2014

**URMG: Enhanced CBMG-Based Method for Automatically Testing Web Applications in the Cloud**

Xiaolin Xu  
*Services Computing Technology and System Lab and Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Hai Jin  
*Services Computing Technology and System Lab and Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Song Wu  
*Services Computing Technology and System Lab and Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Lixiang Tang  
*Services Computing Technology and System Lab and Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Yihong Wang  
*Services Computing Technology and System Lab and Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Follow this and additional works at: [https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology](https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology)

**Recommended Citation**


This Research Article is brought to you for free and open access by Tsinghua University Press: Journals Publishing. It has been accepted for inclusion in Tsinghua Science and Technology by an authorized editor of Tsinghua University Press: Journals Publishing.
**URMG: Enhanced CBMG-Based Method for Automatically Testing Web Applications in the Cloud**

Xiaolin Xu, Hai Jin, Song Wu, Lixiang Tang, and Yihong Wang

**Abstract:** To satisfy the rapid growth of cloud technologies, a large number of web applications have been developed and deployed, and these applications are being run in clouds. Due to the scalability provided by clouds, a single web application may be concurrently visited by several millions or billions of users. Thus, the testing and performance evaluations of these applications are increasingly important. User model based evaluations can significantly reduce the manual work required, and can enable us to determine the performance of applications under real runtime environments. Hence, it has become one of the most popular evaluation methods in both industry and academia. Significant efforts have focused on building different kinds of models using mining web access logs, such as Markov models and Customer Behavior Model Graph (CBMG). This paper proposes a new kind of model, named the User Representation Model Graph (URMG), which is built based on CBMG. It uses an algorithm to refine CBMG and optimizes the evaluations execution process. Based on this model, an automatic testing and evaluation system for web applications is designed, implemented, and deployed in our test cloud, which is able to execute all of the analysis and testing operations using only web access logs. In our system, the error rate caused by random access to applications in the execution phase is also reduced, and the results show that the error rate of the evaluation that depends on URMG is 50% less than that which depends on CBMG.

**Key words:** cloud; web application; performance evaluation; customer behavior; user representation

1 **Introduction**

Nowadays, web applications are becoming increasingly popular in daily life, and because of the rapid development of cloud technologies, these applications can support concurrent usage by several millions or billions of users. To improve the user experience, developers are spending more time seeking to conduct testing and to improve the performance of the applications. Many kinds of performance evaluation methods have been proposed, such as script-based evaluation, website model-based evaluation, and user model-based evaluation. The script-based evaluation is performed by executing a group of scripts that are created by developers. The website model based evaluation builds a model of the website. After analyzing the structure and the process flow of the website, the system can automatically create a group of threads to access the website. The user model based evaluation builds user models after mining the website access logs, and then uses the models to create some virtual users that are represented by web access scripts to simulate visits by real users. In addition to being less costly and time consuming, the model-based evaluation also shows the performance of applications in real runtime environments more accurately, so it has...
been more frequently discussed and studied compared to other evaluation methods.

The key phase of the user model based evaluation is the generation of models from web data such as access logs\(^{[2,3]}\). Recently, the Customer Behavior Model Graph (CBMG) has been one of the most popular models. Each CBMG describes the behavior characteristics of one kind of user of the application. A status is a group of user operations that are similar to each other in some respect. CBMG-based evaluations can generate virtual users to simulate real users visits to the web applications with the aim of accurately reproducing the performance of the tested object. However, there are some shortcomings to the CBMG-based evaluation.

After building the CBMGs, systems begin to generate virtual users, which is a group of status sequences represented by access operations to the application. Usually, the access is chosen from the access records of the web log. Because the records in the same status are equivalent to each other, a random selection method is generally used in this case. However, there are some topology relationships between the access records within the same session, which will be destroyed by this random selection method. In addition, this will produce some access errors during the evaluation execution phase. These errors may affect the normal running of the application, which will eventually reduce the accuracy of the evaluation result. To solve this problem, we propose a new model called the User Representation Model Graph (URMG). The URMG is built based on CBMG, which helps to optimize the evaluation execution. Based on URMG, we designed and implemented an automatic testing and evaluation system for web applications, which can execute the access log analysis, GBMG and URMG construction, virtual user creation, evaluation, and data management without the need for any manual operation.

2 Background

To facilitate the introduction to URMG below, this section will first give a brief overview of CBMG. Each CBMG is represented by two matrices, which are the status-transfer-probability matrix and user-thinking-time matrix. The former represents the transfer probability between any two statuses, while the latter represents the transfer delay between any two statuses.

A typical CBMG is as shown in Fig. 1. Each node in the graph represents a status, and each directed edge represents a transfer relationship. The status connected with the head is the target status, and the status connected with the tail is the source status. Every two statuses that are connected with an edge have a transfer relationship, some of which are unidirectional transfer (such as status 3 and status 4, and status 3 and status 2), while some are bidirectional transfer (such as status 1 and status 3, and status 1 and status 2). There are two kinds of special statuses in every CBMG. These are the entrance status and exit status. The entrance status connects only with the tails of edges, which means that it can only be the source status. The exit status connects only with the heads of edges, which means that it can only be the target status. For example, status 0 is the entrance status and status 4 is the exit status. In CBMG, there is usually one entrance status and one exit status. Each access sequence starts from the entrance status and ends at the exit status.

The two critical matrixes are also described in Fig. 1 by the number pair above the edges. The first number is the value of the status-transfer-probability matrix, and represents the probability that the source status will transfer into the target status, while the second number is the value of the user-thinking-time matrix, which indicates the transfer delay time needed by the source status to transfer to the target status. Let us use \( M_t \) to denote the transfer probability matrix and \( M_u \) to denote the user-thinking-time matrix. The value of \( M_t[i][j] \) therefore represents the probability of status \( i \) transferring into status \( j \), and the value of \( M_u[i][j] \) represents the delay time of status \( i \) transferring into...
status $j$. For example, in Fig. 1, the value of $M_1[3][2]$ and $M_3[3][2]$ are 0.375 and 11.416667, respectively, while the value of $M_1[2][4]$ and $M_3[2][4]$ are both zero.

The most popular method used to build CBMGs from sessions is clustering\cite{4,6}, which uses specific algorithms to determine the common characteristic among sessions, and groups the sessions according to their similarities with each other. The similarity is measured by a given formula. According to the formula, sessions within the same cluster are much closer to each other than to sessions in other clusters. The $k$-means clustering algorithm is a widely used clustering algorithm\cite{7}, which considers the sessions as points. It uses a randomly-produced integer $k$ to represent the number of clusters, and it then randomly selects $k$ points as the initial cluster center. Then, it appends all of the other points to the cluster, which has the center closest to the point in terms of the Euclidean distance\cite{8,9}.

The biggest difference between CBMG and URMG is that there are no statuses in URMG; rather, it has specific access records. However, the status-transfer-probability matrix and user-thinking-time matrix are still the same as with the original CBMG, which ensures that the user behavior characteristics are not affected.

3 User Representation Model Graph

This section describes in detail the method used to transfer CBMG into URMG. Firstly, we propose three rules to identify the particular session used to replace the statuses of CBMG. The rules are integrity, relevance similarity, and topology similarity. Then, we give the mathematical formula and algorithm of the URMG generation.

3.1 Integrity

Integrity refers to the status integrity. Because we will use the access records of one session to replace the status of CBMG, the session has to contain all of the statuses. In other words, all of the statuses of CBMG have at least one access record from the same session. However, it is sometimes difficult to determine this kind of session. Thus, we are required to select the best method that measures the integrity of every session.

Denoting CBMG by $C$, sessions by $S_i$ ($i = 0, 1, \cdots, M, M$ is the number of sessions), statuses by $C_j$ ($j = 0, 1, \cdots, N, N$ is the number of statuses that $C$ has), and access records by $S_{ik}$ ($k = 0, 1, \cdots, K, K$ is the number of access records that $S_i$ has), we can then get Eqs. (1) and (2) below:

$$S_i = (S_{i0}, S_{i1}, \cdots, S_{iK}) \quad (1)$$
$$C_j = (S_{0j}, S_{1j}, \cdots, S_{Kj}) \quad (2)$$

According to Eq. (1), $S_i$ is an ordered sequence of access records with an angle bracket, while Eq. (2) shows that $C_j$ is a set of access records with parentheses.

We use an integrity factor $\Omega_i$ to represent the integrity of session $i$, which is defined by Eq. (3):

$$\Omega_i = \frac{\sum_{j=0}^{N} \overline{w}_j}{N} \quad (3)$$

Factor $\overline{w}_j$ indicates whether the status $j$ contains the access record from the session. When status $C_j$ has an access record from the session, $\overline{w}_j$ is equal to 1; otherwise, it is equal to 0. Because the numerator is not larger than the denominator, which is the total number of statuses, $\Omega_i$ will always be less than or equal to one.

3.2 Relevance similarity

The relevance similarity considers the difference between the status-transfer-probability matrix of CBMG, $MT_C$, and the status-transfer-probability matrices of sessions, $MT_S$. In other words, it considers the difference between CBMG and sessions of the same cluster. In order to reduce the possible loss incurred by the status replacement, we should choose access records from the session that are closest to CBMG. The similarity is defined by the Euclidean distance $\Psi$ between these two matrices as given below:

$$\Psi = \sqrt{\sum_{i=0}^{N} \sum_{j=0}^{N} (MT_C[i][j] - MT_S[i][j])^2} \quad (4)$$

3.3 Topology similarity

3.3.1 Definition

The topology similarity considers the topological relationship between statuses within a session. The topological relationships represent the dependencies that exist between statuses. For example, the access operation of some statuses has to be started after the operation of some other statuses, and before some other operations. The topology similarity analysis is different from the relevance similarity in that it uses a sequence to analyze matrices other than the transfer matrices.

In order to analyze the user behavior on a global level, CBMG tries to build relationships between any two statuses. However, although this kind of model may properly illustrate the transformational relationships between statuses, it also destroys the topological
relationship of statuses, which leads to errors in the evaluation process. This kind of relationship can only be found in the session since the session is an ordered sequence of access records (or statuses). In order to restore the topological relationship, we have to choose a suitable session that can describe the greatest number of topological characteristics of all of the sessions.

### 3.3.2 Longest common subsequence

Because the session is an access sequence, we propose a method based on the Longest Common Subsequence (LCS) to measure the topology similarity between sessions\(^2\). The LCS of a session is the longest common status subsequence of two sessions. For convenience, we use the length of a session to represent the number of access records.

The LCS has a well-studied optimal sub-structure property that is given by Theorem 1.

**Theorem 1** For sequence \( \alpha = (a_0, a_1, \cdots, a_n) \) and \( \beta = (b_0, b_1, \cdots, b_m) \), the LCS \( \gamma = (c_0, c_1, \cdots, c_k) \) has the property given as: First, if \( a_n = b_m \), then \( c_k = a_n = b_m \), and \( \gamma_{k-1} \) is the LCS of \( a_{n-1} \) and \( b_{m-1} \). Second, if \( a_n \neq b_m \), then \( a_n \neq c_k \) implies that \( \gamma_k \) is the LCS of \( a_n \) and \( \beta \), and \( b_m \neq c_k \) implies that \( \gamma_k \) is the LCS of \( \alpha \) and \( \beta_{m-1} \).

Based on Theorem 1, we can calculate the LCS of session \( S_m = (S_{m0}, S_{m1}, \cdots, S_{mK_m}) \) and \( S_n = (S_{n0}, S_{n1}, \cdots, S_{nK_n}) \) by the following steps.

**Step 1** Use the status to replace all of the access records of the session string. Then, we can get the session sequence, e.g., \( S_m = (C_0, C_2, \cdots, C_N) \) and \( S_n = (C_0, C_1, \cdots, C_N) \).

**Step 2** Create a matrix \( M \) with \( i \) rows and \( j \) columns, \( M[i][j] \) is assigned to 1 if \( S_m[i] \) is equal to \( S_n[j] \); otherwise, it is assigned to 0.

**Step 3** Create another matrix \( F \) with \( i \) rows and \( j \) columns; the value of \( F[i][j] \) is defined by Eqs. (5) and (6) as follows:

\[
F[0][0] = M[0][0] 
\]

\[
F[i][j] = \max(F[i][j-1] + M[i][j], F[i-1][j]) 
\]

The value of matrix \( F \) will be calculated by circular computations with matrix \( M \).

**Step 4** Backtracking the matrix \( F \) from the \((0, 0)\) position, we can get the final LCS.

Because the length of the LCS affects the topology similarity between two sessions (as opposed to the content), the fourth step is unnecessary, so we will skip it and stop after Step 3.

### 3.3.3 Algorithm implementation

For each session \( i \), its topological relationship with all of the other sessions is represented by a factor \( \Phi_i \), which is defined by Eq. (7).

\[
\Phi_i = \frac{\sum_{j=0}^{M} \varphi_{ij}}{M - 1}, \quad j \neq i 
\]

Factor \( \varphi_{ij} \) represents the topological similarity between session \( i \) and session \( j \), and is defined by Eq. (8):

\[
\varphi_{ij} = \frac{L_{ij}}{K} \tag{8} 
\]

Factor \( L_{ij} \) is the length of the LCS of session \( i \) and session \( j \). \( K \) is the length of session \( i \). Because the length of the sessions may have a significant difference, it is not proper to use the length of the LCS to represent the similarity of two sessions. We therefore consider the length of the session itself, and use it to divide the length of LCS. In order to calculate the factor \( \Phi_i \), we need to transfer the sessions into sequences of statuses, calculate the length of the LCS of session \( i \) with all of the other sessions \( L_{ij} \), and the similarity of session \( i \) and all of the other sessions \( \varphi_{ij} \). Finally, we need to calculate the sum of all of the factors \( \varphi_{ij} \), and use it to calculate the topological similarity \( \Phi_i \).

### 3.4 Model definition and construction

URMG uses the access records of the same session to replace the statuses in CBMG. In this way, the evaluation case generation becomes much easier and the evaluation also becomes more accurate because the topology relationships are recovered in the URMGs. The formal definition is as follows.

**Definition 1** The user representation model graph is an instantiation model of the customer behavior model graph that appends the topology relationships to the model while maintaining the transfer relationships between statuses.

The key point is to find the proper session, which is defined as the representation session. We propose three rules to identify the optimum session. First, the representation session must contain as many statuses of CBMG as possible. This rule means that the \( \Omega \) factor of the session must be as large as possible. Second, the representation session must be as close to CBMG as possible. This rule requires that the \( \Psi \) factor of the session and CBMG must be as small as possible. Third, the representation session must be sufficiently proper to stand for other sessions topological relationships. This rule implies that the factor \( \Phi \) has to be as large as possible.
According to the clarification above, we propose a factor $\Theta$ to represent the conformity with the three rules. It is defined by Eq. (9):

$$\Theta_i = \frac{\Omega_i^2 \cdot \Phi_i}{\Psi_i}$$  \tag{9}

According to Eq. (9), the larger the $\Omega$ and the $\Phi$ are, the larger will be $\Theta$, while a larger $\Psi$ results in a smaller $\Theta$. Therefore, as $\Theta$ for the session increases, it becomes more suitable for it to be used to build URMG. In order to find the representation session, we therefore have to calculate the $\Theta$ of all the sessions, and select the session with the largest $\Theta$.

The entire URMG construction process is described in detail below.

**Step 1** For each session of the same CBMG, build the status-transfer-probability matrix $MT_i$ and status sequence $SS_i$.

**Step 2** Use $MT_i$ to calculate $\Omega_i$ with Eq. (3), and store the values in an array $W = [\Omega_0, \Omega_1, \ldots, \Omega_M]$.

**Step 3** Use $MT_i$ and CBMG to calculate $\Psi_i$ with Eq. (4), and store the values in an array $Q = [\Psi_0, \Psi_1, \ldots, \Psi_M]$.

**Step 4** Use $SS_i$ to calculate $\Phi_i$ with Eq. (7), and store the values in an array $X = [\Phi_0, \Phi_1, \ldots, \Phi_M]$ as well.

**Step 5** Use $W$, $Q$, and $X$ to calculate $\Theta$ of each session with Eq. (9), and find the largest $\Theta_i$ that is denoted by $S'$, while the related session is denoted by $S$.

**Step 6** Repeatedly select an access record from $S'$ to replace the status to which it belongs until there are no statuses left in CBMG.

After these six steps, CBMG will be transferred into URMG, which contains no status. However, it still retains the user behavior characteristics, namely, the status-transfer-probability matrix and user-thinking-time matrix.

4 Design and Implementation

In this section, we present a detailed design of the automatic performance evaluation system for web applications based on URMG, which can automatically analyze web logs, generate models, and execute evaluations. This will help the developers to conveniently and accurately identify the performance of applications.

4.1 System architecture

The system architecture is shown in Fig. 2. There are three modules: user interface, model construction, and evaluation execution. The user interface module provides a graphical web interface for users to submit evaluation requests and download evaluation results. The model construction module uses the log data to build CBMGs and URMGs. The evaluation execution module uses the URMGs to create virtual users and evaluation cases to evaluate the performance of applications.

4.1.1 User interface

The user interface module provides a friendly graphical interface for users. There are three sub-modules in this module: web log upload, evaluation configuration, and result download.

The web log upload module allows users to upload the log of the web application through the web interface. When a user submits a log file from the browser, it will store it in a database, where it can be found by the other modules.

The evaluation configuration module allows users to configure the operational parameters, which includes both necessary and unnecessary parameters. The necessary parameters include the URL of the application and the evaluation execution time. The unnecessary parameters include the evaluation mode and the form in which the evaluation results are presented. After receiving these parameters, the module will generate a configuration file which will be used during the evaluation execution phase.

The resulting download module allows users to check the evaluation status and download evaluation results from the web interface.
4.1.2 Model generation
The model generation module is the kernel module of the system, which builds models to create virtual users based on the web logs. It is made up of three sub-modules: web log handler, CBMG construction, and URMG construction.

The web log handler module analyzes the logs and then generates a group of sessions. After all the operations are finished, it passes the sessions to the CBMG construction module.

The CBMG construction module creates the status-transfer-probability matrix and user-thinking-time matrix for each session. It then uses a $k$-means algorithm to build CBMGs and delivers them to the URMG construction module.

The URMG construction module creates URMGs using the CBMGs obtained by the algorithm described above, and then transmits them to the evaluation execution module.

4.1.3 Evaluation execution
The evaluation execution module manages the evaluation execution and result collection and processing. It also includes three sub-modules: virtual user generation, evaluation execution, and result management.

The virtual user generation module uses the URMGs to create virtual users, which are a group of scripts used to simulate real user visits. Then, it passes the scripts to the evaluation execution module.

The evaluation execution module uses the scripts and the configuration file to execute the evaluation. It will create several threads to run each single script. During the evaluation, it will collect the outputs and store them in a file where they can be found by the result management module.

The result management module automatically performs a statistical analysis on the evaluation results. It then generates a performance evaluation report with tables and charts, compiles all of the results, and sends them back to the users.

4.2 Process flow
As shown in Fig. 3, the process flow diagram starts with the user submitting the application logs and configuration info, and ends with the return of the evaluation report to the user. First, the system will check the validity of the logs and configuration, and if any of them are invalid, the evaluation will be terminated. Then, the system will build CBMGs and URMGs using the log data. Next, the system creates some virtual users based on the URMGs and configuration file to perform the evaluation. Finally, it generates the evaluation report and sends it back to the user.

4.3 System highlights
4.3.1 Automatic evaluation implementation
The entire performance process is automatically implemented without any manual operation. Users only need to upload a web log and wait for the evaluation report. Compared to other conventional evaluation methods, it significantly reduces the workload of the evaluation requesters, as they do not have to design evaluation plans, execute evaluation cases, or analyze the evaluation results, all of which are time-consuming and burdensome.

4.3.2 Accurate evaluation result
The system creates a group of virtual users based on the models of realistic users who may utilize the web application. In this way, the access load in the real runtime environment can be reproduced, making the evaluation report more accurate and reliable.

5 Case Study and Performance Evaluation
In this section, we use an e-commerce web application to evaluate the function of the automatic performance evaluation system. For the performance evaluation, we perform several evaluations with different numbers of virtual users based on CBMG or URMG, and record the error messages received during the evaluation.

Fig. 3 Automatic process flow diagram.
execution phase. Then, we compare the error rates of the evaluation based on CBMG and URMG.

5.1 Experimental setup

We conducted our case study on an Inspur server that was configured with a 12-core CPU, 16 GB of memory, and a CentOS 5.5 Linux 2.6.32 operating system. The web application under evaluation was deployed in a VM-based environment using KVM with a Linux 2.6.18 kernel. The web server used in the application runtime environment was Apache 2.2.3.

5.2 Case study

To perform the case study, we used an e-commerce website that was deployed in a cloud environment. The valid user operation of the website includes:

View: The user can check all of the valid goods on the home page.

Order: The user can place any goods in an order list if he wants to buy it.

Buy: The user can buy any goods on the order list. When the user tries to buy any goods that were not ordered, the application produces an error message.

Comment: The user can comment on any goods that were bought. This operation will produce an error message if he did not buy the goods.

Delete: The user can delete any goods on the order list.

After submitting the website access log and evaluation configuration parameters, the system returns the evaluation report after processing and evaluating for 10 min. The report includes three CBMGs, three representation sessions, three URMGs, and two charts showing the website throughput and response time during the evaluation.

One of the CBMGs is shown in Fig. 4. There are seven statuses represented by digital numbers in the CBMG, while status 0 and status 7 are added by the system to represent the entrance status and exit status. These two statuses have no access operations. The status-transfer-probability matrix and user-thinking-time matrix are also illustrated in the figure.

The representation session used to process the CBMG in Fig. 4 is shown in Table 1, and the related URMG is shown in Fig. 5. In order to draw the picture of URMG, we use symbols to represent the access records, as shown in Table 1. The only difference observed between Figs. 4 and 5 is the nodes. The statuses in the CBMG are all replaced by the access records in the representation session, while the status transfer probability and user thinking time remain the same as the original values, which means that the transfer relationships between the statuses are all maintained.

According to the final evaluation report, eighty virtual users visit the web application at the same time during this 10 min performance evaluation period. The
total number of requests was 129 475 and the average response time was 0.286 s. The detailed throughput during the evaluation is as shown in Fig. 6.

5.3 Error rate evaluation

The main enhancement of the performance evaluation system based on URMG is the reduction of the error rate during the evaluation execution phase compared to the evaluation system based on CBMG. In this section, we therefore compare the error rate of the evaluation on the e-commerce application based on CBMG and URMG.

In order to measure the error rate, we append an error log to the tested application. Every time the virtual user performs an invalid operation on the application, the system will write an error message to the error log. For comparison, we perform a group of evaluations with different numbers of virtual users. During the evaluation, we set the user thinking time to zero for convenience.

Figure 7 shows the total number of accesses for each evaluation case. As observed in the picture, first, as the number of virtual users increases, the number of requests also increases. However, the growth rate decreases as the number of virtual users continues to increase. This is because the increased number of parallel requests causes the response time be much greater, which subsequently decreases the number of requests that can be performed by each virtual during the evaluation. The total number of requests therefore increases much more slowly when the number of virtual users increases.

Figure 8 shows the total number of error messages received for each evaluation case. As we can see, the number of error messages increases as the number of virtual users increases.

Figure 9 shows the error rate for the evaluations based on CBMG and URMG. According to the graph, when there is only one virtual user, both of the evaluations have very low error rates. The error rate of the URMG-based evaluation is 0.022, while that of the CBMG-based evaluation is 0.176. As the number of virtual users grows, the error rates also increase. However, when there are more than 20 virtual users, both of the error rates fluctuate over an interval. The error rate of the URMG-based evaluation remains below the error rate of the CBMG-based evaluation in all of the evaluation cases. The largest difference is observed to be 0.15 and the average is 0.1, which means that the error rate of the URMG-based evaluation system is 10% less than that of the CBMG-based evaluation system. In other words, the error rate is reduced by an average of 50% relative to the CBMG-based evaluation.

6 Related Works

In recent years, a lot of work has been done to improve the performances of web applications. Some of
them have concentrated on realizing a more advanced structure for the web application. Meanwhile, others have aimed to build more accurate user models of the applications for analysis and evaluation, and to eventually improve the performance by solving the performance bottleneck problem.

Lu and Yeung proposed a framework that will increase the effectiveness of the commercial web applications\textsuperscript{[10]}. They designed a group of rules to facilitate the web application development. A meta model of a generic web application structure was described in Ref. \textsuperscript{[11]}, and splits the websites into several components: web pages, frames, links, and forms. Based on this model, Ricca and Tonella\textsuperscript{[11]} designed a system to automatically analyze the websites. Additional web models were proposed in Refs. \textsuperscript{[12, 13]}.

There are two main kinds of user models used in web application performance analysis, namely, Markov models and CBMG. Mark and Csaba\textsuperscript{[14]} studied the connection between CBMG and the Markov model, and proposed an algorithm to transfer CBMGs to Markov chains by performing matrix manipulation.

There are also many studied on the automatic performance evaluation and analysis of web applications. An automatic web performance simulation and prediction system which is capable of automatically creating an online web performance simulation and conducting trend analysis of the system under evaluation was proposed in Ref. \textsuperscript{[15]}. In Refs. \textsuperscript{[16-19]}, some similar web data based analysis tools were proposed. WALTY\textsuperscript{[20]} used CBMG to implement a performance evaluation system of web applications, which is a set of tools that allows the performance analysis of web applications by means of a scalable what-if analysis on the test bed. The proposed approach is based on a workload characterization generated from information extracted from log files.

The difference between Ref. \textsuperscript{[20]} and this paper is the method employed to perform performance evaluation on the application. In Ref. \textsuperscript{[20]}, a group of sessions is generated based on the CBMGs in the evaluation execution process, while in our system, virtual users are generated based on URMG.

7 Conclusions

The CBMG is a useful method for the analysis of the users of web applications. However, it is not well suited to performance evaluation execution. The generation of CBMG is a generalization process, which collects the similarities of sessions. However, during the clustering process, some characteristics are lost, such as topology relationships between accesses within the same session, which is important for evaluation execution. In order to rebuild this relationship, this paper proposes a new model called the URMG. Based on some specific rules, URMG chooses a representation session to replace the statuses in CBMG. In this way, URMG gains the topology relationship between accesses and maintains the transfer relationships of CBMG. The performance evaluation shows that the error rate of evaluation based on URMG is reduced by 50\% relative to the evaluation based on CBMG.

Our future work will concentrate on the optimization of URMG. Because the representational session sometimes does not exist, and there may be some complex operations in the session which will affect the URMG generation process, the selection of both the representation session and the access records should be optimized. Besides, the evaluation result management module should be more intelligent, which can both analyze the evaluation results and identify the bottlenecks in the application.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61232008), the National High-Tech Research and Development (863) Program of China (Nos. 2013AA01A213 and 2013AA01A208), Chinese Universities Scientific Fund (No. 2013TS094), and Guangzhou Science and Technology Program (No. 2012Y2-00040).

References


Xiaolin Xu received his BS degree from Huazhong University of Science and Technology (HUST), China, in 2009. Currently he is a PhD candidate at HUST. His research interests are in the area of cloud computing and virtualization, focusing on resource management in the cloud.

Lixiang Tang received his BS and MS degrees from Huazhong University of Science and Technology in 2010 and 2013, respectively. His research interests are in the area of cloud computing and virtualization, focusing on management platform of the cloud.

Song Wu is a professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. He received his PhD degree from HUST in 2003. He is now the director of Parallel and Distributed Computing Institute at HUST. He has also served as the vice director of Service Computing Technology and System Lab (SCTS) and Cluster and Grid Computing Lab (CGCL) of HUST. His current research interests include grid/cloud computing and virtualization technology.
Hai Jin is a Cheung Kung Scholar, chair professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. He is now the dean of the School of Computer Science and Technology at HUST. Jin received his PhD degree in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. Jin worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. Jin is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of the National Key Basic Research and Development (973) Program Project of Virtualization Technology of Computing System. Jin is a senior member of the IEEE and a member of the ACM. Jin is the member of Grid Forum Steering Group (GFSG). He has co-authored 15 books and published over 400 research papers. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security.

Yihong Wang received his BS degree from Huazhong University of Science and Technology (HUST) in 2012. Currently he is a master student at HUST. His research interests are in the area of cloud computing and virtualization, focusing on resource management in the cloud.