MANAGEMENT OF CONTEXT-AWARE SOFTWARE RESOURCES DEPLOYED IN A CLOUD ENVIRONMENT FOR IMPROVING QUALITY OF MOBILE CLOUD SERVICES

Sohame Mohammadi[1], Kamran Zamanifar[2] and Sayed Mehran Sharafi[3]

Department of Computer Engineering, Najaf Abad Branch, Islamic Azad University, Isfahan, Iran

ABSTRACT

In cloud computing environments, context information is continuously created by context providers and consumed by the applications on mobile devices. An important characteristic of cloud-based context aware services is meeting the service level agreements (SLAs) to deliver a certain quality of service (Qos), such as guarantees on response time or price. The response time to a request of context-aware software is affected by loading extensive context data from multiple resources on the chosen server. Therefore, the speed of such software would be decreased during execution time. Hence, proper scheduling of such services is indispensable because the customers are faced with time constraints. In this research, a new scheduling algorithm for context aware services is proposed which is based on classifying similar context consumers and dynamically scoring the requests to improve the performance of the server hosting highly-requested context-aware software while reducing costs of cloud provider. The approach is evaluated via simulation and comparison with gi-FIFO scheduling algorithm. Experimental results demonstrate the efficiency of the proposed approach.

KEYWORDS

Cloud Computing, Context-Aware Computing, Context Management, SLA.

1. INTRODUCTION

Cloud computing is a distributed and parallel computing system that builds on the convergence and advancement of several technologies, especially in utility and grid computing, autonomic computing, hardware virtualization, service-oriented architecture, web services and the existing relationship among them [1]. According to the NIST's definition [2], Cloud Computing consisting of five essential features namely on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. All services and resources in this technology are provided as different services while cloud computing architecture is classified based on a three-layer structure comprised of Infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [1, 2].

Due to the improvement of mobile applications and emergence of the concept of cloud computing in the last decade, cloud computing was introduced as a potential technology for mobile services including context-aware services. Although most computing devices (e.g. mobile phones, PDAs, laptops, tablets, etc.) in the cloud computing environment are in small sizes and can accompany the users in all places, they face with resource constraints (e.g. battery life, cpu speed, storage,

DOI:10.5121/ijdps.2014.5501

and bandwidth). Therefore, resource constraints prevent quality of mobile services. Hence, the data processing and storage would be carried out outside the mobile device in the cloud while mobile device operates as a displaying platform [3].

A context-aware service continually collects the context of the environment and adapts its operation to the context data. When a request is sent by a customer to the context-aware software in the cloud computing distributed environment, the response to the request of context-aware software is affected by loading extensive context data from multiple resources on the chosen server [4, 5]. Therefore the performance of such software would be decreased during execution. Moreover, in the cloud computing environment, the users of cloud services are grouped into various classes according to the costs they have paid. The services are managed properly provided that those of higher qualities are dedicated to the users who have paid more. Meeting service level agreement and decreasing managing costs of resource are important Challenges for Cloud providers. If cloud providers either fail to meet the objectives set in the agreements or provide services of lower qualities to customers than what agreed upon, they would be penalized by claiming them the costs or through recharging their accounts [1, 5], which results in reduction of the total profit earned by the cloud providers.

In this study, a new scheduling algorithm is suggested relying on classifying similar context queries and dynamic scoring of requests to enhance the efficiency of servers hosting highly requested context-aware software. The proposed approach is based on the idea that communications costs during execution of an application are regarded as overhead, if especially communications are established for loading context information from far resources. The costs of communications could be compensated by making use of multithread applications through running proper requests with regard to the context data type which results in the improvement of the efficiency of provided services. Also what might happen is that many users are looking for similar context information. In the other words, lots of requests in queue are related to getting access and processing other similar context information. Therefore, conscious scheduling considerably reduces the indicator of mean response time and the costs of cloud providers.

The rest of the paper is organized as follows. The related works are discussed in the next section. Sections 3 and 4 are dedicated to the background concepts and Problem formulation. The proposed algorithm is represented in section 5. Section 6 describes the experimental results and the overall evaluations. Finally, the conclusion remarks and states future direction of this research are identified in Section 7.

2. RELATED WORK

In recent years, many job scheduling methods have been proposed in the grid and other distributed environments. In [7], a comprehensive review of the scheduling algorithms in the grid environment and their comparisons are presented. In contrast, due to the unique features of the cloud computing environment, its scheduling problems are different. On the other hand, a new generation of applications known as context-aware applications has emerged and cloud infrastructure has been introduced to be a suitable technology for context-aware applications. Context-aware job scheduling, management and obtaining required context data are the main challenges for the context-aware services. A notable number of context provisioning and management approaches have been proposed. Surveys of which have been published in order to understand the features of existing systems for instance in [8-11]. In the following, a brief review on some of the previous studies related to the present study is discussed.

Boloor et al have investigated a research regarding the scheduling and dynamic allocating of requests to manage the context-aware software in the cloud distributed environment [4-6]. In this

study, the cloud distributed environment is modeled as several data centers. An attempt is made to reduce the overall penalties charged to the cloud through proposing Data-aware Session-grained Allocation with gi-FIFO Scheduling (DSAgS), a novel decentralized request management scheme deployed in each of the geographically distributed datacenters, context cache replacement policies to be applied at the time of loading new context data to replace the prior contexts when the cache is full and considering the service level agreements aiming at saving time[4-6].

S. L. Kiani et al in the article [12] have proposed a new mechanism of cache management in the cloud environment to manage the context-aware software. In this mechanism, according to different types of context information, validity period and access patterns, the cache is divided into two sections in a scalable way and the efficiency of the whole system is improved by employing appropriate scheduling policies for each section. Although the proposed idea could be considered an effective solution to improve the system, but the centered design and usage of a cache in the context broker, which is main coordinating component in this system, leading to the creation of bottleneck. Therefore, the architecture lacks scalability and location transparency.
Assuncao et al in the article [13] have resorted to the cloud computing infrastructure to overcome the resource constraints in the mobile environment. Combining the context awareness and adaptive job scheduling technologies, the study has proposed a context-aware scheduler. The main objective of this model is resources utilization in the cloud computing environments to improve the quality of services. Y. Zhu et al in [14] have proposed a framework for the logical stream that organizes services in the cloud computing environment in a way that makes them capable of adapting to the environment's contextual changes and using information and resources of the cloud computing during execution. The model has been implemented on Java platform. Making use of context information to adapt services to the environment's changes, sharing resources and improvement of the utilization of the cloud resources, portability, flexibility and interoperability of the services could be considered its principal features.

Badidi & Esmahi in [15] did their attempts to optimize scalability, interoperability and quality of services. Therefore, a multi-attributes decision algorithm and a framework for preparation of the context information were proposed in a way that they were based on the implementation of the context-aware services on the cloud side, utilization of the context broker as a mediator between context consumers and context services and employment of publish/ subscribe model. The context providers in the publish/subscribe model expose their data items to the context-aware applications to be read. If an application wants to subscribe to a context, it sends its subscription request to the broker. The subscriptions have to be adapted to the data items when the context information is changed. Therefore, the broker decides to send an alarm to the applications subscribed to this context. Then the applications could take delivery of the published context values [16]. Although the proposed idea in this study is to attract attentions, but no discussion is made regarding the ways brokers could be used as mediators and it has merely remained as an idea.

3. BACKGROUND

Some of the concepts related to the present study are defined in this section.

3.1. Cloud-based context services

Dey and Abowd [17] define context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves". Context includes two categories [17]: primary contexts are location, time, activity and identity, which are considered as the most important context information. Secondary contexts involving

any other information that can be retrieved from the primary contexts such as weather, email, etc. different types of contextual information have varying temporal validity durations and remaining valid for a particular duration.

Context-aware services continuously adapt their state behavior with the context changes. Context services are located on the clouds side, and context consumers (CxCs) look up and invoke the services [18]. In cloud environments, CxCs are context data regarded as inputs to adapt their behavior to the users' current situation (e.g. context aware services). Context provider (CxP) is an entity which provides context information. An CxP gathers raw data from context sources(e.g., sensors, networks, mobile phones, web services, etc), and convert them to meaningful information, aggregation, modeling, reasoning and uploaded to the context broker(CxB).The CxB acts as an intermediary to control context flow between CxPs and CxCs. Its main function is to query resolution, event management, registration of available CxPs, provide routing and lookup services [19, 20].

There are two cases of utilizing context information in cloud environment [18]: The first context service is used to guide and provide improved service to the user. The second, context is used to find fault and maintain the system stable and adapt their operations to change environmental conditions.

3.2. Service level agreement

Cloud computing is a parallel and distributed computing systems that enable the deployment and execution of applications and services on remote data centers with a pay-per-use business model based on SLA over the internet. The SLA established through negotiation between the cloud provider and the cloud consumer to ensure service quality. Negotiation strategy between the cloud provider and consumer can investigate through the third party called broker. The SLA may involve a variety of metrics to measure the quality of service requirements such as availability, execution time, price, reliability, cpu usage and the storage used by the consumer[1,21].

One of the most applicable types of SLA is based on percentage which service Provider tries to allocate resources to the customers in accordance with the agreed percentile during certain periods of time. While in case of any failure regarding meeting the agreed level, they have to pay fine [1].

The probabilistic SLA has the following parameters [6, 22]: Response time(R), which is duration of the time it takes from when a request arrives at the cloud and leaves it. Response time threshold (R_T), the value of R_T is the deadline of a request which must leave the cloud. Actual conforming percentile: the percentile of requests of a particular user class which have met the required response time. This value is controlled by the request management policy chosen by the cloud provider. Desired conforming percentile: For different classes of users, this value is determined through the negotiation between the cloud provider and the consumer. A penalty (P), clause can be applied to the cloud provider, if the quality of service requirements is less than a given value.

There are various penalty functions to calculate the cloud service providers' penalties including linear, exponential and stepwise functions. The main focus in this study takes stepwise penalty function into accounts while bases requests on deadline to meet the satisfaction of the consumers.

4. STATEMENT OF THE PROBLEM

In recent years, increased use of mobile applications (e.g. context aware applications) and emerging of cloud computing concept, cloud computing infrastructure has been introduced as a potential technology for mobile-based environments in order to cope with resource constraints for

mobile users[3,13]. There are numerous servers in a distributed cloud environments, each one hosting one or more context aware applications to provide services for multiple classes of CxCs. When a request for a context aware service arrives at the datacenter through the internet, the scheduling agent submits it to a suitable server. The job has to be scheduled by internal scheduler at the end-server to execute it on resource and to send the answer back to the scheduling agent.

Context aware services utilize context to adapt themselves to their changing environment. The response time of a request for a context aware application involves context provisioning, context provider lookup times and time required to load context information on the chosen end server [4,5,12]. Therefore, the speed of context-aware software would be reduced during execution times in the cloud computing environment. Moreover, users of cloud services are grouped into various classes according to the costs they have paid. Proper management of these services is met provided that services of higher qualities are dedicated to those who have paid more [1, 5].

To solve the above mentioned problem, a study was carried out by Boloor et al regarding scheduling and dynamic allocating of requests to manage context-aware software in the distributed cloud environment [4- 6]. In this study, they propose Data-aware Session-grained Allocation with gi-FIFO Scheduling (DSAgS) in a distributed cloud that is modeled as several datacenters. The proposed scheduling scheme is based on the gi-FIFO scheduling algorithm which has been mathematically proven in [23] to be the most appropriate for probabilistic SLAs for a single server serving multiple class jobs. The scheduling policy they use to schedule requests queued at each cloud server for the cloud provider hosting the context aware applications and it could be described as follows: Firstly, choose the request class with the highest penalty; then, amongst all the queued requests of the chosen class, choose one with maximum waiting time resulting in a response time less than or equal to R_T. If there is not such request, it will be chosen the request with higher waiting period, consequently resulting in a response time greater R_T [4- 6].

Despite the fact that the proposed method in the present study could be considered a proper solution regarding managing requests of the server hosting context-aware software in the cloud environment but given the fact that the cloud servers are capable of responding to limited number of requests at a specific moment and they should process and respond to high volumes of requests based deadlines at that specific moment, so the number of customers losing deadline at that operation would be increased while the efficiency of such highly-requested servers would be drastically decreased and consequently cloud providers penalties could be increased. Therefore, the present research aims at proposing a proper scheduling algorithm for the requests of context-aware software in cloud computing environment with regard to the represented challenges.

5. THE PROPOSED CONTEXT-AWARE ALGORITHM

The decision making process of the scheduling algorithm to choose the next request to be executed among requests queued at the end server is affected by classes' penalties, deadline of requests, context data storage type and differing patterns of updating context information in dynamic and evolutionary environments. According to this problem, the context-aware scheduling algorithm is engaged in multiple dimensions and complications related to finding a way to estimate indefinite scores in a multidimensional space. Hence, a new approach of scheduling is proposed for management of context-aware software requests.

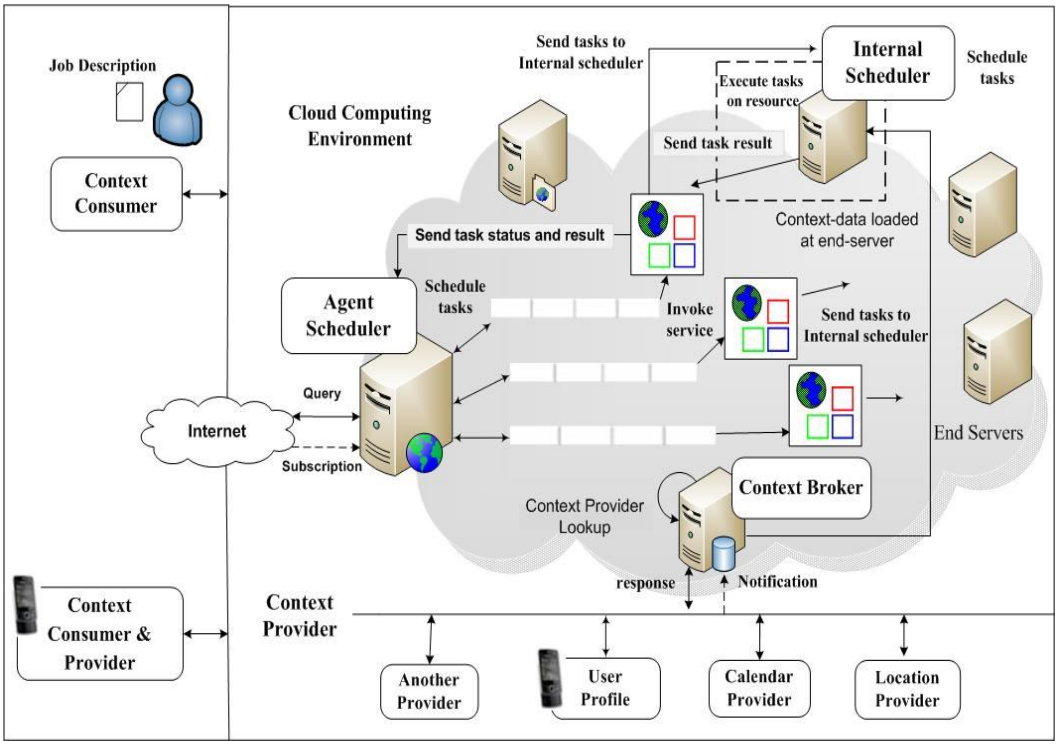

Figure 1. Overall Architecture of the cloud based context provisioning system

The internal architecture of the context provisioning system containing some context providers on a cloud infrastructure is depicted in figure 1. In a cloud computing environment, the context customer including all computing devices of customers (e.g. mobile phone, PDAs, laptops and other computational devices of cloud environment) send their requests for a particular context type through the Internet[5, 6]. Users' requests are allocated to the cloud end servers based on context-aware and dynamic scoring [5, 6]. Cloud servers might host one or more various context providers, which each of them requiring for numbers of users' requests. There are numerous requests which call for similar context data types from such servers. The server hosting context-aware software has to respond to lots of requests at any moment. Under these circumstances, given the fact that the server is capable of responding to limited number of requests at a moment and providing required context data for requests takes time, therefore the efficiency of the server would be considerably reduced.

The problem is solved through proposing a solution which is based on classifying and dynamic scoring of requests with regard to their user classes and the required context scope. Scope is a set of closely related context parameters and context exchange unit that always requested, updated, provided and stored at the same time [20, 24]. In this approach, each group of requests requiring similar context types is considered a cluster in related scope while penalties for each cluster of scope will be determined with regard to the number of requests related to the needed context data and requests' user classes.

```
(1) K=5      //Number of user classes
(2) S=12     //Number of clusters & context-data sources (Scope)
(3) while Queue of requests is not empty do
(4)    for k = 1 to K do
(5)       Current-job= Apply gi-FIFO policy for the requests queued  //Output e of gi-FIFO
    scheduling is request to be scheduled
(6)       if scope of current-job is valid then
(7)          Request to be processed = current-job
(8)       else
(9)          while Localtime < Context load time do
(10)            for s = 1 to S do
(11)               scope_s = Select the scope with the shortest expiry time
(12)            end for
(13)            if Category_scope_s = 'long' then
(14)               scope_s = Select scope which has highest penalty value among scopes & is valid
(15)            end if
(16)            New-job= Apply gi-FIFO policy for jobs of scope_s
(17)            Request to be processed = New-job
(18)         end while
(19)         Request to be processed = current-job;
(20)   end for
(21) end while
```

Figure 2. The pseudo code of the proposed algorithm

Aiming at restricting the multidimensional space of request scoring, the gi-FIFO scheduling algorithm is run from the beginning of running the proposed algorithm to estimate classes' penalties and requests' response times. Nevertheless, given that loading context data takes time, if the required context information for running a request is not valid on a server at the time of choosing the next request to be operated, it is indispensable that the required context data be loaded on the server. During loading context data in order to compensate the communications costs by computations, the scope should be chosen with the shortest expiry time on the server. Regarding the updating pattern of the chosen scope, there are two main strategies behind the proposed scheduling policy namely select the soonest expiring first (SE) for scope with short update pattern and select the highest penalty (HP) score among the scopes that are valid on the server for the scope with the long update pattern. Therefore, if the conditions are met with regard to the loading time of context data, the requests related to the chosen scope would be implemented based on gi-FIFO scheduling. figure 2 shows the pseudo-codes of the proposed algorithm.

6. IMPLEMENTING AND EVALUATING

The proposed approach is simulated and evaluated by JAVA language programming. The conformance levels, average response time and cloud providers' penalties for executing requests are the parameters analyzed. As for simulation, the simulation parameters of articles [5, 12] are used. The input parameters for simulation are considered as follows: 1) 5 user classes, 2) 100 distinct sessions-id to assign the customer's class, 3) customers are entered the system through Poisson distribution with constant rate λ which is the number of customers at a time unit, 4) the service processes are fixed 5) twelve different scopes are used in this simulation and the parameters for each scope include: scope ID, the mean request processing time at context provider, validity period of produced context data and types of scope updating patterns. The

values of scopes using from table 1, 6) the time of loading context data is considered three times more than the average service rate for each request.

Table 1. Simulation Parameters [12]

CxP:ScopeID	Processing time[ms]	Validity[s]	Category
CxP:1	70	60	Short
CxP:2	70	60	Short
CxP:3	80	80	Short
CxP:4	80	80	Short
CxP:5	90	180	Short
CxP:6	90	240	Short
CxP:7	70	360	Long
CxP:8	70	400	Long
CxP:9	80	600	Long
CxP:10	80	900	Long
CxP:11	90	1200	Long
CxP:12	90	1200	Long

After determining the simulation values, the users' requests as the consumers of the context data would be produced and the gi-FIFO scheduling algorithm and the proposed approach would be implemented. Figure 3,4 show the plots of queues' conformance levels vs. the number of user requests. As the results show, regarding the gi-fifo scheduling policy, the growing rate of the conformance levels decreases as the number of user requests increases and after it roughly reaches zero. Thus experimental results reveal that the conformance levels obtained from the proposed algorithm are higher than gi-FIFO scheduling algorithm for all user levels. As the experimental results show, the proposed algorithm enjoys the best state at scheduling queue requests relevant to the user class with the highest penalty while the worst state is at scheduling those of the user class with the lowest penalty.

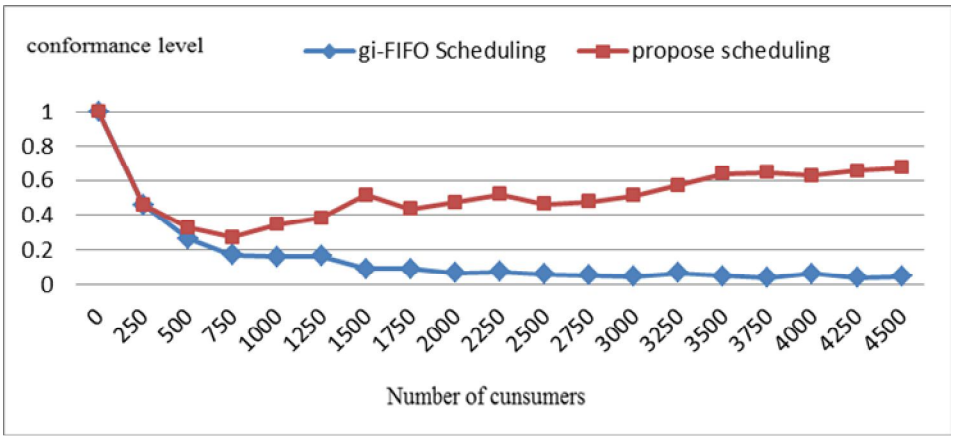

Figure 3.Comparison of conformance levels for the user class with the highest penalty

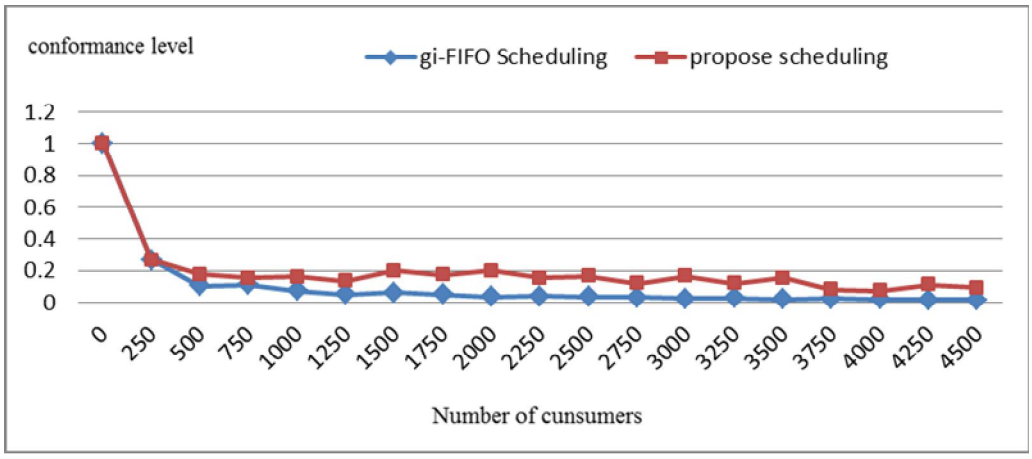

Figure 4. Comparison of conformance levels for the user class with the lowest penalty

Figure 5 shows the mean response time against the users' requests. Given that a server is capable of responding to a finite number of requests at a moment and it takes time to provide services with the required context data, so according to the gi-FIFO scheduling policy, the percentile of customers losing their deadlines at that operation would increase. Therefore, there would be a significant increase in the mean response time. Nevertheless, In the proposed approach given that requests are classified based on the required context information, they are scored dynamically and the loading time of the context data is overlapped with the implementation of the requests, the mean respond time is significantly reduced while increasing the total profit charged by the cloud providers.

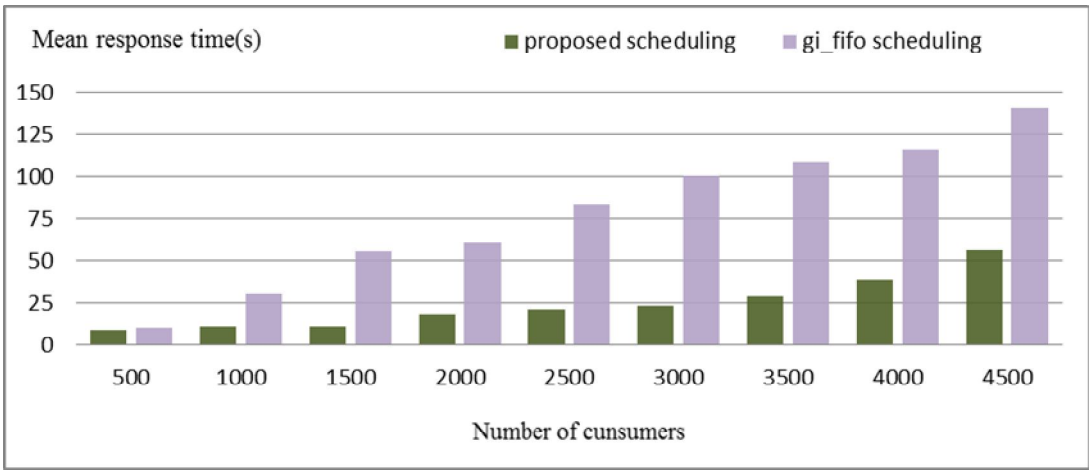

Figure 5. Comparison of average response time for proposed and gi-fifo schedules

For different classes of users are considered distinct penalty functions. The penalty functions for multi-class jobs define the way penalty could be calculated. Our method for calculating the penalties is based on the way presented by boloor [6] and is defined as follows[6]: If the effect of delaying the recently arrived request causes the non-conformance to increase beyond that given in the SLA, then the cloud provider is assigned a penalty of P$, else if the the non-conformance of request is low, no penalty would be added to the total penalty. The value of current non-conformance is calculated in below equation:

$$\text{non-conformance} = ((1 - cc_k) * X_k + 1)/(X_k + 1) \quad (1)$$

Where X_k is total number of requests serviced of class k from the start of the observation interval as measured by server d and cc_k is current conformance of class k in cloud as calculated by d [6].

Figure 6 shows the penalty of 4500 consumer queries incurred by the cloud provider with different context data update pattern. The proposed policy is evaluated with different management strategies regarding short validity (SV) scopes and long validity (LV) scopes in increment of 25% [12]. Typical result in the figure shows that gi-FIFO policy results in the maximum penalties charged. As for the approach with the shortest expiry time for scheduling as the duration of the context load time, the lowest penalty is assigned when the scope distribution in the context-data queries is biased towards SV scopes. The highest penalty value policy decreases the total penalty charged to the cloud across all context queries. The above-mentioned strategy, however, is better applied to LV context-data queries than SV context-data queries. The proposed scheduling makes use of two different policies that are suitable for both short and long validity scopes distributed context queries.

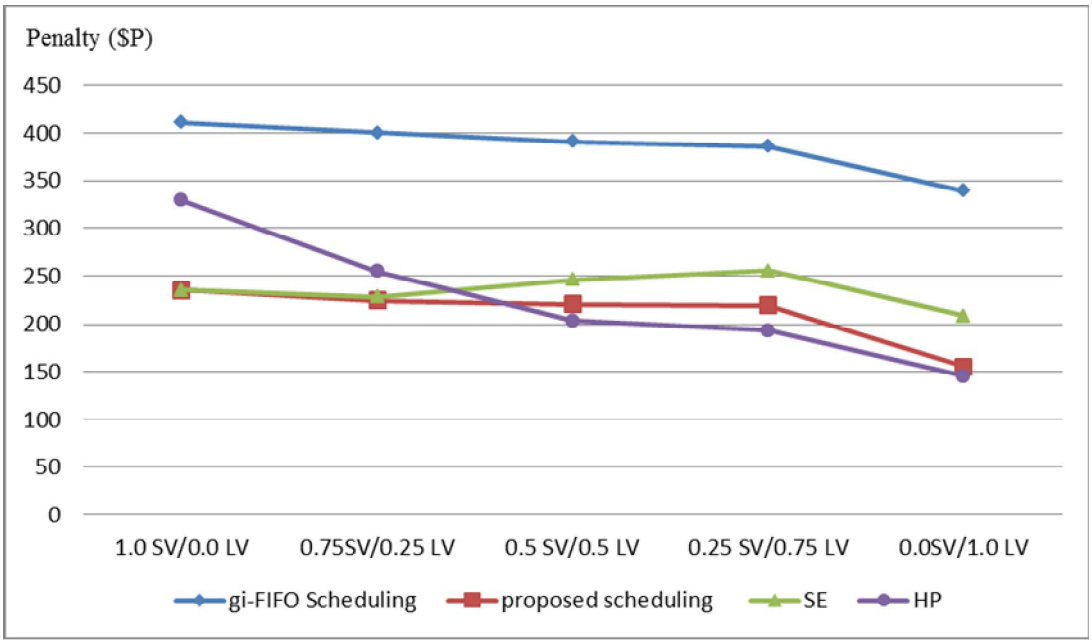

Figure 6: Comparison of total penalty incurred in gi-fifo and proposed schedules

7. CONCLUSIONS AND FUTURE WORK

In the cloud computing environment, since loading context data takes time, servers hosting context-aware software are faced with the problem of low speed of providing services. Therefore, the profit obtained decreases rapidly for cloud providers. In this research, a new scheduling algorithm for context aware services is proposed which is based on dynamic scoring of the requests to improve Qos. The proposed algorithm runs based on classifying requests in accordance with the type of the required context data, dynamic scoring and overlapping loading time of the context data with operating proper requests with regard to the storage type of the context data and their differing updating patterns. Experimental results demonstrate the efficiency of the proposed approach while highlight the necessity of managing the requests of the context-aware software in the cloud environment.

In the future, we will investigate on modeling the probabilistic SLA globally in a distributed cloud computing environment with numerous servers to stimulate developments in this infrastructure.

REFERENCES

[1] R.Buyya, J.Broberg, and A.M.Goscinski, "Cloud Computing Principles and Paradigms," Wiley Publishing, USA, 2011.

[2] P.Mell and T.Grance, "The NIST Definition of Cloud Computing", National Institute of Standards and Technology, Information Technology Laboratory, Technical Report Version 15, 2011.

[3] L.Guan, X.Ke, M.Song, and J.Song, "A Survey of Research on Mobile Cloud Computing," in Computer and Information Science (ICIS), 2011 IEEE/ACIS 10th International Conference on, pp. 387-392, 2011.

[4] K.Boloor, R.Chirkova, Y.Viniotis, and T.Salo, "Dynamic Request Allocation and Scheduling for Context Aware Applications Subject to a Percentile Response Time SLA in a Distributed Cloud," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pp. 464-472, 2010.

[5] K.Boloor, R.Chirkova, T.Salo, and Y.Viniotis, "Management of SOA-Based Context-Aware Applications Hosted in a Distributed Cloud Subject to Percentile Constraints," in Services Computing (SCC), 2011 IEEE International Conference on, pp. 88-95, 2011.

[6] K.Boloor, "Management of soa-based, data-intensive applications deployed in a distributed cloud subject to response time percentile service level agreements", Phd Thesis, North Carolina State University, Raleigh, NC, USA, 2012.

[7] D.Amalarethinam, P.Muthulakshmi, "An Overview of the Scheduling Policies and Algorithms in Grid Computing", International Journal of Research & Reviews in Computer Science, vol. 2, no. 2, pp. 280, 2011.

[8] M.Baldauf, S.Dustdar, and F. Rosenberg, "A survey on context-aware systems", International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, no. 4, pp. 263–277, 2007.

[9] H.Truong and S.Dustdar, "A survey on context-aware web service systems", International Journal of Web Information Systems, vol. 5, no. 1, pp. 5–31, 2009.

[10] J.Hong, E.Suh, and S.Kim, "Context-aware systems: A literature review and classification", Expert Systems with Applications, vol. 36, no. 4, pp. 8509–8522, 2009.

[11] M.Knappmeyer, S.L.Kiani, E.S.Reetz, N.Baker, and R.Tonjes, "Survey of Context Provisioning Middleware," Communications Surveys & Tutorials, IEEE, vol. 15, pp. 1492-1519, 2013.

[12] S.L.Kiani, A.Anjum, K.Munir, R.McClatchey, and N. Antonopoulos, "Context Caches in the Clouds," Journal of Cloud Computing: Advances Systems and Applications, 2012.

[13] M.D.Assuncao, M.A.S.Netto, F.Koch, and S. Bianchi, "Context-Aware Job Scheduling for Cloud Computing Environments," in Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on, pp. 255-262, 2012.

[14] Y.Zhu, R.Y.Shtykh, and Q.Jin, "A Human-Centric Framework for Context-Aware Flowable Services in Cloud Computing Environments," Journal of Information Sciences, 2012.

[15] E.Badidi, L.Esmahi, "A Cloud-Based Approach for Context Information Provisioning", Journal of World of Computer Science and Information Technology Journal (WCSIT, pp. 63-70), 2011.

[16] E.Badidi, "A Publish/Subscribe Model for QoS-Aware Service Provisioning and Selection", Journal of Computer Applications, vol. 26, 2011.

[17] G.Abowd, A.Dey, P. Brown, N. Davies, M.Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness", in Handheld and Ubiquitous Computing. vol. 1707, H.-W. Gellersen, Ed., ed: Springer Berlin Heidelberg, 1999, pp. 304-307.

[18] H.Jung, S.Dong, "A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services", Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010.

[19] H.Vahdat-Nejad, K.Zamanifar and N.Nematbakhsh, "Towards a Better Understanding of Context Aware Middleware: Survey and State of the Art", To be published, 2013.

[20] S.L. Kiani, A.Anjum, M. Knappmeyer, N.Bessis, and N. Antonopoulos, "Federated broker system for pervasive context provisioning", Journal of Systems and Software, vol. 86, pp. 1107-1123, 2013.

[21] R.Rajavel and T.Mala, "Achieving Service Level Agreement in Cloud Environment Using Job Prioritization in Hierarchical Scheduling", in Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam,

India, January 2012. vol. 132, S. Satapathy, P. S. Avadhani, and A. Abraham, Eds., ed: Springer Berlin Heidelberg, pp. 547-554, 2012.

[22] K.Boloor, R.Chirkova, T. Salo, and Y.Viniotis, "Analysis of Response Time Percentile Service Level Agreements in SOA-Based Applications", in Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, pp. 1-6, 2011.

[23] N.Agarwal and I.Viniotis, "Performance Space of a GI/G/1 Queueing System Under a Percentile Goal Criterion," in Performance Modelling and Evaluation of ATM Networks, D. Kouvatsos, Ed., ed: Springer US, pp. 474-484, 1995.

[24] M.Knappmeyer, S.L.Kiani, C.Fra, B.Moltchanov, and N.Baker, "ContextML: A light-weight context representation and context management schema", in Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on, 2010, pp. 367-372.