CHAPTER 9

SLA PARAMETERS RANKING STRATEGY FOR WEB SERVICE SELECTION

9.1 INTRODUCTION

Service-oriented computing and cloud computing have a reciprocal relationship — one provides the computing of services, and the other provides the services of computing (Yi Wei and Blake 2010). SOC aims to use services as basic blocks to construct rapid, low-cost - yet secure and reliable - applications (Michael Papazoglou et al 2008). A service is different from a traditional software artifact in that it’s autonomous, self-described, reusable, and highly portable (Yi Wei and Blake 2010).

Cloud computing means computing through internet and it is one of the fast growing segments in computing arena today. Cloud computing is changing the computing environment: scalable, virtualized resources are provided as services over the Internet (Hahn-Ming Lee et al 2012). Cloud computing is that realization, as the paradigm facilitates the delivery of computing-on-demand much like other public utilities, such as electricity and gas (Yi Wei and Blake 2010). Clouds also promise to scale by credit card, that is, to scale up at once and provisionally within the limitations imposed by the available monetary resources, as opposed to the physical limitations of adding nodes to clusters or even supercomputers and to the administrative burden of over provisioning resources (losup et al 2011).
Cloud computing has a service-oriented architecture in which services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), which includes equipment such as hardware, storage, servers, and networking components; Platform-as-a-Service (PaaS), which includes hardware and software computing platforms such as virtualized servers, operating systems, and Software-as-a-Service (SaaS), which includes software applications and other hosted services (Furht 2010). Clouds can be a cheap alternative to supercomputers and specialized clusters, a much more reliable platform than grids, and a much more scalable platform than the largest of commodity clusters (losup et al 2011). Hence considering the above mentioned facts of cloud computing, performance becomes a significant factor in cloud computing.

9.2 LITERATURE REVIEW

The issue of Cloud service selection has been discussed in some research papers. Cloud feature models were used as representation mechanism and it was used in cloud service selection process (Erik Wittern et al 2012). QoS ranking procedure was proposed and used for optimal cloud service selection process from a set of functionally equivalent service candidates (Zheng et al 2012). Multi-criteria based cloud service selection was experimented by ur Rehman et al (2011) for better cloud service selection. QoS based service selection was proposed for cloud service composition by creating Web service composition tree (WSCT) (Huihui Bao and Wanchun Dou 2012).

Collaborative filtering approach for predicting QoS values of Web services and making Web service recommendation by taking advantages of past usage experiences of service users (Zibin Zheng et al 2011). A framework for ranking of cloud computing services with a set of metrics was
proposed by Saurabh kumar garg et al (2012). Tree data structure was deployed for indexing in cloud computing (Sundareswaran et al 2012) (Sai Wu et al 2010) (Qi Yu and Bouguettaya 2010). Sundareswaran et al (2012) deployed B+ tree for indexing the information of a large number of cloud service providers and used for cloud service selection.


9.3 BACKGROUND AND MOTIVATION

A cloud computing system consists of a collection of interconnected and virtualized computers dynamically provisioned as one or more unified computing resource(s) through negotiation of service-level agreements (SLAs) between providers and consumers (Rajkumar Buyya et al 2009). Service Level Agreement (SLA) is an official agreement between service provider and service consumer which guarantee the definite level of web service performance based on various quality aspects.

Many cloud service providers are available in market and they provide variety of services with different sets of requirements and so users can choose the one which suits their needs. The real challenge lies in, on one side, user to select the correct service as per their choice and also what is demanded by application. On the other side, cloud service providers to know the preferences of user and provide many services to allow users to have more
freedom to select the cloud service and users also can optimize cloud service selection in terms of effort, time, cost and performance.

Many global software organizations like Microsoft, Amazon, Google etc., ventured into cloud computing business and also succeed in their attempts. For instance, Amazon web services (AWS) provides elastic compute cloud (EC2) as one of its cloud services. It has several options for users to choose among the services as listed below.

M1 Small Instance (Default) 1.7 GiB of memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), 160 GB of local instance storage, 32-bit or 64-bit platform

M1 Medium Instance 3.75 GiB of memory, 2 EC2 Compute Units (1 virtual core with 2 EC2 Compute Units each), 410 GB of local instance storage, 32-bit or 64-bit platform

Similar to this, many software organizations provide cloud services for different utilities including hotel reservation, flight booking etc., and for same utility many cloud service providers are available. The crux of the problem is discussed below. Every cloud service provider publishes what they provide as services and its parameters which concerns resources and non-functional characteristics like scalability, security, response time etc. Gartner (2009) states the need of cloud brokers as “The future of cloud computing will be permeated with the notion of brokers negotiating relationships between providers of cloud services and the service customers”. Today the Cloud customer is facing a challenging problem of selecting the appropriate cloud offers that fits his needs (Foued Jrad et al 2010). Hence real challenges exist in seeking solution for the following issues:
How to make users to express their choices instead of making them to think in terms of how the service is designed by particular or single cloud service provider?

What is a way to recognize users’ preferences in terms of resources and non-functional characteristics?

How to record priority chosen, among all those SLA parameters, by end users?

How the above data can be stored using existing data structures and use it for further analysis?

Many cloud service providers are available and how it can benefit for users to choose the right service from right provider?

In this context, cloud brokers play a vital role between cloud service providers and customers. Generally, two types of brokers in a Cloud market can be distinguished. Firstly, there are brokers that concentrate on negotiating relationships between consumers and providers without owning or managing the Cloud infrastructure. They provide, for example, consultancy services to the potential Cloud customers for moving their IT resources into a suitable Cloud. Secondly, there are brokers that add extra services on top of a Cloud providers’ infrastructure / platform/ software to enhance and secure the Cloud environment for the consumers (Sheikh Mahbub Habib et al 2012). This work contributes to both categories of cloud broker and the significance is in cloud service selection part of cloud broker.
Even if cloud computing is an emerging field, the need to move out from the limitations from a single provider is gaining interest both in academic and commercial research (Antonio Cuomo et al 2012). The challenge in the service discovery phase is to run a query against the cloud services registered in the search engine’s database by matching consumers’ functional, technical, and budgetary requirements (Kwang Mong Sim 2012).

Hence, this work addresses the problem of effective and efficient cloud service selection by applying tree based hierarchical approach to recognize priority and the performance level of several SLA parameters such as resource, availability, scalability, cost, performance etc. demanded by users. One of the valid reasons for this attempt is that some user applications may demand for security as first preference and some other applications may demand for least level security or moderate security or even no security etc., This is applicable for any SLA parameter including resource parameter like CPU, memory etc., and also non-functional parameters. That is, priority given to a SLA parameter differs among applications, users and providers also. This work attempts to expose, store, analyze and use it for user satisfaction and business value for cloud service provider. This will have significant contribution in cloud broker because one of the vital functionalities of cloud broker is finding an ideal match between cloud customer and cloud service provider.

9.4 Proposed Approach

Let us assume a cloud service, CS, for any functionality. Every cloud service has its own resource criterion parameters like CPU, memory etc., and non-functional criterion parameters like security, scalability, performance etc. based on its functionality. Sample SLA parameters list for IaaS is given in Table 9.1. The importance given to these parameters differ and it is purely based on nature of service, application and users etc. This
approach attempts to gather customers’ preferences by making them to specify explicitly.

**Table 9.1 Sample SLA parameters for IaaS (Foued Jrad et al 2012).**

<table>
<thead>
<tr>
<th>Functional</th>
<th>Non – functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU cores</td>
<td>Response time</td>
</tr>
<tr>
<td>Memory size</td>
<td>Budget</td>
</tr>
<tr>
<td>CPU speed</td>
<td>Completion time</td>
</tr>
<tr>
<td>in/out bandwidth</td>
<td>Data transfer time</td>
</tr>
<tr>
<td>OS type</td>
<td>availability</td>
</tr>
<tr>
<td>Storage size</td>
<td>persistence</td>
</tr>
</tbody>
</table>

Let us represent the whole list of parameters for cloud service, CS, as follows:

\[
P(CS) = \{P_1, P_2, P_3, \ldots, P_N\}
\]

\(P_1, P_2, P_3, \ldots, P_N\) represent parameters for cloud service CS. For example, \(P_1\) may denote CPU cores; \(P_2\) may denote response time etc.

Many cloud service providers are available for same cloud service and so each parameter may have different set of values based on options provided by several service providers. For example, sample parameter values are given in Table 9.2. The parameter values are to be expressed in standard terminology and in standard units of measurements in order to compare and evaluate services from several providers. This will help in improving understandability of customers with regard to several parameters of cloud service.
Table 9.2 Sample parameter values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Sample values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU cores</td>
<td>1, 2, 4 etc.,</td>
</tr>
<tr>
<td>Memory size</td>
<td>2Gb, 4 Gb, 8 Gb etc.,</td>
</tr>
<tr>
<td>Response time</td>
<td>26 ms, 30ms, 35 ms etc.,</td>
</tr>
<tr>
<td>In/out bandwidth</td>
<td>First 1 Gb, 100 Gb, 10 Tb etc.,</td>
</tr>
<tr>
<td>availability</td>
<td>100%, 99% etc.,</td>
</tr>
</tbody>
</table>

Each parameter will have its own set of values in different numbers as shown below:

\[ V(P_1) = V_P^{11}, V_P^{12}, \ldots, V_P^{1M_1} \]

\[ V(P_2) = V_P^{21}, V_P^{22}, \ldots, V_P^{2M_2} \]

\[ \ldots \]

\[ V(P_N) = V_P^{31}, V_P^{32}, \ldots, V_P^{3M_N} \]

So, if \( P_1 \) denotes CPU cores, then \( V(P_1) \) values for parameter \( P_1 \) will be as follows.

\[ V_P^{11} = 1, \]

\[ V_P^{12} = 2, \]

\[ V_P^{13} = 4 \text{ and so on.} \]
The number of parameters and values of parameters are completely dependent on nature of service and service provider. In this context, user is allowed to choose the parameters on their own choice as per priority order desired by them. After accepting input for the parameters, the candidate cloud services which suits all parameter options, if not possible, candidate services which suits maximum number of options will be available to the user to take final decision. As user is permitted to express requirements on their own choice and also as per the priority decided by them, a data structure is required to model the users’ needs. Tree data structure is better suitable for modeling hierarchical data and it is deployed here as follows:

Number of SLA parameters is denoted as pn and number of possible ranking structures is denoted as NPRS.

Let us take number of SLA parameters, pn = 2, then the two possibilities of ranking SLA parameters for user will be as shown in Figure 9.1.

![Figure 9.1 Ranking parameter tree for pn=2](image)

**Figure 9.1** Ranking parameter tree for pn=2

Ranking structures are:

P₁ – P₂

P₂ – P₁
Number of possible ranking structures (NPRS) = 2

Let us take $p_n=3$, then the possibilities of ranking parameters for user will be as shown in Figure 9.2, Figure 9.3, Figure 9.4.

Figure 9.2  Ranking parameter tree for $p_n=3$ with root node $P_1$

Figure 9.3  Ranking parameters tree for $p_n=3$ with root node $P_2$
Figure 9.4  Ranking parameter tree for pn=3 with root node $P_3$

Hence, if pn=3 then,

Possible ranking structures are:

$P_1 - P_2 - P_3$

$P_1 - P_3 - P_2$

$P_2 - P_3 - P_1$

$P_2 - P_1 - P_3$

$P_3 - P_1 - P_2$

$P_3 - P_2 - P_1$

Number of possible ranking structures (NPRS) = 6

For pn=4, possibilities of ranking parameters with root node $P_1$ will be as shown in Figure 9.5.
Similar to the above, the structure is repeated with $pn=4$ and root node $P_2$, $P_3$ and $P_4$ and it shown in Figure 9.6, Figure 9.7 and Figure 9.8.
Figure 9.7  Ranking parameters for pn=4 and root node $P_3$

Figure 9.8  Ranking parameters for pn=4 and root node $P_4$
As shown in fig. 9.5, Ranking parameters tree for n=4 and root node P_1 has six different parameter hierarchies. Similar to this, pn=4 and root node P_2, P_3 and P_4, each has six parameter ranking hierarchies as shown in Figure 9.6, Figure 9.7 and Figure 9.8 respectively.

So, for pn=4, NPRS=24

Similarly, for pn=5, NPRS=120

By applying recursive algorithm as shown below, we can estimate NPRS for n parameters.

\[
\text{NPRS}(n) = \begin{cases} 
1 & \text{if } n < 2 \\
2 & \text{if } n = 2 \\
\text{n} \times \text{NPRS}(n-1) & \text{else} 
\end{cases}
\]

This tree structure will recognize the priority among the SLA parameters for particular cloud service considering multiple criteria from multiple vendors.
9.5 SPARSE MATRIX FOR STORING PARAMETER RANKING

As discussed above, the tree data structure is used to recognize the parameter priority. The actual values for parameters may be stored separately in databases or in any other file format. The pattern of SLA parameter hierarchy is stored in the form of sparse matrices. In the mathematical subfield of numerical analysis, a sparse matrix is a matrix populated primarily with zeros (Stoer and Bulirsch 2002).

Let us assign $m = \max(m_1, m_2, \ldots, m_n)$

$m_1, m_2, \ldots, m_n$ are subscripts used in representation of maximum number in values of parameters for each parameter.

The size of the sparse matrix will be in the order of $(m+1)\times(m+1)$. The first column is reserved for storing parameter number except last row in first column which is used for storing count for that type of matrix. The elements will be zeroes in all rows and columns except first column and in places where the options chosen by the user, is stored with one’s.

For example, if a user has chosen $P_3$ as first priority and fourth option for $P_3$, then first row and first column is filled with 3 and fifth column in the first row is filled in with number 1. That is, it is to mean that fourth option (column value -1) has been chosen for parameter $P_3$ and $P_3$ is given first priority.

Similarly, if user has chosen $P_5$ as second priority and third option for $P_5$, then second row and first column is filled with 5 and fourth column in the second row is filled in with number 1. In the last row, except first column, all other columns in last row are reserved for future use for storing
some other useful information. The first column in the last row is reserved for storing count value for the particular type of matrix. The sample matrix will be as shown in Figure 9.9.

![Sample sparse matrix](image)

Figure 9.9 Sample sparse matrix

Total number of sparse matrices required to cover all the choices and patterns is determined as follows.

Number of sparse matrices required = NPRS * (product of number of choices in each parameter)

= NPRS * (m₁ * m₂ *…mₙ)

So if there are two parameters and two choices in each parameter, then

NPRS = 2

Number of parameters = 2
Number of choices in each parameter = 2

That is, \( m_1 = 2 \) and \( m_2 = 2 \)

Number of sparse matrices = \( 2 \times 2 \times 2 = 8 \)

Size of sparse matrix = 3*3

If a pattern appears first time, sparse matrix is created for the pattern, and the same pattern occurs repeatedly, then count value of the corresponding sparse matrix is incremented by one. The number of sparse matrices will be same irrespective of any number of users for particular pattern. So memory space is saved due to the fact that repetitive patterns are stored by just incrementing the count value stored in corresponding matrix.

The skipped entries by user may be kept zeroes in that particular row which indicates that no preference given by customer and any value or default value may be taken for the parameter. The usage of sparse matrix helps in saving memory space and analyzing patterns in user needs.

### 9.6 PERFORMANCE METRICS

Software performance is one of the non-functional requirements of software and becoming a vital concern for both users and developers. Software Performance Engineering (SPE) is a systematic, quantitative approach to cost-effectively constructing software systems that meet performance requirements (Smith and Williams 2002). Software metrics helps to understand, control and improve various aspects involved in software development. The metrics proposed in this work helps in improving performance of cloud service selection in cloud broker which in turn helps in meeting user performance requirements of software. A set of metrics is proposed and usage of the metrics also discussed below.
Number of times the parameter in highest priority (NOPHP)

This can be calculated from the sparse matrix, that is, number of times the SLA parameter number appears in first row and first column of the matrix. The use of this metric is that cloud service users and cloud service providers can understand number of times this parameter has been given highest importance.

Number of times the parameter in least priority (NOPLP)

This can be calculated from the sparse matrix, that is, number of times the SLA parameter number appears in last but one row and first column of the matrix. The use of this metric is that cloud service users and cloud service providers can understand number of times this parameter has been given least importance.

Priority of parameter (POP)

This can be calculated as maximum number of times the parameter number appears in first column of particular row of sparse matrices. The use of this metric is to know general priority of this parameter.

Parameters Ranking Path sequence repetition (PRPSR)

This metrics attempts to capture repetitions in SLA parameters ranking and can be calculated in two aspects:

(i) Strict equality between sparse matrices
(ii) Slack equality between sparse matrices

- Strict equality between sparse matrices

In strict equality between the patterns, it can be retrieved from count value stored in each sparse matrix and the count value is updated every time a sparse matrix is created, using the following algorithm.

- Algorithm for strict equality

This algorithm checks for equality for all non-zero elements of matrices except last row. If there is match between two matrices, then old count stored in existing matrix is incremented by one and parameter values are stored separately. If no matrix matches with the new pattern then new sparse matrix is created for that particular pattern. The algorithm will be as shown below.

Repeat for all matrices

{  
Match=0

For i=1 to m do

For j=1 to m do

{  
If(a(i,j)!=0)
If\(a(i,j)\neq b(i,j)\)

Exit and check with next matrix

}\}

Count=\text{count+1}

Match=1

Exit from loop

}\}

If( match==0) then create a new sparse matrix

- Slack equality between sparse matrices

In this case, parameter number equality in the first column in both matrices, that is, parameter \(P_i\) appears in particular row is considered equal to another sparse matrix with \(P_i\) appears in the same row but the choice value may be different. That is, checking only for parameter sequence not the choice values. This helps in finding frequent path sequence desired by users. In case of (i), strict equality of the pattern, it is enough to refer the count value available in that sparse matrix. In case of (ii), as discussed above,
checking only for equality of an element in first column of matrix is to be done.

In case of checking only for parameter sequence, then it can be retrieved by checking for equality in the sequence available in first column except the last row element which is reserved for count.

- Algorithm for checking only for parameter sequence

Repeat for all matrices

{  
for i= 1 to m do  
{  
If(a(1,i)!=b(1,i))  
Exit and Check with next matrix  
}  
Count=count+1  
}

The use of this information to know the repetitions took place in the pattern of customers’ requirements. This information is very much useful for cloud service providers so that to increase services in highly repeated set of requirements and allocate more resources according to the frequency of repetitions.
Statistics associated with parameter selection at each stage

The statistics information associated with parameter value at each stage calculated from past experiences will greatly help new users in decision-making.

At the first step of selection of parameter, for each parameter, percentage value associated with that parameter may be given as information to user.

Percentage of parameter \( P_i \) occurs in first priority

This is calculated as percentage value and as shown below.

\[
\frac{\text{NOPHP for } P_i}{\text{total number of matrices}} \times 100
\]

Percentage of parameter \( P_j \) occurs in level \( n \) and \( P_i \) occurs in level \( n-1 \)

From the second step onwards, the percentage calculation may be applied for selecting \( P_j \) at level \( n \) from the parameter \( P_i \) at level \( n-1 \).

\[
\frac{\text{Number of times } P_i \text{ at level } n-1 \text{ and } P_j \text{ selected after } P_i}{\text{Number of times } P_i \text{ at level } n-1}) \times 100
\]

These statistics information may be displayed with each parameter at each stage which will help users to make better decisions in selecting next parameter. They enable cloud service providers to focus on adding more and
more differentiated services with different combination of qualities desired by customers.

➤ Significance of the metrics proposed

All the above metrics provide a way for cloud service providers to understand priority of requirements of users.

The cloud service providers can use this information to design new cloud services.

The metrics data may be further categorized by users, application, demography and domains etc., and this will help in developing customized solutions by focussing on particular group of users or field of application. New users may utilize this information to take better decision in initial stages.

Cloud brokers will be able to make best effort to find a match between customer requirements and cloud service. They can use past metrics data to allocate resources in such a way that cloud service fulfils and match with customers’ expectations in various parameters of cloud service.

9.7 CLOUD SERVICE SELECTION

Sometimes existing cloud services may not absolutely match user requirements. If a cloud service from any cloud service provider matches completely then it is included in candidate service list. Otherwise, list of services which matches to a certain extent will be included in the candidate list. In initial stages, possible solution may be several instances of existing service will serve the purpose to certain extent. For example, two instances Amazon web service M1 and three instances of Amazon web service M2 etc.
But new cloud services may be designed to suit majority of customer needs by understanding the user’s dynamic desires and needs and also as per user’s application demand for resources and non-functional characteristics. If cloud service metadata information that is complete data about cloud services from several providers are organized with standard ontology and also up-to-date, then software may be designed to process and suggest solutions for customer needs. Design of proper front-end software systems is very much required to gather user requirements. Front end system doesn’t have to be designed with any restriction due to this data structure, and it is to be very much user friendly. The mapping and processing of data according to data structure may happen in background.

### 9.8 SUMMARY

This solution is efficient in terms of providing solutions for the questions raised in this work.

- User can choose parameter value already decided by cloud service providers but not from single cloud service provider.
- User can express the requirements on their own preferences
- Priority is modelled through tree data structure
- Tree data structure and sparse matrix is used and it helps in analysing requirements and designing new services.
- Cloud service provider may also be included as one of the parameters and so all information is available under single system.
In a nutshell, this provides an efficient system for cloud service selection from several cloud service providers and complete automation of this through software will make storing, processing and analysing fast and easy.

In service level agreement parameters ranking strategy, the technique is applicable for any web service irrespective of implementation because the technique deals only with Quality of Service (QoS) parameters of web service. This technique provides an efficient mechanism to understand and record the user preferences in selecting a web service in cloud broker. The metrics data associated with this technique helps cloud brokers to satisfy customers with existing web services from several web service providers and developers to design new web services to suit the needs of customers. This is applicable for any kind of web service due to the fact that it deals only with service level agreement parameters.