A Profit Oriented Session-based Admission Control Mechanism for the Cloud Environment

Abdel Rahman Tawakol, Hoda M. Hassan, Samah Senbel
College of Computing and Information Technology (CCIT)
Arab Academy for Science, Technology and Maritime Transport (AASTMT)
Cairo, Egypt
atawakol@gmail.com, huda.hassan@aast.edu, senbel@aast.edu

ABSTRACT
The proliferation of cloud-computing services has encouraged business-owners to migrate their applications to the cloud. With the existence of the Service Level Agreement (SLA) in the cloud, the business-owners can guarantee the quality of service they were promised to get from the providers. However, in such shared environment, there is a high probability of overload situations to occur violating the SLA, and causing profit losses to both providers and clients. This paper proposes an SLA-aware Profit-Oriented session-based Admission Control (POAC) mechanism that manages the overload problem in three-tier Web-based applications hosted on a cloud environment. The proposed Admission Control (AC) was validated using simulation. Our experiments show that the proposed approach makes profit-aware decisions, while taking the SLA in consideration. Thus benefiting both the service providers and the business-owners.

Categories and Subject Descriptors
C.5.5 [**Computer Systems Organization**]: Computer System Implementation — *Servers*; D.4.8 [**Operating Systems**]: *Performance — Measurements, Modeling and prediction*; K.5.m: [**Legal Aspects of Computing**]: Miscellaneous — *Contracts*.

General Terms
Algorithms, Design, Economics, Experimentation, Legal Aspects, Measurement, Performance.

Keywords
Cloud Computing, Admission Control, Quality of service, SLA, Profit oriented, Three-tier web application.

1. INTRODUCTION
In recent years, cloud computing has emerged as a new computing paradigm, where resources are exposed to customers as services [1], [2]. Cloud computing has several features and benefits that attract the business owners, like elasticity and the "pay-as-you-go" model, in which services are provisioned based on demand. Thus customers are charged based on their utilization. Consequently, business owners are no longer required to purchase expensive software and hardware up front, but rather they can start their business with a small budget, and increase their computing resources as their business grows [1], [2], [4].

In the cloud environment, the services are offered based on a Service level Agreement (SLA), which make the "pay-as-you-go" model practical. An SLA is a contract between the service provider and the customer. This contract includes the specifications related to the guarantees on the quality of the service provided, the cost of the service provided, and the penalties in case the stated guarantees are violated [5].

Recently, with the increase in the number of cloud providers, more business-owners are migrating their business to the cloud environment [3], [4]. However, business-owners need to guarantee the quality of the service that they have been promised in the SLA. Otherwise, they can penalize the service providers for any SLA violation. On the other hand, service providers aim to maximize their profit by serving more customers while honoring the SLA. This promotes a win-win situation for both service providers and business-owners. Accordingly, service-providers strive to guarantee the service level specified in the SLA by managing overload situations. However, in such shared environment it is common to have sudden bursts of workload that is followed by a period of normal load [6]. During peak load, overloaded servers experience a drop in throughout and an increase in response time. A situation that usually results in a violation of the SLA leading to penalties incurred upon the service-providers.

To counteract the overload problem, several approaches have been proposed in literature ranging from static server over-provisioning to dynamic server provisioning to employing admission control (AC) mechanisms. However, most of the proposed approaches have not considered the SLA while managing the overload problem. This paper proposes an SLA-aware admission control approach to manage the overload problem in Web-based applications hosted in a cloud environment. The proposed approach draws on previous work that considered a request-based profit-oriented AC approach that manages access to database systems stored on the cloud [6]. Although the request based AC approach is adequate for some applications, it is not suited for other applications that are based on sessions, which are composed of a sequence of requests such as e-commerce and retail applications. In session-based applications, if one request fails the whole session would fail leading to end-user dissatisfaction, and profit loss for the service provider. In addition, this paper proposes a new profit calculation model. Accordingly, the main contributions of this work are as follows:

PerCAM 14, October 15, 2014, Dubai, UAE.

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.257243

- An SLA-aware Profit-Oriented session-based Admission Control (POAC) mechanism for three-tier Web applications. POAC makes online admission control decisions based on the SLA contract to maximize the service-provider profit.
- A proposal for an SLA template that clearly states the terms and conditions for profit gained and penalties incurred upon service providers.

POAC was validated using simulations, which showed that POAC makes profit-aware decisions while considering the SLA.

The remainder of this paper is organized as follows. Section 2 surveys the related work. Section 3 presents POAC system architecture and technique. Section 4 describes our simulation setup and we present our findings and results in section 5. Finally, section 6 concludes the paper and outlines possible future work.

2. RELATED WORK

Admission control (AC) techniques applied to servers can be generally classified as request based AC and session based AC. In request-based AC techniques policies are applied on request-by-request basis, while for session-based AC techniques policies are applied on session-by-session basis, where a session can be composed of one or more requests. In this section, we present an overview of both schemas.

In [6], the authors proposed a profit oriented AC for Database as a Service (DaaS) hosted in the cloud. In such environment, bursty and unpredictable workload is very likely to occur causing server overload and impacting the service-provider revenue. The authors developed a prototype module called ActiveSLA with the objective of maximizing the service-provider's profit. ActiveSLA consists of two components: the Prediction module and the Decision module. The Prediction module predicts the probability of one query to meet its deadline using a non-linear classification model. The Decision module uses the output of prediction module along with the profit and penalty to decide either to accept or reject the query. An advantage of this technique is that it considers the SLA in its decision mechanism. However, a main drawback of ActiveSLA is that it considers a request-based admission control rather than a session-based admission control. Considering that in databases, transactions usually span more than one query request, applying ActiveSLA technique may result that the server accepts the first query in a transaction, and rejects the rest of the queries resulting in a non-complete job from the client's perspective.

The authors in [8] presented an admission schema that represented the base for several proposals in the field of the session based admission control (SBAC). In SBAC, when a session is rejected all its related requests will be rejected as well. However, the challenge in SBAC is to guarantee enough server resources to successfully complete a session, regardless of the time and capacity required to satisfy all of its related requests. The AC decisions for incoming sessions are obtained for a defined interval. The AC calculates the predicted utilization based on the measured utilization in the last time interval along with the recent utilization history. If the predicted utilization is bigger than a predefined threshold, which represents the server capacity, the new sessions will be rejected within the next interval. However, requests that belong to accepted sessions will continue to be accepted. If the predicted utilization drops below the threshold, the server starts to accept new sessions again. The drawback of this technique is that it uses a "predefined" threshold to represent the server capacity. Therefore human intervention is needed to specify a threshold value for each case according to the used hardware configuration and the expected workload.

Multi-programming level (MPL) is an AC technique in which the server defines a threshold for the maximum number of simultaneous requests that it can handle. The server accepts the incoming requests as long as the total number of concurrent requests is less than the MPL predefined threshold. Several contemporary works have proposed techniques to tune the MPL threshold to the optimum value for a given hardware and workload configurations [9], [10], [11].

3. PROPOSED SYSTEM ARCHITECTURE

3.1 Proposed SLA Contract

In current commercial cloud services, the SLA suffers from the following drawbacks:1) There is no guarantee on the performance of the computing service offered in terms of response time and throughput etc, 2) Most of the SLAs express the cost for a service per Virtual Machine (VM) image instance basis and not per request/session basis, 3) Usually the service-provider does not automatically discover SLA violations, but leaves the customer to present evidence of a violation and 4) the penalties are described as percentage of the customer bill and not based on request by request violations [12]. Thus, commercial SLAs contracts do not provide a real reflection of the service provided as they miss clear service specifications.

We argue that the SLA contract should include performance parameters that allow the customer to measure the quality of the service provided. In our proposed SLA contract, we integrated response time as a quality parameter. The SLA contract imposes an upper limit on the response time (RT_{SLA}) for each request issued by a customer, such that the response time for all clients' requests should be less than RT_{SLA} value. We use a per-customer SLA, which means that each customer will have an SLA document that will be applied to all requests arriving for the VM of that customer. This SLA document will be sent with the first request issued by that customer (i.e. session initiation request). In addition, the SLA document will define the profit and penalty notations. The service providers get the profit value for each request that meets the response time metric value in the SLA, while they pay the penalty for each unsatisfied request in case of a violation. The revenue that service provider gain will be equal to the sum of the profit values minus the penalty values for all received requests. The metrics used in our proposed SLA document are as follows

- Response time (RT_{SLA}): This metric represents the maximum time that is allowed for a server to serve the request and generate a response.
- Profit (PF_{SLA}): This metric represents the profit that the service provider will get for each request that has met its specified RT metric.
- Penalty (LT_{SLA}): This metric represents the penalty that the service provider has to pay for each request that misses its specified RT metric.

Table 1 shows an example of a typical SLA document. The first two columns, "SLA Metric" and "Metric Value", represent the metrics specified in the SLA and their corresponding values that

are sent by a customer. The last column "Metric Measured" shows whether the metric is measured or not.

3.2 POAC System Architecture

This work proposes a **P**rofit-**O**riented session based **A**dmission **C**ontrol (POAC) mechanism for three-tier web applications that are hosted in the cloud environment. To make the admission decision, the POAC records the usage level of resources and predicts the expected future usage in light of the per-customer SLA document.

In the cloud environment, the three-tier web applications are hosted on virtual machines (VMs) deployed on the Virtual machine monitor. The web applications represent the Software As A Service (SAAS) layer of the cloud-computing environment. POAC is implemented as a proxy in front of the web application. All requests coming to the web application are processed by POAC. Based on the server conditions, and the quality metrics defined in the SLA document, POAC forecasts the probability of satisfying a new session. Using this probability value, POAC decides whether to accept or reject the session-initiating request with an objective of maximizing the service-provider's profits while reducing any penalties.

As illustrated in figure 1, POAC intercepts the HTTP requests directed to the web applications and takes admission decisions. POAC system architecture is composed of four modules: the Monitoring Module, the Forecasting Module, the Profit Calculation Module, and the Decision Execution module.

3.2.1 Monitoring Module

The monitoring module monitors different system parameters that affect the system performance such as CPU capacity, memory utilization, incoming request information. These parameters are sent to the forecasting module to be used during its offline training phase and its online decision execution phase as will be explained in the next subsection. We classify performance parameters into the System parameters, Database parameters and Request parameters. The Database parameters and the System parameters are measured for a predefined interval while the Request parameters are measured per request. Table 2 summarizes the parameters specified for each type.

The parameters S1 to S4 represent the overall system state, which affects the overall performance of the system. These data represents the utilization of the CPU, memory, the speed of the disk and the available disk size.

For the Request parameters, R1 represents the type of requested operation. These operations can be determined during the training phase of POAC. The other parameters from R2 to R6 are the actual values that reflect the state of the overall web application system while executing a specific request.

Usually in the three-tier web applications, the database is the bottleneck tier. Therefore, detailed information about the database engine is required to correctly predict the overall system performance. This information is represented by parameters D2 to D5 and can be retrieved by directly querying the database. The "Slow Queries" parameter (D4) is system dependent and represents the number of queries that has been recorded to exceed a specific time threshold. The "Number of Queries" (D5) is the total number of the *select*, *insert*, *update*, *delete*, and *commit* commands performed against the database.

Table 1: SLA typical document example

SLA Metric	Metric Value	Metric Measured
(RT_{SLA})	<= 10 second	Yes, By Server
(PF_{SLA})	= 2 $	No, Provided By SLA
(LT_{SLA})	= 1 $	No, Provided By SLA

The DB buffer cache hits (D1) and system cache hits (S5) represents the memory cache parameters. The buffer cache is the cache used by database during its execution of the queries, while the system cache is the general cache that is used by the operating system for all of its operations without accessing the slow disk. The value, which is assigned for both cache, are the number of 'cache hits' from the performance counters of the DBMS and the system. The hit value represents the amount of data retrieved direct from the cache with no need to access the slow disk. The higher the value of the hits the higher speed of retrieving data and vice versa.

Using these parameters, offline/online training can be applied to the forecasting module to enhance the accuracy of its predictions.

3.2.2 Forecasting Module

The Forecasting module predicts the ability of the web server to meet the deadline as specified in the SLA. The module operates in two different phases; the offline training phase and the online prediction phase. In the offline training phase, machine learning techniques are used to determine server-overload state. Server overload is measured in terms of response time as a function of the level of system utilization using the values recorded by the monitoring module for the system performance parameters. Accordingly, the output of this phase is a set of tuples representing the combination of threshold values for the monitored parameters that defines server-overload.

Using these measurements, the forecasting module builds a non-linear regression model. A non-linear regression model builds a relationship between a dependent variable and one or more independent variables [13]. Our choice of non-linear regression model is based on the fact that web-application interactions usually exhibit a non-linear nature [14]. This offline model is built using different workloads and different mixes of sessions and requests.

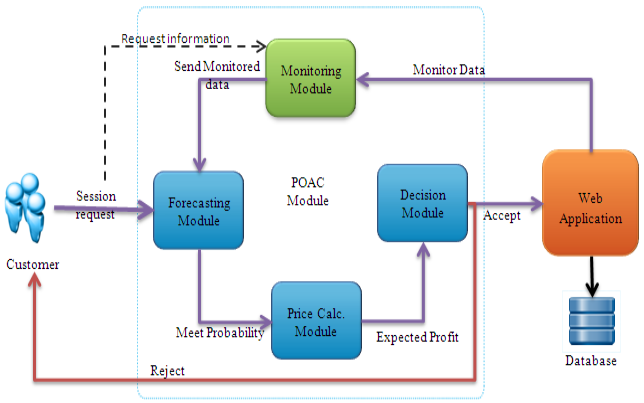

Figure 1: Admission Control Modules

Table 2: Monitored Parameters

		Parameter	Parameter Description
Request Parameters	R1	Request type	The requested Operation.
	R2	Response time	The exact response time of the request
	R3	Throughput	# of simultaneous requests
	R4	Avg. response time	The average response time in the last interval
	R5	Avg. throughput	The average throughput of all requests in the last interval
	R6	# of sessions	The total number of sessions that are run parallel.
Database parameters	D1	DB cache	The database buffer cache hits.
	D2	Threads running	# of threads running in the DB engine in the last interval
	D3	Bytes Exchanged	The total number of bytes that is sent and received from DB in the last interval
	D4	Slow Queries	The number of queries that have taken long time to complete in the last interval
	D5	Number Of Queries	The total number of statements that executed on the DB in the last interval
System Parameters	S1	Disk Space	The free space of the disk.
	S2	Disk I/O	Speed of I/O operations.
	S3	CPU Utilization	% of the CPU used.
	S4	Memory Utilization	% of the memory used.
	S5	System Cache	The OS cache hits.

In the online prediction phase, the monitoring module continues to provide periodic real-time measurements of the performance parameters to the regression module, which, on its turn, predicts the response time (RT_P) and calculates the probability of missing the deadline based on the response time defined in the SLA (RT_{SLA}) using equation (1). The probability of an event is given by dividing the outcomes of the event by the total number of outcomes in the sample space [15]. Therefore, as illustrated in equation (1), the probability of meeting the predicted response-time is calculated by computing the ratio of the predicted response-time value to the sum of the predicted response-time value (RT_P) and the desired response-time value (RT_{SLA}). The logic behind calculating the probability in such way is to check how close the predicted response-time value is compared to the response-time defined in the SLA.

$$\text{Probablity } (Prob) = \frac{RT_P}{(RT_P + RT_{SLA})} \quad (1)$$

3.2.3 Price Calculation Module

The Price Calculation module calculates the expected profit from accepting an incoming session. Based on this value, the Decision Execution module will make its final decision as to whether or not an incoming session should be accepted. The Decision Execution module attempts to maximize the profit and minimize the probability of violating the SLA. When calculating the expected profit (EP), the following three parameters are considered; the SLA profit metric, the SLA penalty metric, and the probability of missing the deadline. The first and second parameters are taken directly from the SLA definition. The third parameter is calculated using equation (1).

The expected profit is calculated using equation (2), where (EP) represents the Expected Profit, PF_{SLA} represents the SLA specified Profit, LT_{SLA} represents the SLA specified penalty, and Prob represents the probability of missing the deadline as defined in equation (1). Equation (2) is adapted from the profit-based admission mechanism that was proposed in [16] for provisioning connection requests in optical networks. However, equation (2) differs from that defined in [16] in two main aspects; First we consider the SLA. Second, we define equation (1) for calculating the probability

$$EP = PF_{SLA} - Prob * LT_{SLA} \quad (2)$$

3.2.4 Decision Execution Module

In the Decision Execution module, the final admission decision is made whether to accept or reject the incoming session. The Decision Execution module will make its decision based on the following equation.

$$F_{AC} = \begin{cases} admit, & EP > Pofit\ Ratio \\ reject, & otherwise \end{cases} \quad (3)$$

,where F_{AC} represents the final decision for the admission control module, the (EP) is the expected profit that is calculated by the Price Calculation module, and the "Profit Ratio" is a value that ensures that only requests with high probability of achieving a profit will be accepted. We define the "Profit Ratio" using the following equation

$$Profit\ Ratio = \frac{PF_{SLA}}{(PF_{SLA} + LT_{SLA})} * PF_{SLA} \quad (4)$$

Again the PF_{SLA} and LT_{SLA} are the profit and penalty as defined in the SLA respectively. The "Profit Ratio" represents the ratio of the potential profit gain to the possible penalty loss. Thus, when the penalty value in the SLA increases, the "Profit Ratio" will decrease. Therefore, the probability of acceptance will decrease as well.

Following is an example that illustrates POAC execution. Let's assume that a request is sent from a customer with an SLA document that contains the following constraints

- The response time (RT_{SLA}) is 10 second.
- The profit (PF_{SLA}) of meeting the RT_{SLA} = 1$
- The penalty (LT_{SLA}) of missing the RT_{SLA} = 2$

When receiving the request, the forecasting module predicts the response time. For the sake of this example we will assume the

predicted response time is 4 sec. The forecasting module will then calculate the probability of meeting the predicted (RT_P) using equation (1) as shown below

$$Prob = \frac{RT_P}{\left(RT_P + RT_{SLA}\right)} = \frac{4}{\left(4 \quad + \quad 10\right)} = \frac{4}{14} = 0.29$$

Then the request will be sent to the price calculation module to calculate the expected profit (EP) using equation (2) as follows

$$EP = PF - Prob * LT = 1 - \left(0.29 \ * \ 2\right) = 1 \ - \ 0.58 = 0.42$$

The request will then be handed to the Decision Execution module to evaluate the final decision using equation (3). First the module will calculate the "Profit Ratio" using equation (4)

$$Profit\ Ratio = \frac{PF_{SLA}}{\left(PF_{SLA} + LT_{SLA}\right)} * PF_{SLA} = \frac{1}{\left(1+2\right)} = 0.33$$

Therefore, the module will evaluate the final decision by comparing the expected profit with "Profit Ratio". Thus, in this example the final decision will be to accept, as the expected profit (EP) is bigger than Profit Ratio.

4. EXPERIMENTAL ENVIRONMENT

To validate POAC we have set up an experimentation environment to simulate a three-tier web application using the TPC-W Benchmark [19] and workload generator. We present the experimentation environment in this section and the details of the simulations and results in the next section.

4.1 Hardware and software

The experiments were performed on one IBM laptop Lenovo T430 server. The machine has Intel Core i5-3320M CPU with 2.60 GHz, 2601 MHz, 2 Core(s), and 4 Logical Processor(s). It has 8 GB RAM and 300 GB Hard disk with 7200rpm. The server machine runs Windows 7 Professional - 64bit version 1.09.00.AG B10. The web server used is the Apache tomcat version 6.0.28 with MYSQL version 5.6.14 (innodb version 5.6.14) as the database server. The POAC is implemented using the JAVA 1.6.0 programming language and were hosted in a proxy on the same server machine. It intercepts all the calls to the web server to make admission decisions. If the request is accepted, it is passed to the web server in which the application logic is performed and the corresponding database statements are executed on the database tier as required. The database tier sends the statements to MYSQL using JDBC API through MYSQL database connector.

In this work, we used the machine-learning package WEKA to implement the non-linear regression model. WEKA contains several machine learning algorithms [17]. We choose the "Additive Regression" that utilizes REPTree. The REPTree applies information gain to build a regression tree using reduced-error pruning [18].

4.2 TPC-W Benchmark

We used the TPC-W benchmark [19] to simulate the three-tier web application. TPC-W is an e-commerce web application that can simulate the functions and procedures of a transaction server. Despite being deprecated, the TCP-W benchmark has been used in most of the contemporary works addressing admission control [6], [7], [14]. In this work we use TPC-W to measure the capability of our algorithm to honor the SLA metrics in a heavy loaded environment like e-commerce applications. In such environment, several concurrent users could access the application for a long time performing a shopping session, which simulates similar activities in the cloud-computing environment. Therefore the web server has to wisely accept and reject sessions with profit-oriented decisions in consideration.

As mentioned by the TPC-W specification [19], the TPC-W benchmark consists of fourteen business functions that are common in an Internet commerce application. It provides operations for browsing, searching, shopping and ordering items. The specification defines eight tables in which the data utilized by web applications are stored. The tables are Author, Item, Order line, Orders, Credit card, Customer Address and Country. In addition, we employ two more operational tables, shopping cart and shopping cart line. Theses extra tables provide the required functions for the shopping cart to operate as desired.

We built a custom generator to populate the database with test data. The generator is configured to generate a test data for a 100K Items and 800 emulated browsers (EB). The total size of the test data in the database is 3.43 GB with an additional 2.4 GB of GIF images. Each item in the database has one normal image and one thumbnail image.

The TPC-W specification defines three types of web interactions mix, namely, the browsing mix, the shopping mix and the ordering mix. In this work we choose to work with the browsing mix to perform our experiments since we argue that the browsing activity usually generates a longer sequence of requests than the other two mixes.

4.3 Workload Generator

The University of Wisconsin TPCW implementation [20] is used as a base for our three-tier web application. It implements both the server side as well as the workload generator. The workload generator follows the closed loop generation model in which a new request is generated only after the response to the previous request is received [21]. We chose the closed workload because it simulates the behavior seen in the e-commerce web applications, where the user waits for the response before issuing the next request.

The workload generation process starts by generating a number of EBs based on a pre-configured value. The EB represents a virtual user that mimics the behavior of a real user on the system. Each EB starts its process by opening a session, and then it sends the first request to the web server and waits for the response. After it receives the response, it waits for a specific period of time that simulates the think time of a normal user before generating the next request. The think time is exponentially distributed with a mean of 7 seconds and a maximum value of 70 seconds. This process is continued until the session is finished. Once the session is finished, the generator logs the session results and then starts a new one. In addition to the normal required parameters of the requests, the EBs send the SLA profit, penalty and the response time to the server to be used by the admission control proxy.

In all the experiments conducted, we used the same settings to derive the system towards its overload state. The number of EBs is 800. The EBs are started in an incremental way such that the ramp up period lasts for 100 seconds. The ramp down period takes another 100 seconds when the generator starts to terminate the EBs. The generator continues to generate requests against the

server for 1000 seconds. This means that the total time of experiment execution lasts for 1200 seconds.

4.4 Evaluation Equations

For comparison purposes, we define the following equations and use them in the experimental evaluation in section 5. The first equation is the Meet Power equation, which assesses the effectiveness of the proposed admission control technique.

$$Meet\ Power = \frac{N_{Meet}}{T_{Total}} \quad (5)$$

,where N_{meet} is the total number of requests that meets the deadline, and T_{total} is the total number of incoming requests including both accepted and rejected requests.

In the ideal case, the Meet Power will be equal to 1 indicating that all received requests are accepted and have met their deadline. However, in real situations, Meet Power will be less than 1 since N_{Meet} will be less than T_{Total} indicating that the server has rejected some of the received requests. As the server successfully process the incoming requests, the Meet Power approaches 1 as the ratio of N_{Meet} to T_{Total} increases. On the contrary, as the server rejects incoming requests the Meet Power approaches 0 as the ratio of N_{Meet} to T_{Total} decreases

Another measure for the efficacy of the POAC mechanism is the Profit Percentage (profit %), which represents the percentage of achieved net-profit with respect to the expected-profit. Profit Percentage is calculated in equation (6).

$$profit\ \% = \frac{Real\ profit}{Expected\ profit} * \quad 100 \quad (6)$$

, where the real profit represents the net profit achieved by the service provider and the expected profit represents the total achievable profit at the time of accepting requests. Real profit and expected profit are calculated using equations (7) and (8) respectively, where PF_{SLA} and LT_{SLA} are profit and penalty values as defined in the SLA document. The N_{miss} is total number of requests that misses the deadline and T_{ACC} is the total number of accepted requests.

$$Real\ profit \ = \ \left(N_{meet} * PF_{SLA}\right) \ - \ \left(N_{miss} * LT_{SLA}\right) \quad (7)$$

$$Expected\ profit \ = \ T_{ACC} \ * \ PF_{SLA} \quad (8)$$

In the best case, the Profit Percentage will be equal to 100% indicating that all accepted requests have met their deadline. Therefore, when some requests miss their deadline, the real profit will be less than expected profit. Thus, the higher the expected profit the less the requests that miss their deadline.

5. EXPERIMENTAL EVALUATION

We conducted several simulations to validate the efficacy of our algorithm to take profit-oriented decisions using different profit and penalty values. We further conducted comparisons between POAC, MPL techniques and when no admission control is applied. Each simulation presented was performed 10 times. In addition, we show the average (AVG), variance (VAR), standard deviation (STD) and confidence interval (CI) at 95% for our metrics.

5.1 POAC Evaluation

For POAC, we performed the experiments with different profit/penalty configurations. Three different configurations were tested. 'POAC 1-1' refers to the case when the profit and penalty have the same cost. 'POAC 1-2' refers to the case when the penalty has twice the cost of the profit, and 'POAC 2-1' refers to the case when the profit has twice the cost of the penalty. The Meet Power values are shown in Table 3. There are no wide discrepancies among the recorded values, which demonstrate that the proposed AC shows a stable performance with different profit/penalty configurations. However, POAC performance comes with a price. To make a decision, a request needs to be processed by POAC's four modules. This adds to the total response time of the request, which might occasionally violate the SLA, especially for very small response time values. In addition, building the non-regression model incurs extra delays. The regression model is build periodically online. This consumes some of the server's processing power. Normally this overhead can be negligible. However it could affect the server performance especially during peak load situations

As shown by Table 3, "POAC 1-2" rejects more requests than "POAC 1-1" which in turn reject more than "POAC 2-1". This can be attributed to the fact that for "POAC 1-2" the penalty is too high, which makes the algorithm more aggressive in rejecting the requests. In contrast to "POAC 2-1", where the algorithm becomes less aggressive in rejecting the requests as more profit gains are possible. The Missing and Meeting percentages for the different profit/penalty configurations are similar. However "POAC 1-2" yields the lowest Missing percentage as it rejects more requests. Table 4 shows the Profit and the Profit% values. As expected, "POAC 2-1" achieves the best Profit and Profit% as the profit value equals twice the penalty value, while "POAC 1-2" achieves the lowest values.

5.2 Multi-programming Level Evaluation

In order to evaluate their proposed AC algorithm, the authors in [14] calculated the best MPL value for a request-based technique. We followed the same approach to calculate the best MPL value for our scenarios. However, we calculated the best MPL value for a session-based technique for our environment using experiments. Accordingly, for our experiments the MPL limit represents the maximum number of sessions to be accepted by the server. We conducted a set of experiments in which we varied the MPL threshold values. The MPL values used are 1000, 1500, 2000 and 2500. For each experiment, the server will accept new sessions as long as the total number of concurrent sessions is less than configured MPL value. Once the number of concurrent sessions exceeds the configured MPL value, all new incoming sessions will be rejected until the number of active concurrent sessions drops below the predefined MPL threshold. In our experiments an active session is closed when the client sends a request to the server to invalidate the session.

Table 5 shows the simulation results obtained for different MPL values. It shows that the Meet Power increases as the MPL value increases with an average of 86.48% at MPL 2000. This can be attributed to the fact that as the MPL value increase less requests will be rejected. However, accepting requests without satisfying their deadline will result in a decrease in the Meet Power as seen in the case of MPL 2500, where number of missed-requests increases affecting the calculated Meet Power values.

Table 3: Proposed Algorithm (POAC) Experiments results

	Meet Power				Miss				Reject			
	AVG	*VAR*	*STD*	*CI (95%)*	*AVG*	*VAR*	*STD*	*CI (95)%*	*AVG*	*VAR*	*STD*	*CI (95%)*
POAC (1 - 1)	90.96	0.34	0.59	± 0.36	7.68	0.18	0.43	± 0.27	16.27	5.61	2.37	± 1.47
POAC (2 - 1)	93.21	1.27	1.13	± 0.7	6.26	0.46	0.68	± 0.42	6.86	7.22	2.69	± 1.67
POAC (1 - 2)	91.87	0.52	0.72	± 0.45	5.98	0.13	0.36	± 0.22	23.29	3.96	1.99	± 1.23

Table 4: Proposed Algorithm (POAC) Profit results

	Net Profit				Profit%			
	AVG	*VAR*	*STD*	*CI (95%)*	*AVG*	*VAR*	*STD*	*CI (95%)*
POAC (1 - 1)	102,120.4	2,323,406.71	1,524.27	± 944.74	84.64	0.74	0.86	± 0.53
POAC (2 - 1)	213,851	12,717,660.44	3,566.18	± 2210.3	88.50	0.37	0.61	± 0.38
POAC (1 - 2)	92,974.2	5,261,503.51	2,293.80	± 1421.68	77.76	4.99	2.23	± 1.38

Table 5: Different MPL Experiments Results

	Meet Power				Miss				Reject			
	AVG	*VAR*	*STD*	*CI (95%)*	*AVG*	*VAR*	*STD*	*CI (95)%*	*AVG*	*VAR*	*STD*	*CI (95%)*
MPL 1000 (1 - 1)	53.25	3.07	1.75	± 1.09	7.93	0.40	0.63	± 0.39	90.1	0.43	0.65	± 0.41
MPL 1500 (1 - 1)	72.81	0.34	0.58	± 0.36	9.73	0.03	0.17	± 0.1	74.98	0.59	0.77	± 0.48
MPL 2000 (1 - 1)	86.48	3.60	1.90	± 1.18	12.45	3.89	1.97	± 1.22	13.7	3.67	1.92	± 1.19
MPL 2500 (1 - 1)	85.8	2.09	1.45	± 1.18	14.2	2.09	1.45	± 0.9	NA	0.00	0.00	NA

Table 6: Different MPL Experiments Profit results

	Net Profit				Profit%			
	AVG	*VAR*	*STD*	*CI (95%)*	*AVG*	*VAR*	*STD*	*CI (95%)*
MPL 1000 (1 - 1)	58,598.2	6,602,724.62	2,569.58	± 1592.61	84.14	1.61	1.27	± 0.79
MPL 1500 (1 - 1)	78,933	814,956.67	902.75	± 559.52	80.55	0.11	0.33	± 0.21
MPL 2000 (1 - 1)	90,535.7	22,993,141.57	4,795.12	± 2971.99	75.09	15.57	3.95	± 2.45
MPL 2500 (1 - 1)	85,767.3	13,328,963.82	3,650.89	± 4133.7	71.61	8.36	2.89	± 1.79

Table 7: Different approach experiments results

	Meet Power				Miss				Reject			
	AVG	*VAR*	*STD*	*CI (95%)*	*AVG*	*VAR*	*STDEV*	*CI (95)%*	*AVG*	*VAR*	*STDEV*	*CI (95%)*
NO AC Case	83.44	4.72	2.17	± 1.35	16.56	4.72	2.17	± 1.35	NA	0.00	0.00	NA
POAC (1 - 1)	90.96	0.34	0.59	± 0.36	7.68	0.18	0.43	± 0.27	16.27	5.61	2.37	± 1.47
MPL 2000 (1 - 1)	86.48	3.60	1.90	± 1.18	12.45	3.89	1.97	± 1.22	13.7	3.67	1.92	± 1.19

Table 8: Different approach Profit results

	Net Profit				Profit%			
	AVG	*VAR*	*STD*	*CI (95%)*	*AVG*	*VAR*	*STD*	*CI (95%)*
No AC Case (1- 1)	82,560	31,954,083.11	5,652.79	± 3503.57	66.88	18.87	4.34	± 2.69
POAC (1 - 1)	102,120.4	2,323,406.71	1,524.27	± 944.74	84.64	0.74	0.86	± 0.53
MPL 2000 (1 - 1)	90,535.7	22,993,141.57	4,795.12	± 2971.99	75.09	15.57	3.95	± 2.45

When considering the Rejection and Miss rates, we find that MPL 1000 scored the highest Rejection rate and the least Miss rate. As the MPL increases, the Rejection percentage decreases, while the Miss percentage increases. However, over rejecting requests that could have been otherwise accepted and met their deadline affects the Meet Power, which recorded the lowest percentage at 53.25% for MPL 1000.

Table 6 shows the effect of the MPL-based AC decisions on the Net Profit. The MPL 2000 achieves the highest Profit with 90535.7 on average. As for the Profit%, MPL 2500 recorded the lowest Profit%. That can be attributed to the fact that the number of Miss requests increases as the MPL value increase. From the above results we can deduce that MPL 2000 is the best MPL values for our environment.

5.3 POAC vs. MPL vs. NO AC

To further evaluate POAC performance in achieving high profit gains, we compared our different performance metrics in the following three cases; "POAC 1-1", MPL 2000, and no admission control applied for the same profit and penalty values. Table 7 shows the Meet Power, the Miss and Reject values for the different cases. POAC achieves the best Meet Power and lowest Miss Percentage on average. The NO AC case achieves the lowest Meet Power and the highest Miss percentage in all cases. In addition, Table 7 shows that POAC rejects more requests than the MPL. However POAC achieves better Miss Percentage on average. This because POAC makes a profit oriented AC decisions, while MPL AC makes its decisions based on the total number of simultaneous sessions. Thus, the MPL AC could accept sessions in spite of an overloaded server, only because the total number of sessions is below the preset threshold value. Furthermore, MPL AC could reject sessions in spite of having the server unloaded only because the number of accepted sessions has exceeded the preset threshold value. In contrast, POAC performs educated decisions based on learning the effect of workload on the performance of the server.

Table 8 shows that POAC achieves the best Net Profit among the two other approaches. The Net Profit for POAC is 102,120.4 on average, while MPL 2000 achieved 90,535.7 on average. Furthermore, POAC achieves a better Profit% than the MPL AC as indicated by the values in Table 8.

Figures 2 and 3 show the Miss percentage and the Profit over the time in seconds for the different AC approaches for one of the experiments. It can be seen that POAC is performing well over time in comparison to the two other approaches. It achieves the least Miss percentage and the best Profit value. Figure 4 shows the rejection of POAC in comparison to MPL AC. The MPL AC rejection graph shows a step-wise like function over the time. The graph also shows that the first request-rejection occurred after some time (i.e. around the second 210). This is because the MPL does not start to reject any requests until the threshold value has been reached regardless of the server load. In contrast, POAC starts the rejection decisions earlier (i.e. around second 20) as it starts to predict that the server will be overloaded soon.

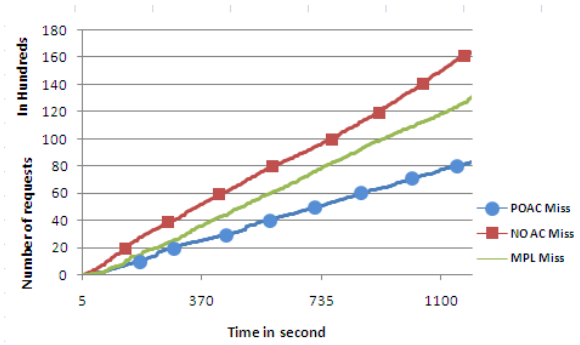

Figure 2: Miss Requests over time for different AC approaches

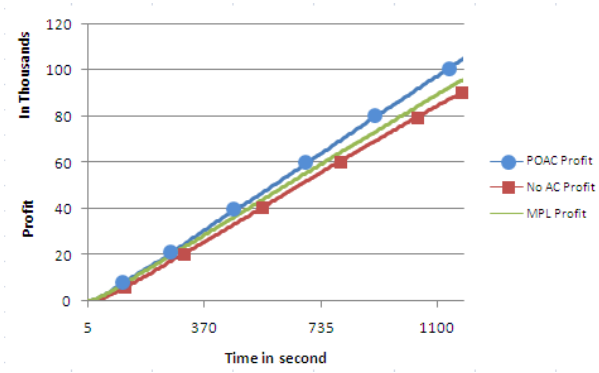

Figure 3: Profit over time for different AC approaches

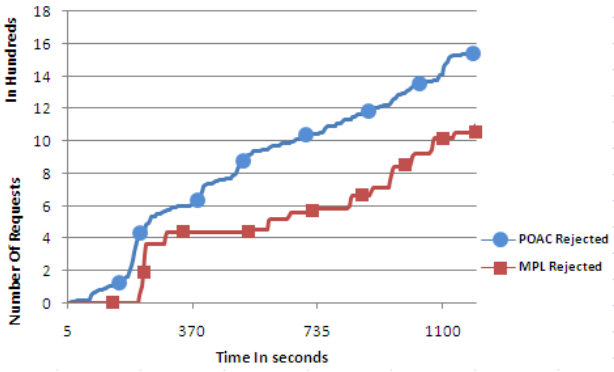

Figure 4: Reject requests over time for different approaches

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a Profit Oriented Admission Control schema for web applications hosted in the cloud. POAC applies session-based admission control decisions based on a predefined SLA quality metrics. POAC can be applied to any three-tier web application hosted in the cloud. Our experiments show that POAC decisions results in profit gains in comparison to other AC approaches. As future work, we plan to enhance POAC for dynamic provisioning to address cloud elasticity such that it can decide on whether or not more VMs are required to satisfy peak load situations. We also plan to experiment with other machine-learning techniques to improve on POAC prediction and forecasting decisions.

7. REFERENCES

[1] Zhang, Q., Cheng, L., and Boutaba, R. Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 1, 1 (2010), 7-18.

[2] Patel, P., Ranabahu, A. H., and Sheth, A. P. Service Level Agreement in Cloud Computing. Wright State University, 2009.

[3] Khajeh-Hosseini, A., Sommerville, I., and Sriram, I. Research Challenges for Enterprise Cloud Computing. In the 1st ACM Symposium on Cloud Computing, SOCC (2010), 1-11.

[4] Endo, P. T., Gonçalves, G. E., Kelner, J., and Sadok, D. A Survey on Open-source Cloud Computing Solutions. In 8th Workshop on Clouds, Grids and Applications (Gramado, Brazil 2010), 3-16.

[5] Chen, Y., Iyer, S., Liu, X., Milojicic, D., and Sahai, A. SLA Decomposition: Translating Service Level Objectives to System Level Thresholds. In Proceedings of the Fourth International Conference on Autonomic Computing ICAC '07 (Jacksonville, FL 2007), 3.

[6] Xiong, P., Chi, Y., Zhu, S., Tatemura, J., Pu, C., and Hacıgümüs, H. ActiveSLA: A Profit-Oriented Admission Control Framework for Database-as-a-Service Providers. In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '1) (2011).

[7] Elnikety, S., Nahum, E., Tracey, J., and Zwaenepoel, W. A method for transparent admission control and request scheduling in e-commerce web sites. In Proceedings of the 13th international conference on World Wide Web (WWW '04) (2004), 276 - 286.

[8] Cherkasova, L. and Phaal, P. Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites. IEEE Transactions on Computers, 51, 6 (June 2002), 669-685.

[9] Carey, M. J., Krishnamurthi, S., and Livny, M. Load control for locking: the "half-and-half" approach. In Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS '90) (1990), 72-84.

[10] Moenkeberg, A and Weikum, G. Conflict-driven load control for the avoidance of data-contention thrashing. In Proc. Seventh International Conf. on Data Engineering (Kobe 1991), 632 - 639.

[11] Heiss, H. and Wagner, R. Adaptive Load Control in Transaction Processing Systems. In Proceedings of the 17th International Conf. on Very Large Data Bases (VLDB '91) (1991), 47-54.

[12] Baset, S. A. Cloud SLAs: present and future. ACM SIGOPS Operating Systems Review, 46, 2 (July 2012), 57-66.

[13] (2014, April) Non Linear regression model. [Online]. http://en.wikipedia.org/wiki/Nonlinear_regression.

[14] Tozer, S., Brecht, T., and Aboulnaga, A. Q-Cop: Avoiding Bad Query Mixes to Minimize Client Timeouts Under Heavy Loads. In Conference on Data Engineering (ICDE), IEEE 26th International (Long Beach, CA 2010), 397 - 408.

[15] (2014, April) Probability definition. [Online]. http://www.mathsteacher.com.au/year10/ch05_probability/04_probability/prob.htm

[16] Das, A. Maximizing profit using SLA-aware provisioning. In Network Operations and Management Symposium (NOMS), IEEE (Maui, HI 2012), 393 - 400.

[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The WEKA data mining software: An update. ACM SIGKDD Explorations, 11, 1 (June 2009), 10-18.

[18] Witten, I. H. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Series in Data Management Systems, 2005.

[19] (2014, April) Transaction Processing Performance Council, "TPC-W Benchmark (e-commerce)". [Online]. http://www.tpc.org/tpcw/default.asp

[20] (2014, April) TPC-W Java Implementation Distribution at University of Wisconsin. [Online]. http://pharm.ece.wisc.edu/tpcw.shtml

[21] Schroeder, B., Wierman, A., and Harchol-Balter, M. Open Versus Closed: A Cautionary Tale. In Proceedings of the 3rd conference on Networked Systems Design & Implementation (NSDI'06) - Volume 3 (2006).