SLA-based Dynamic Pricing Model for Cloud Services

Taj Eldin Suliman M. Ali, Hany H. Ammar

1 College of Graduate Studies, Computer Science and Information Technology
Sudan University for Science and Technology, Khartoum, Sudan
2 Lane Department of Computer Science and Electrical Engineering, College of Engineering and Mineral Resources, West Virginia University, Morgantown, USA

Abstract

In the paper, we use business service provider and service provider interchangeably. Cloud computing has become a widely used technology in recent years. An important problem in this technology is how to decide the cost of cloud computing services. Cloud computing cost models have been the subject of intensive research. Much of the published work focused on static pricing models based on service level agreements or they do not include clearly the needed brokerage services. To tackle this problem, we propose a brokerage-based dynamic model that includes service level agreements negotiation and monitoring. The model helps service providers and customers to evaluate service's performance and cost. We describe the pricing model using a Business Process Model and Notation (BPMN), and present an architectural model showing the main components. We then illustrate the model using a simple example. This model focuses on monitoring and metering cloud services at run time based on current service's features which are governed by third party, broker.

Keywords: Cloud Computing; Service Level Agreement (SLA); Broker; Dynamic pricing; Quality of Service (QoS);

1. INTRODUCTION

Cloud computing [22] is a new computing paradigm that uses the Internet for provisioning virtual computing resources (services) on-demand. Cloud providers can deploy their resources at different shapes including hardware,
software, storage, and network. Commonly, several scientific domains (e.g. education, health, economic…etc) accept the concept of Cloud computing. The number of providers has been increased dramatically as well as the number cloud customers. At this competitive environment – where service providers are competing in the service's type, cost, quality of Service (QoS), performance…etc. – , service customers face several problems, one of which is how to find a suitable provider which can deploy services which satisfy their needs. The second is that the service configuration may change unpredictably during run time, but the service cost remains as it is.

The paper is further arranged in this way that section II discusses several terminologies that have been used to design the model. The already done studies related to this topic are discussed in section III. Our proposed model is discussed in section IV. We conclude our paper in section V.

1.1 Cloud Computing

“A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumer.” [1].

To provision services, Cloud computing uses several deployment models; Public Cloud, by which Cloud provider deploys services to general customers. Private Cloud, by which Cloud provider provides services to particular customers in an organization. Community Cloud, this model supports service
deployment to customers (organizations) that have common objectives and policies. **Hybrid Cloud**, it is a combination of two or more mentioned deployment models.

Cloud providers have different ways to offer services to customers such as software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS). At **SaaS model**, cloud services (e.g. Facebook and Twitter) are hosted in provider’s company (e.g. Google Docs). At **PaaS model**, cloud providers (e.g. Azure) deploy programming environments, tools, and configuration management to software developers. Google AppEngine is an example of PaaS provider. At **IaaS model**, IaaS services such as virtual machines (VMs), storage places are deployed to systems developers. Amazon EC2 is an example of IaaS provider.

1.2 **Service Level Agreement (SLA)**

1.2.1. **What is SLA?**

The output of a successful negotiation between Cloud provider and its customers is an agreement between them. This agreement called Service Level Agreement (SLA), which contains the service's terms and conditions (e.g. specific price and QoSs), the parties' names, and actions that can be taken if one of service parties violates the agreement.

Since 1980s several domains have been implementing SLA, and each of them uses SLA at a different view, due to that several SLA definitions have been emerged. Some of these definitions are:

- **Internet domain:** “SLA constructed the legal foundation for the service delivery. All parties involved are users of SLA. Service consumer uses
SLA as a legally binding description of what provider promised to provide. The service provider uses it to have a definite, binding record of what is to be delivered”.[2]

- **Cloud domain:** “An explicit statement of expectations and obligations that exist in a business relationship between two organizations: the service provider and customer”. [3]

1.2.2. **SLA Components:**
SLA has been used by several domains to manage treatments between service providers and their customers. Due to realizing this objective, SLA has multiple components, and each of them should play a specific role. Jin et al. [4] discussed those roles comprehensively, as follow:

- **Purpose:** it is parties' objectives which can be realized during SLA usage.
- **Restrictions:** the important constraints that should be considered to ensure that the provider provides a service as it requested.
- **Validity period:** a period during which an SLA has been alive.
- **Scope:** defines two types of services, services that should be provided to customers and others that will not be covered in SLA.
- **Parties:** defines providers and their customers (companies or citizens).
- **Service-level objectives (SLO):** Specifies the agreed QoSs such as reliability, performance, and availability.
- **Penalties:** determine punishments that should occur when SLOs does not achieve.
- **Optional services:** define services that are not basic but might be needed.
- **Administration:** tasks that are used for SLOs achievement assurance.

1.2.3. **SLA Lifecycle:**
As it was mentioned at [5] Ron et al. discuss the lifecycle of an SLA in three phases, as follow: **Creation**
phase: at this phase, customers select a suitable service provider that can deploy a required service. Operation phase: in which a customer has read-only access to the SLA. Removal phase: in which an SLA is terminated and all associated configuration information is removed from the service systems.

Later, six’s phases SLA lifecycle has been detailed by the Sun Microsystems Internet Data Center Group [54], as follow: Discover service providers: in which a third party (Broker) selects provider based on consumer's needs. Define SLA: SLA terms such as services, parties, penalty policies and QoS parameters will be defined in this phase. After terms definition, a negotiation between service parties will be established. Establish agreement: during it, service parties use SLA template for agreement commitment. Monitor SLA violation: in which the monitor agent (Broker) measures the QoSs of deployed service versus the contract. Terminate SLA: in which SLA will be terminated. Enforce penalties for SLA violation: it is the last phase in which if there is any violation, its penalty are implemented.

1.3 Pricing Cloud services

At the Cloud environment – an internet – based parallel and distributed computing environment –, a single service can be deployed and executed at multiple clients, each of which has a different form (multitenancy). Each Cloud customer may use a single service with a different configuration and it may be changed at any times, therefore the service cost should not be static.

Due to the dynamic nature of the cloud environment and the providers’ competitions, the main
objective of the cloud provider is maximizing revenues by improving customer satisfaction level; as well the main objective of cloud customer is minimizing cost. Static pricing mechanisms have some drawbacks like over paying – service is provisioned over the customer needs, so customers are paying over their real usage – as well as an opposed problem named under-provisioning. To achieve the objectives of the parties – providers and customers –, static pricing was replaced by dynamic pricing. According to Yeo et al. [6], fixed pricing model is more straightforward and easy to understand, but it is not fair for all customers because they are not having the same needs. Dynamic pricing – sometimes called demand pricing, or time-based pricing – is a pricing strategy in which Cloud providers set flexible prices for service depends on some factors such as current supply and demands, competitor pricing, and other external factors in the market.

1.4 Cloud Broker
Due to the rapidly increasing of Cloud providers and their services, the Cloud customers often fail to find a suitable provider or select a service that satisfies their needs. Those services are distributed, and provided by providers who have different administrative policies. Although the cloud services can be accessed by a huge number of customers, the configuration of those services can be updated randomly. This environment supports a lot of opportunities coming to improve business organizations, but at the same time, there are some challenges that
should be faced. Cloud brokerage has lately been emerged to achieve those opportunities as well tackle those challenges.

Cloud broker is a match-maker between service's parties; it has several roles such as negotiation establishment, SLA management, resources allocation, price determination, and billing. Cloud broker provides a uniform interface that can be used to manage treatments between providers and their customers [7]. Broker profit represents the difference between the "buying price" – between broker and provider – and the "selling price" – between broker and customer [8].

1.5 BPMN

The Business Process Modeling and Notation (BPMN) is a new standard to model business process flows and web services. BPMN provides a graphical notation that can be used to specify business processes in a business process diagram. This diagram is readable and understandable by all business stakeholders such as business analysts, technical developers, and the business managers. We use the BPMN to specify cloud service lifecycle, by one diagram we specify service's selection, provisioning, monitoring, and billing.

2. RELATED WORK

Authors of [9] propose a Conceptual platform for the third party between the Cloud providers and the consumers. Through this platform, Cloud customers can search and select the services, which are advertised by the Cloud providers. If the customer finds the service which meets his requirements, the platform enables him to negotiate with the provider. However, the negotiation at this work is done directly between provider and its customer, not
through Broker. Also, this platform has a week monitoring process.

At [10] authors present framework which supports service providers to publish their services at a directory, in which service customers can easily select service from it and determine their required QoSs measures. This work focuses on SLAs reliability and web services conformity issues. Also uses Cloud broker for web services composition and real-time testing. The framework focuses on the customers' service level objectives (SLOs) and SLA performance capabilities. However, this framework ignores the measuring and monitoring of the QoS parameters.

Park Ki-Woong et al. [11] propose a secured mutually verifiable billing system in order to solve future conflicts that may occur between service's parties. They are only considered the reliability issue of purchasing transactions. They ignore some broker features such as pricing, resource management, and refunding. Rogers Owen et al. [12] present a resource allocation mechanism, but their work does not focus on resource prediction and billing. Wang Wei et al. [13] propose a brokerage service for on-demand reservation of IaaS resources. Their work considered only on-demand tasks and ignores anything beyond that. At [14], Jrad Foued et al. present a generic architecture of broker; their work considers SLA management and interoperability issues. However, they do not focus on resource management feature of brokerage. Shadi Ibrahim et al. [15] present the concept of fairness in pricing in respect of micro-economics, their work does not show how to
calculate services' price based on its types.

3. THE PROPOSED PRICING MODEL

3.1. Model Components:
As depicted at Fig1, the model has three components that are Customer, Broker, and Provider. This section discusses those components in details.

Customer: Represents business organization that contains tens or hundreds of employees. This organization requests services with specific QoSs from the Broker. Due to the customer has not sufficient information about providers, the broker negotiates with them on behalf of the customer for service deployment. At the end of a successful negotiation, the customer receives and stores a copy of the SLA, and then he can use the service. During the service usage, if a violation occurs the customer receives violation message which determines violation's type and cost. Monthly, the customer receives the service's cost for the last month.

Broker: It is a trusted third party, which accepts customers' requests for service deployment. The main objective of the Broker is to assess customer to find a suitable service from the best provider. To achieve the objective, the Broker makes a negotiation with multiple cloud providers based on the customer's functional and non-functional requirements. Through a negotiation, the Broker determines suitable provider and selects the service from which, deploys service to the customer, monitors its usage, and calculates its cost based on the agreed service parameters stated at the SLA. The establishment of these tasks will be done through the following Broker activities:
• **Explore Provider:** this activity uses service catalog to perform two things; one of which is exploring service providers that can satisfy customer's requirements. The second is mapping the values of customer's QoS parameters to corresponding providers' QoS parameters. In Cloud computing systems, resources' availability and capability are dynamic in nature, therefore some important issues such as the ability of provider to deal with service scalability, QoSs dynamic changes, and resources heterogeneity should be considered during this activity

• **Negotiate SLA:**
Negotiation is a bi-directional dialogue between service's parties who have conflicting goals and need to achieve acceptable agreement [16]. Sometimes the negotiation process is more complex due to these conflicting goals. For example, a customer tries to decrease a service price, whereas the service owner – provider – wants to increase a service price. For a negotiation goals achievement, a negotiation parties use some negotiation strategies. This complexity affects the human participants' efficiency; as a result, autonomous negotiation replaces the traditional – human – negotiation.

To automate a negotiation, some specification languages, and some negotiation protocols will be needed. Several specification languages have been developed for defining services with Service Level Agreement (SLA). The two common examples of these languages are Web Services Agreement Specification (WS-Agreement) [17], and Web Service Level Agreement (WSLA) [18]. Several negotiation protocols have
been established for defining communication, common terms, actions, and order of events between negotiation participants. Some examples of these protocols are SNAP [19] and WS-Agreement.

Negotiation may take two forms based on the number of issues that it is focused on. One of which is a single-issue negotiation, that focuses on a single issue e.g. price. The other is multiple issue negotiation that focuses on multiple issues, such as price, quantity, and Quality of Service (QoS). [20].

According to the SLA templates of selected providers, through this activity uses some algorithms to rank those providers. Then the broker uses negotiation strategies to make a negotiation with them. At the end of a negotiation, the broker selects the best provider to achieve service level objectives (SLOs). The negotiation focuses on the price and quality metrics in order to classify a service to multiple levels each of which has a specific price.

Generally, a negotiation has several steps [21], including:

1. Negotiation participants specify an offer by defining services' range and some constraints (e.g. time limitations).
2. The provider submits an offer to the broker.
3. The broker analyzes the offer by checking QoSs and price reasonability.
4. The broker compares the offer with other providers' offers.
5. If the broker accepts the offer, the provider allocates services/resources.
6. Negotiation participants sign the SLA.

- Establish SLA: this activity uses the output of previous activity, "Negotiate SLA", to establish SLA between service providers.
and their customers, and sends a copy of the SLA to them.

- **Monitor Service:** Service monitoring begins once a service has been provisioned. Monitoring a service means determining whether Service Level Objectives (SLOs) are achieved or violated. The main objective of this activity is defining violation's type, when occur, and by whom – provider or customer. This activity support parties that a service is being deployed as it was defined in SLA. Based on the established SLA, this activity monitors the progress of service execution, measures the current service's level, and assesses service's QoSs. Hourly, this activity sends the cost of the current hour to *calculate total cost* activity. In case a violation occurs, monitor service activity stores violation data (e.g. id, date, time, type, and cost), subtract the violation cost from the hourly cost, and send the violation id to *send violation information* activity.

- **Send Violation Information:** depends on the violation id, this activity sends violation data to service's parties – provider and customer. Sometimes, according to the violation type renegotiation should be established between the parties.

- **Calculate total Cost:** this activity represents an aggregate summary function, which adds the cost for the last hour to the previous hourly costs.

- **Send total Cost:** at the end of the term of deployment – monthly – this activity sends the total cost for service deployment to the provider as well to the customer. The details of total cost payment are out of the scope of this paper.

**Provider:** It is Cloud Service Provider (CSP) which providing
cloud services to customers through the Broker based on Service Level Agreement (SLA). First of all, several competitive providers publish their service catalog (service info) and their SLA templates at a global directory. When a service requested by the Customer, the Broker uses these catalogs and templates for making a negotiation with those providers. Secondly, the winning provider receives and stores a copy of the SLA, and deploys the requested service. Monthly, the provider receives a message specify the violation type and cost if it occurs, and another message that determines the total cost for service usage during this period.

3.2. Model Mechanism

1. The Service providers publish their services' information with the SLA templates at a public directory.

2. Based on the current requirements, the services' consumers send their requests with specific QoSs to the broker for service deployment.

3. The Broker selects suitable providers and maps the customer's QoSs to corresponding service terms – provider's QoSs – based on service catalog.

4. The Broker uses published SLA templates to start the negotiation – on behalf of the customer – with the providers who can provide the requested service. After selecting the best one of them, the Broker continue the negotiation with him to commit all service levels by defining price and QoSs for each level. At the end of a successful negotiation, each level with its price will be defined.
5. The Broker establishes an SLA between the selected provider and a customer, and sends the copy to them. This SLA defines service's levels parameters in details which include QoSs and focusing on price.

6. The provider starts the service deployment, as well the customer start the service usage.

7. The Broker starts the service monitoring when service is started, and collects service measurement information, and compares it with the agreed service parameters extracted from the SLA. Hourly during the monitoring, if a violation occurs, its cost should be subtracted from the cost of the current hour.

8. In case a violation occurs, the violation information (violation type and cost) will be sent to service's parties.

9. The Broker adds hourly cost for the last hour to the previous hourly costs.

10. Monthly, the Broker sends the total cost to the service provider as well to the customer for billing.

3.3. The Architectural Model

As depicted in Fig3, the architectural view is composed of three cornerstones that are: Customers, Broker, and Providers. We are focusing on the middle, Broker. Following, we are summarizing the sequence of operations at the architecture.

1. The provider publishes their services and SLA templates at a service directory.

2. The customer requests a service from the Broker.

3. The Negotiator (3.1) uses converted customer's QoSs to (3.2)
search for required service from service directory.

4. The negotiation is established between Negotiator and some competitive providers, and only one of them will win the game.

5. The Negotiator sends negotiation result to SLA manager.

6. SLA manager establishes an SLA, and sends it to the participants.

7. The provider deploys the service.

8. The service monitor (8.1) monitoring based on the SLA, (8.2) sends hourly cost to the Cost Manager, and (8.3) sends violation ID – if violation occurs – to the Violation Manager.

9. The Violation Manager (9.1) specify violation cost (penalty) based on the SLA, (9.2) sends violation cost to the Cost Manager, and (9.3) sends violation message to the Provider.

10. Monthly, the Cost Manager sends total cost for the last month to service parties.

Fig1 | SLA-based Dynamic Pricing Model – using BPMN
3.4. The Example

At our country – SUDAN – we have ten universities, suppose those universities need to backup their graduates’ certificates using the Cloud. The service is a "Certificates Backup", the customers are Sudanese universities, and some
companies that can download a certificate to employ the graduates. The customer – universities and companies – requests a service from the Broker after specifies the following parameters:

1. **ResTime:** it is a response time which is maximum time waiting for a result.
2. **Avail:** it is availability, which represents the probability that the service is ready to use.
3. **Req/Res size:** it is the size of input/output data (file) uploaded or downloaded by the customer.
4. **ContLength:** it is the contract length which represents the rental period.

The Broker negotiates with several SaaS providers and selects the best one to deploy the service. If the provider has an infrastructure to run the service, then we have only one SLA between him and the customer. The SLA defines the parameters values and the hourly cost which is affected by ContLength (as shown in Table 1). The SLA also specifies the violation cost – service credit – (as shown in Table 2)

If SaaS provider hasn’t an infrastructure, then he rents infrastructure from IaaS provider; as a result, we have two SLAs one is mentioned and the other is an SLA between the SaaS provider and the IaaS provider. This SLA specifies the IaaS parameters (as shown in Table 3) and IaaS violation cost (as shown in Table 4).

In this case, a SaaS provider represents the customer and may need the Broker to find the best IaaS provider. The customer – SaaS provider – request a service from the Broker based on the following parameters:

1. **ResTime:** it is a service initiation time which represents the time to deploy a VM.
2. **Processing Speed:** it is a time at which the VM can process the request.

3. **Networking Speed:** it is a time at which the I/O data is transferred.

4. **ContLength:** it is the contract length which represents the rental period.

The SaaS provider and IaaS provider provision services at two different packages. Each package has QoS parameters (Table 1 and Table 3) and violation cost (Table 2 and Table 4). H/D, H/W, and H/M represent hour per day, an hour per week, and hour per month respectively.

In case a violation occurs, one of two scenarios should be applied. The first is discount the hourly cost by x% (service credit), and the second is provision x% of service free.

### 4. RESULT AND DISCUSSION

As we mentioned at the previous section, if SaaS provider owned service infrastructure, we have one SLA and three stakeholders which are customer, broker, and SaaS provider. The customer requests a service from broker, which select best provider. The provider deploys services at two different packages (P1 and P2). At package1 (P1) the response time (ResTime) is less than 30 seconds, the availability is greater than 99.9%, the request/response size (Req/Res Size) can be from 10 to 20 GB, and the hourly cost is defined depends on addition factor that is contract length (ContLength). The contract length can be a day; a week, or a month. So the hourly cost per day (H/D) is $40, per week (H/W) is $10, or per month (H/M) is $3. If service provider violates one of service level objective (SLO),
he/she should pay violation cost. For example, if response time greater than 30 or 50 seconds, provider deploy 10% or 25% service credit respectively, as depicted at table 2. At package 2 (P2) the response time (ResTime) is less than 60 seconds, the availability is greater than 98%,

the request/response size (Req/Res Size) can be from 1 to 9 GB, and the hourly cost per day (H/D) is $30, per week (H/W) is $7, or per month (H/M) is $1. The violation cost is similar to violation cost of P1 but with different values (table 2).

<table>
<thead>
<tr>
<th>Packages</th>
<th>ResTime</th>
<th>Service credit</th>
<th>Avail</th>
<th>Service credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>&gt;30 sec</td>
<td>10%</td>
<td>&lt;99.9%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>&gt;50 sec</td>
<td>25%</td>
<td>&lt;99%</td>
<td>25%</td>
</tr>
<tr>
<td>P2</td>
<td>&gt;60 sec</td>
<td>10%</td>
<td>&lt;98%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>&gt;70 sec</td>
<td>25%</td>
<td>&lt;98%</td>
<td>25%</td>
</tr>
</tbody>
</table>

If SaaS provider hasn’t owned service infrastructure, then the model can be applied with three stakeholder customer (SaaS provider), broker, and IaaS provider. In this case the customer...
uses broker to rents service infrastructure from IaaS provider. There are four factors to determine the hourly cost of rented infrastructure which are response time (ResTime), processing speed, network speed, and contract length (ContLength).

IaaS provider deploys service with two different packages (P1 and P2) each of which has different values (as shown in table 3). Similarly to SaaS provider, if IaaS provider violates one of service level objectives, the violation cost will be applied against him (as shown in table 4).

Table 3: some of IaaS parameters

<table>
<thead>
<tr>
<th>Packages</th>
<th>ResTime</th>
<th>Processing Speed</th>
<th>Networking Speed</th>
<th>ContLength/ hourly cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>&lt; 30 sec</td>
<td>&lt; 10 sec</td>
<td>&lt; 20 sec</td>
<td>H/D=$30, H/W=$7, H/M=$1</td>
</tr>
<tr>
<td>P2</td>
<td>&lt; 60 sec</td>
<td>&lt; 20 sec</td>
<td>&lt; 40 sec</td>
<td>H/D=$25, H/W=$4, H/M=$0.5</td>
</tr>
</tbody>
</table>

Table 4: IaaS violations cost

<table>
<thead>
<tr>
<th>Packages</th>
<th>ResTime</th>
<th>Service credit</th>
<th>Processing Speed</th>
<th>Service credit</th>
<th>Networking Speed</th>
<th>Service credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>&gt;30 sec</td>
<td>10%</td>
<td>&gt; 10 sec</td>
<td>10%</td>
<td>&gt; 20 sec</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>&gt;50 sec</td>
<td>25%</td>
<td>&gt; 15 sec</td>
<td>25%</td>
<td>&gt; 25 sec</td>
<td>25%</td>
</tr>
<tr>
<td>P2</td>
<td>&gt;60 sec</td>
<td>10%</td>
<td>&gt; 20 sec</td>
<td>10%</td>
<td>&gt; 40 sec</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>&gt;70 sec</td>
<td>25%</td>
<td>&gt; 30 sec</td>
<td>25%</td>
<td>&gt; 50 sec</td>
<td>25%</td>
</tr>
</tbody>
</table>
5. CONCLUSION AND FUTURE WORK

Cloud services are rapidly gaining significant demand in information technology sectors around the world, due to which, cost management has become a major issue. A single service may be provisioned by several competitive providers in different prices based on its configuration. Cloud brokers work as mediators between different cloud providers and their customers to perform different tasks such as resource allocation, pricing, and billing. Due to diversity in the types of services for providers, and customers, it is necessary to have a dynamic model that handles the negotiation and monitoring activities of cloud providers and their customers. In this paper, we focused on this issue and presented an SLA-based dynamic brokerage model which can help providers attract customers by providing cost-effective services.

As future work, we intend to extend this work by applying it to several case studies and comparing it with relevant previous models.

6. Acknowledgement

This work was made possible by NPRP grant # [7-662-2-247] from Qatar Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

REFERENCES


