the data is divided into several pieces, and a Central Management Server (CMS) stores the pieces in a distributed manner based on the server status. When the data is needed, several pieces of the data are downloaded simultaneously from the distributed servers. Therefore, monitoring resources in the cloud networks is especially important in order to guarantee the QoS. A survey on cloud monitoring and its related issues can be found in [4], and several approaches of guaranteeing system performance are studied in [5]. The major problem associated with CMS is in the lack of scalability that the entire system can be vulnerable when significant amount of data is requested simultaneously, or when the CMS fails to operate.

Distributed approaches have been widely studied as a solution to these problems In [6], a network resource aware strategy that minimizes the latency between numerous data centers is proposed. For better data exchanges among servers in cloud networks, a file exchange strategy deployed in peer-to-peer (P2P) networks is adopted [7], [8]. In [7], servers in cloud networks can interact with each other by exchanging their resource autonomously based on the tit-for-tat strategy. The tit-for-tat strategy is used as a data exchange rule and is actually deployed in BitTorrent systems [9]. While the resource exchange strategy based on tit-for-tat can significantly improve the overall download rates, its focus is rather on the efficient delivery of delay-insensitive data, but not the delivery of delay-constrained data. As a consequence, only limited performance is guaranteed for delay-constrained data.

Moreover, there has been efforts to minimize the energy consumption for cloud systems and cloud data centers (e.g., [10]–[12]). In [10], a packet-level simulator is designed to capture energy consumption from several components such as servers, switches, links, etc. Power consumption can be

This research was in part supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2010-0009717) and in part by the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2013-H0301-13-1002, NIPA-2013-H0301-13-1008) supervised by the NIPA (National IT Industry Promotion Agency).

978-1-4799-2851-4/13/$31.00 ©2013IEEE

minimized based on virtualized backbone topology [11] and QoS requirements are additionally considered [12]. Since these works mainly focus on minimizing energy consumption in cloud systems, performance in service quality may be limited to real-time applications.

In this paper, we propose a utility-based resource management strategy for servers in cloud networks that can address the limitations discussed earlier. Unlike prior utility-based approach in [13], where only QoS and operating costs are considered, we design a utility function that can capture serviced data priority, contribution of other servers, and the level of CPU utilization. More specifically, if higher priority data packets that have stringent delay deadlines are serviced, or if servers can operate in a desired region of CPU utilization, or if more contributions of previous resource exchanges are recorded, higher utility can be achieved. Therefore, servers can make their decisions such that their utilities are maximized. Our simulations show that the proposed utility-based resource management strategy enables the system (i.e., servers in cloud networks) to support different levels of services based on the data priorities and to provide incentives to other servers' contributions while operating in a desired level of CPU utilization. Thus, delay-constrained data that has higher priority can be serviced faster while servers in the cloud networks can manage their loads efficiently, in particular, in networks with high congestions.

This paper is organized as follows. In Section II, several key features in cloud networks are discussed. In Section III, we propose algorithms that servers can deploy for requesting data packets or responding to the requests based on utility in cloud networks in Section III. The evaluation of the proposed approach is presented in Section IV and conclusions are drawn in Section V.

II. System Setup

In this paper, we consider an overlay cloud network that consists of multiple interconnected servers, where users are connected to the servers for services. We denote a server i as S_i and a set of servers as $\mathbf{S}$, i.e., $S_i \in \mathbf{S}$. Servers in a network can exchange data by *requesting* and *responding* to the requests. In this section, we present the three features included in the proposed utility function, i.e., data priority, server contributions, and CPU utilization.

A. Prioritized Data

We assume that input data can have various classes according priorities (e.g., real-time multimedia has higher priority than general file data). More specifically, there are N_d different data types and data n can be divided into K_n packets having a fixed size. The kth packet in data n requested by server j, S_j, is denoted by $d_{S_j}(n,k)$. The priority of kth packet of data n in server S_j is denoted by $\rho(d_{S_j}(n,k))$ and each of priority can be determined by its characteristics. For example, multimedia data is in general delay-constrained compared to general file data, meaning that multimedia data packets have higher priorities. The priority is specified by α_p, i.e.,

$$\rho(d_{S_j}(n,k)) \in \{\alpha_1, \ldots, \alpha_p, \ldots, \alpha_P\} \tag{1}$$

where $0 \leq \alpha_1 < \cdots < \alpha_p < \cdots < \alpha_P$. The total number of priorities is P which can be determined based on the data characteristics, network conditions, etc. A data packet with priority α_p has higher priority than a data packet with α'_p if $\alpha_p > \alpha'_p$.

B. Contribution of Servers

In the proposed approach, the contribution of associated servers (i.e., servers that responded to requests from a server) is explicitly considered. In order for efficient data exchange among servers, it is important for individual servers to retrieve necessary data packets from the other servers. Thus, the cooperation among servers is essential, and in particular, we adopt tit-for-tat strategy from game theory [14], [15] in order to promote their cooperation in this paper. It has been shown that tit-for-tat strategy is very efficient for data exchange and has been widely used in practice (e.g., in P2P networks [9]). Since tit-for-tat strategy exploits contributions of agents in the system (in our case, servers), we define a measure of contribution of server S_j to server S_i at time (decision cycle) t as

$$c_{ji}(t-1) \in \{\alpha_1, \ldots, \alpha_p, \ldots, \alpha_P\}. \tag{2}$$

This means that the contribution of S_j to S_i is basically determined by the data priority serviced by S_j for S_i. More specifically, the contribution of server S_j to server S_i is determined by 1) whether server S_j is responded to the previous request from server S_i and by 2) the data priority that server S_j provided. If server S_j does not respond to a request from server S_i, $c_{ji}(t-1) = 0$. Hence, it is reasonable that a server with higher contribution can be considered as more cooperative. Consequently, individual servers can efficiently exchange necessary data packets by requesting necessary data packets from servers that have higher contribution, which is discussed in detail in Section III.

C. CPU Utilization of Servers

For each server, responding to the requests from other servers becomes loads for its CPU. In this paper, the CPU loads are represented by CPU utilization and it is assumed that the CPU utilization proportionally increases as the amount of CPU loads increases, i.e., the number of responses increases. Unlike general approaches of resource optimization, where available resources should be allocated entirely, CPU utilization needs to be limited below a certain level. For example, in [16], if the CPU utilization is 50%, 70% and 90%, then it takes 2 seconds, 3 seconds and more than 10 seconds for CPU to complete a given task, respectively. While lower levels of CPU utilization can guarantee a short delay of service time, it may not be energy efficient. Energy efficiency in general increases as CPU utilization becomes high – the energy efficiency of a server can reach 90% when the CPU load is more than 50% [17]. Therefore, servers need to consider these

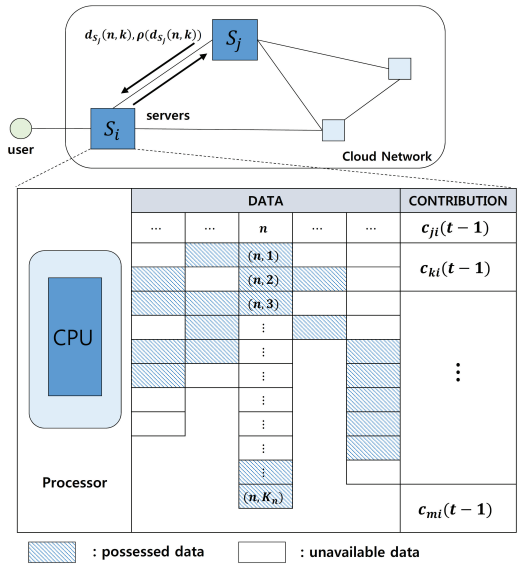


Fig. 1: Proposed cloud networks

tradeoffs when they respond to requests from the other servers. More specifically, it is important that servers need to maintain their CPU utilization levels in a desired operating region. In this paper, CPU utilization is measured by $\eta_{L_i}(t)$ based on CPU load L_i of server S_i at time (decision cycle) t. The CPU loads are divided into M sub-regions that are specified by thresholds $\theta_m(\%)$, expressed as

$$\eta_{L_i}(t) = \begin{cases} \beta_1, & \text{if } \theta_1 \le L_i < \theta_2 \\ \beta_2, & \text{if } \theta_2 \le L_i < \theta_3 \\ \vdots & \vdots \\ \beta_M, & \text{if } \theta_M \le L_i < \theta_{M+1} \end{cases} \quad (3)$$

The optimal region of CPU utilization level can be determined based on target applications, tasks, etc. The system setup is depicted in Fig. 1.

III. Utility-based Resource Management Strategy in Cloud Networks

In this section, we propose a utility-base resource management strategy in cloud networks. We define a utility function that can capture the key features discussed in Section II and then propose algorithms that servers can deploy for requesting data packets and responding to requests.

A. Utility Function of Servers

As mentioned, servers in cloud networks can have either the entire data packets or a part of data packets. Therefore, for a user's data requests, 1) if servers have the entire data, they can directly provide the data that they have, or 2) if they have only a part of data, they provide the part of data while forwarding the other part of data by retrieving them from the other servers. Thus, it is essential for servers to efficiently exchange data packets. Moreover, it is also important for servers to process data packets having higher priorities such that QoS requirements can be satisfied.

These are captured by a *utility* in the proposed approach, where each server makes decisions how to request necessary data packets or how to respond to data packet requests based on its utility. The utility function of a server includes data priority, contributions of servers and CPU utilization. More specifically, the utility of server S_i, denoted by $U_{S_i}(t)$, is expressed as a weighted sum of data priority, the measure of contribution and the measure of CPU utilization, i.e.,

$$U_{S_i}(t) = \sum_{j=1}^{N_{SS_i}(t)} \{\omega_\rho \cdot \rho(d_{S_j}(n,k)) + \omega_c \cdot c_{ji}(t-1) + \omega_\eta \cdot \eta_{L_i}(t)\} \quad (4)$$

where $N_{SS_i}(t)$ denotes the number of data that server S_i responds to requests from other servers. The utility function in (4) consists of three terms that represent data priority $\rho(d_{S_j}(n,k))$, contributions of servers $c_{ji}(t-1)$ and CPU utilization $\eta_{L_i}(t)$. The corresponding weights are respectively denoted by ω_ρ, ω_c and ω_η which can be appropriately determined based on target services of cloud networks. Then, each server makes its decisions such that its utility can be maximized. Therefore, data priority, contributions and CPU utilization included in the utility function can be indirectly controlled.

The ultimate goal of deploying the utility function is to prevent servers from overload and enable servers to operate strategically, leading to high quality services to end users in cloud networks.

B. Algorithms of Data Exchanges

Based on the tasks of servers, servers need to *respond* to the requests from other servers or need to *request* necessary data packets to other servers. In Section III-A, since utility function is defined such that maximizing the utility can lead to 1) delivering data packets having higher priority faster, 2) providing incentives for contributions and 3) maintaining CPU utilization levels in a desired operating region, we design two algorithms for responding to the requests and requesting necessary data packets that can be deployed in servers.

1) Algorithm for Responses: When responding to data packet requests from other servers, server S_i selects requests of higher priority data packets from servers with higher contributions. Then, the requests can be eventually accepted by the server if its CPU utilization is in a desired operating range. This process repeats until accepting additional data packet requests results in utility decrease. Hence, server S_i selects server S_j at time (decision cycle) t that can improve its utility the most, i.e.,

$$S_j^* = \arg\max_{S_j \in \mathbf{S}} U_{S_i}(t) \quad (5)$$

where utility function is defined in (4). The set of servers selected by server S_i is denoted by $\mathbf{S}_i^*$. The block diagram for the algorithm is shown in Fig. 2.

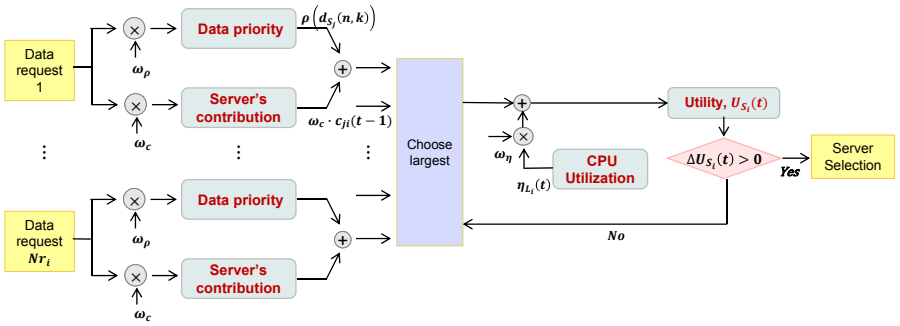


Fig. 2: Block diagram for proposed algorithm of response

2) Algorithm of Data Request: Motivated by tit-for-tat strategy, the algorithm of data request aims to maximize the probability that a server S_i requesting necessary data packets to other servers is serviced. Since other servers deploy the algorithm of response discussed in Section III-B, server S_i can expect that the other servers that were serviced by server S_i respond to its request with higher probability. This is because $c_{ij}(t-1)$ is contributed to the utility of server S_j. Moreover, if the data priority that server S_i processed is higher, its contribution to the utility of S_j is also larger. Thus, server S_i requests data packets from servers S_j^* such that

$$S_j^* = \arg\max_{S_j \in \mathbf{S}} c_{ij}(t-1). \tag{6}$$

If multiple requests are allowed, several servers can be selected and multiple requests can be sent.

IV. Simulation Results

In this section, we quantitatively evaluate the proposed resource management strategy based on our simulation results and show that it outperforms conventional algorithms such as random-based and tit-for-tat-based approaches. Servers deploying random-based approach randomly choose the other servers when they request data packets or respond to incoming requests. Alternatively, tit-for-tat-based approach only considers the previous action of associated severs, i.e., the contribution of the associated servers. In order to highlight the proposed utility function, we focus on the impact of the three features on the system performance.

A. Simulation Setup

In our simulations, there are 100 servers in the network, i.e., $|\mathbf{S}| = 100$. Every request and respond from each server is performed in each decision cycle. For illustration, we assume that there are 100 different types of data ($N_d = 100$) and 10 different priorities ($P = 10$). Moreover, we assume that data priorities are set by $\alpha_p = p$ for $p = 1, \ldots, 10$, which is an illustrative realization of the condition in (1). 40% of the entire data N_d is initially distributed to servers. In order for more realistic scenario, we set up a request probability in each server, meaning that each server can either request or not in every decision cycle with a predetermined probability. The request probability is set by 40% (i.e., $P_{req} = 40$) in the following illustrative simulations. For CPU utilization levels, we assume that there are five levels, which are specified as

$$\eta_{L_i}(t) = \begin{cases} \beta_1 = 0.5\alpha_{max}, & \text{if } 0\% \le L_i < 50\% \\ \beta_2 = \alpha_{max}, & \text{if } 50\% \le L_i < 70\% \\ \beta_3 = -0.5\alpha_{max}, & \text{if } 70\% \le L_i < 80\% \\ \beta_4 = -\alpha_{max}, & \text{if } 80\% \le L_i < 90\% \\ \beta_5 = -1.5\alpha_{max}, & \text{if } 90\% \le L_i < 100\% \end{cases} \tag{7}$$

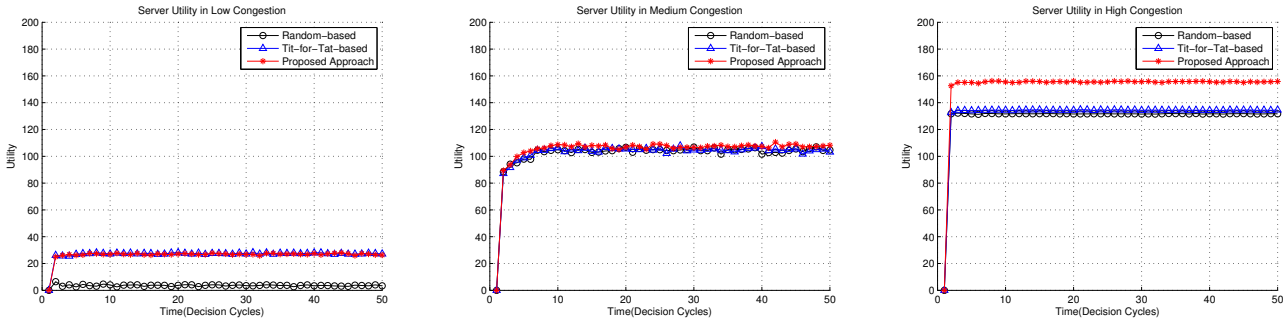
These illustrative levels and thresholds can be adaptively redesigned in different scenarios or network conditions. In the operating regions given in (7), the measure of CPU utilization is the largest, β_2, which means that the best range of CPU utilization in this example is determined between 50%∼70% of CPU loads. If CPU loads are below 50%, CPU utilization needs to be boosted up, so that the corresponding measure β_1 is determined such that $0 < \beta_1 < \beta_2$. If CPU loads are above 70%, however, the response time to provide services may become significantly slow. Thus, in order to prevent CPU from operating in this region, the measure of CPU utilization is set as negative values.

The degrees of network congestions in the simulations are emulated by changing the number of simultaneous data packet requests from each server. In particular, the number of simultaneous data requests is 10, 30, and 80 for low, medium, and high congestions, respectively. The experiments are independently performed 100 times and the average data are used for evaluation of performance.

B. Server Utility

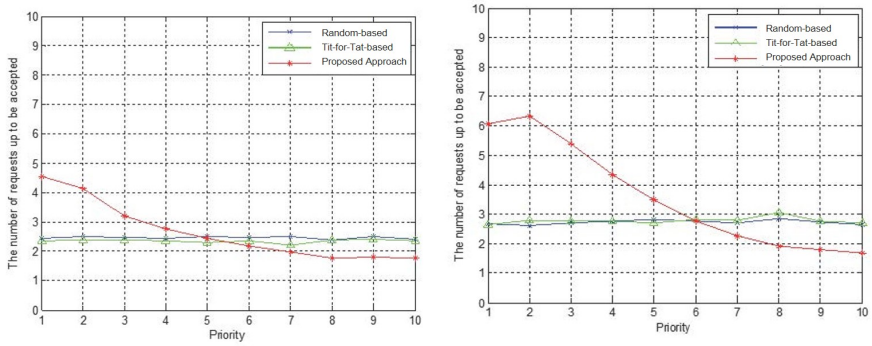
We first evaluate the performance of the proposed algorithm in the perspective of the utility. In our simulation, we consider that the impact of data priority, server contribution and CPU utilization on the utility is the same, meaning that $\omega_\rho = \omega_c = \omega_\eta = 1$. However, the weights can be appropriately adjusted.

Fig. 3 shows the utilities achieved by the three algorithms for different conditions of network congestion. It is observed that higher utility can be achieved as network conditions become congested for all strategies. This is because the utility defined in (4) increases as more data packets are processed. More importantly, it is observed that smarter strategies (i.e., tit-for-tat and proposed approach) can play an important role



(a) Low Congestion (b) Medium Congestion (c) High Congestion

Fig. 3: Server utilities in different network congestions.



(a) Medium Congestion (b) High Congestion

Fig. 4: The number of data packet requests until they are finally accepted over different priorities and network conditions.

in cases where resources are scarce. For example, if there are smaller number of data requests (i.e., low congestion), cooperation between servers become important. Thus, tit-for-tat or the proposed approach that explicitly consider the contributions of servers can achieve higher utility. Alternatively, if there are excessive number of data requests (i.e., high congestion), servers that selectively process data packets having higher impact on server utility (i.e., higher priority data packets) can achieve higher utility. Thus, the proposed approach can achieve the highest utility.

C. Prioritized Data Processing

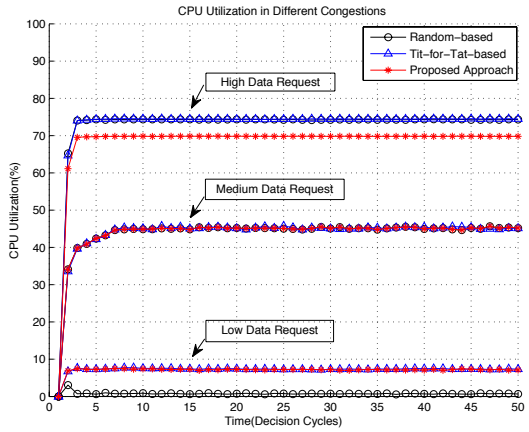
In order to confirm that data packets with higher priority are processed (i.e., delivered or transmitted) faster than those with lower priority, the number of requests until they are finally accepted and processed is measured. As discussed in Section III, each server accepts or responds to the requests if it improves its own utility. Thus, if a data packet request from a server S_i does not improve the utility of server S_j, the request is rejected by S_j so that server S_i needs to find other servers that can accept its request. Therefore, if the number of data packet requests until they are finally responded is smaller, it can be considered that it takes less time to be serviced.

Fig. 4 shows the number of data packet requests until they are responded based on data priority in different network conditions. It is obvious that it takes longer and shorter time to process data packets with lower and higher priorities, respectively, if the proposed approach is deployed. This is because the proposed approach explicitly considers the data priorities. However, the processing time is almost constant when random-based or tit-for-tat-based approaches are deployed, as they do not consider the data priority. It can also be observed that overall processing time increases as the network condition becomes congested.

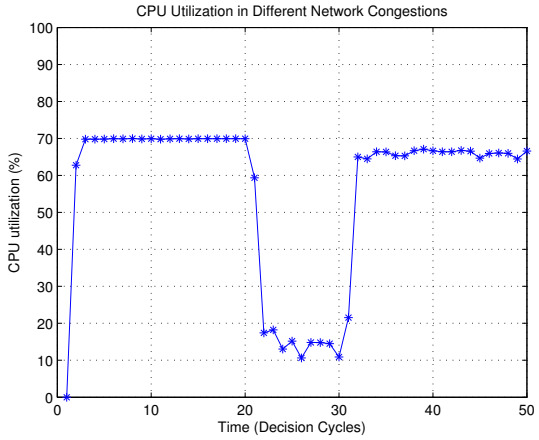
D. CPU Utilization

We finally evaluate the proposed approach in terms of the CPU utilization. The CPU utilizations achieved in different network conditions are presented in Fig. 5.

In Fig. 5(a), it is shown that overall CPU utilization levels increase as network conditions become congested. This is because there are more loads that CPU need to process as more data packets are requested in the network. While overall CPU utilization levels increase, only the proposed approach can prevent CPU loads from being over 70%, which is clearly shown in Fig. 5(a). This is because of the utility function is defined such that the measure of CPU utilization is negative in the range of 70%∼100% for CPU loads and is the highest in the range of 50%∼70% for CPU loads. Similar to the results discussed in Section IV-B, it is again observed that



(a) CPU Utilization in Different Congestions (Data Requests)



(b) Various Network Congestions

Fig. 5: Overall average CPU utilization in different network conditions.

smarter strategies (i.e., tit-for-tat and proposed approach) can play an important role in the cases where resources are scarce. For more realistic scenario, we adopt a time-varying request probability of servers on each decision cycle, which is shown in Fig. 5(b). The request probabilities are set by 70%, 20% and 40% for the decision cycles in $0 \leq t \leq 20$, $20 \leq t \leq 30$, and $30 \leq t \leq 50$, in order to emulate high (congested), low and medium network conditions, respectively. In summary, our experiment results confirm that servers that adopt the proposed approach can achieve the highest server utility. They can correspondingly provide better services by processing data with higher priorities faster and can maintain in a desired operating range.

V. Conclusion

In this paper, we consider cloud networks where multiple servers are trying to provide services to end users. In order to guarantee QoS requirements, we propose a strategy how servers exchange data packets based on several constraints induced from service requests (e.g., service types, data priorities, delay constraints) and servers' resource usages (e.g., current loads of servers, contributions of servers). We define a utility that can capture the performance of servers and the servers can make their decisions such that their utilities are maximized. Our experiment results show that the servers that deploy the proposed approach can achieve the highest server utility. They can correspondingly provide better services by processing data with higher priorities faster and can maintain in a desired operating range.

References

[1] H. Park and M. van der Schaar, "Bargaining strategies for networked multimedia resource management," *IEEE Trans. Signal Processing*, vol. 55, no. 7, pp. 3496–3511, Jul. 2007.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proc.ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, 2003, pp. 29–43.

[3] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website*, vol. 11, p. 21, 2007.

[4] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey." *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[5] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 5, pp. 44–48, 2012.

[6] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *Proc. IEEE International Conference on Computer Communications(INFOCOM)*, 2012, pp. 963–971.

[7] O. Babaoglu, M. Marzolla, and M. Tamburini, "Design and implementation of a P2P cloud system," in *Proc. ACM Symposium on Applied Computing*, 2012, pp. 412–417.

[8] Z. Chen, Y. Zhao, X. Miao, Y. Chen, and Q. Wang, "Rapid provisioning of cloud infrastructure leveraging peer-to-peer networks," in *Proc. IEEE International Conference on Distributed Computing Systems Workshops*, 2009, pp. 324–329.

[9] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. P2P Economics Workshop*, 2003.

[10] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," in *Proc. IEEE Global Telecommunications Conference(GLOBECOM)*, 2010, pp. 1–5.

[11] B. Kantarci, L. Foschini, A. Corradi, and H. T. Mouftah, "Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems," in *Proc. IEEE Global Telecommunications Conference Workshops(GLOBECOM workshops)*, 2012, pp. 708–713.

[12] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 826–831.

[13] D. Minarolli and B. Freisleben, "Utility-based resource allocation for virtual machines in cloud computing," in *Proc. IEEE Symposium on Computers and Communications*, 2011, pp. 410–417.

[14] K. Binmore, *Fun and Games: A Text on Game Theory*. Lexington, MA: D.C. Health, 1992.

[15] D. Fudenberg and J. Tirole, *Game Theory*. Cambridge, MA: MIT Press, 1991.

[16] IBM. Informix 11.50 information. [Online]. Available: http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp

[17] "Cisco energy efficient data center solutions and best practices," White Paper.