A Nonintrusive Approach to Estimate Web Server Response Time

Mehul Nalin Vora

***Abstract*—With proliferation of internet and web-applications usage, server performance is a critical research area in last decade. This paper presents a nonintrusive approach to estimate server response time for user-initiated web-page requests from web-server access-log data using a variant of fuzzy c-means algorithm. The approach presented here is particularly suitable for a system of web-application(s) accessed concurrently by multiple users and running in a constrained production environment where minimal amount of monitoring or logging data is available. In such scenarios, proposed approach will be useful conducting advance system level testing as well as in verifying compliance to service level agreements (SLAs). Algorithm presented here has been tested using open source web application JPetStoreApp and java based load testing framework Grinder.**

***Index Terms*—Access log, common log format, response time estimation, web-mining.**

I. Introduction

Last decade of Social Web and current generation of Semantic Web have witnessed an exponential growth in web usage and thereby have presented vast opportunities to researchers and analysts. Many researchers have focused on enhancing quality of service (QoS) delivered by server and thereby improving user-experience by measuring several web-server performance parameters. There are numerous tools available in market today [1], [2]. Some are commercial, some are open source and others are research oriented. Most of these tools require execution of additional scripts either at server side or client side. But for all of them, response-time is key attribute in quantifying QoS and deriving user-satisfaction index [3], [4]. Response time measurement at client side will include not only server processing time but it will also include time required to establish connection as well as transmission time over network [5]-[7]. Although measuring response-time at client side will give accurate representation of quality of service experienced by web-users, it involves those components which are not governed by web-server and may not be helpful in gaining deep insight into current status of server and measuring web-server performance alone. On the other side, if we measure response-time at server side, it will incur some overhead on server and will consume some amount of system resources which may compete with running applications and hinder their performance. For the class of performance-critical applications it may not be possible to share system resources with monitoring tools and therefore usage of these tools become prohibitive. In absence of system monitoring tools, it becomes difficult to ascertain whether application is able to perform as desired and server is able to process user requests within the preset bounds. In this paper, we present an approach to estimate server response time for applications running in such strict environment in a nonintrusive way by analyzing web-server access-log in offline fashion.

During normal course of operation, any web-application in a production environment generates enormous amount of information in form of logs. The hidden information in these logs remains completely useless unless it is probed, uncovered, massaged and converted into meaningful representation which can result into fruitful actions. Predominantly monitored and used by system administrators for debugging purpose, web logs also have been proven as a powerful instrument for researchers in diverse areas like system management and performance analytics, profiling, prediction and business analytics, security analysis etc [8]. Although the field of Web-Mining (application of data mining to discover patterns from web data like server logs) has made significant progress, several challenges remain open. Here, we present an approach to analyze web-access logs to estimate server response time for a user initiated web-page request.

Rest of the paper is organized as follows: section 2 provides an introduction to access-log and describes challenges involved in the analyzing these log data. Section 3 outlines the approach and sketches the model used for log analysis and estimation of server response-time. Section 4 describes the experimental outcome of the proposed model. Section 5 lists some inherent limitations of the proposed solution followed by concluding section 6.

II. Access Log Data

Each http request initiated by a web-user gets logged at server in form of access-log data. The World Wide Web Consortium (W3C) maintains a standard format - Common Log Format (CLF) [9] for access-log data, but other formats also exist (e.g. NCSA extended / Apache Combined Log Format [10], W3C Extended Log Format [11], etc). Following is an example of entries in the CLF format and Table I explains fields for the CLF format:

127.0.0.1 - bob [16/Oct/2012:13:55:36 +0530] "GET /index.html http/1.1" 200 2326

127.0.0.1 - bob [16/Oct/2012:13:55:36 +0530] "GET /logo.gif http/1.1" 200 323

127.0.0.1 - bob [16/Oct/2012:13:55:36 +0530] "GET /counter.jpg http/1.1" 200 232

Manuscript received October 20, 2012; revised November 25, 2012.

Mehul is with the Innovation Labs, TATA Consultancy Services (TCS) Ltd., Mumbai INDIA (e-mail: mehul.vora@tcs.com).

TABLE I: FIELDS FOR COMMON LOG FORMAT (CLF)

Value	Description
127.0.0.1	IP address or host name of http client that made the request
-	The remote log-name of client
Bob	The username with which client has authenticated (if known)
[16/Oct/2012:13: 55:36 +0530]	Date, time, and time zone of the request.
"GET /index.html http/1.1"	Request line containing http request method, requested resource, and http protocol version
200	http Status code
2326	Size of the response object excluding the header returned to client, measured in bytes

For the purpose of determining server response time of a web-page request from access-logs, first we need to identify a set of requests belonging to that particular web-page request as one web-page is constructed by large number of independent components (alternatively called web-resources) and one user request for a single web-page results into multiple http request entries in server access-log, each corresponding to the constituting components of that page (e.g. *logo.gif* and *counter.jpg* are part of *index.html*). Once we have this information, we can estimate server response time for a web-page from timestamp data of these set of entries tagged that parent web-page request.

The task of analyzing access-log entries and tagging of each web-resource to a parent web-page is relatively simple for static HTML pages but in today's date, most of web applications use a dynamic system to generate a large number of pages from a small number of templates and a database causing complexity for analysis. Moreover it becomes increasingly difficult in concurrent and multiuser scenario. Since a web-server maintains a single access-log file, in case of concurrent web-users, access-log file contains interleaved entries resulting from those concurrent web-user actions and it becomes increasingly difficult to identify which http requests belongs to which web-user actions just by mere inspection. In addition, due to advent in NAT and NAPT usage, multiple web-users residing behind a common firewall/switch and accessing web concurrently, are masked by a common IP [12]. This adds an additional degree of difficulty in segregating web-resource requests. This task of page-tagging could have been relatively easier to achieve if server can be configured to capture extended information like session and referrer details. But this will impose additional disk-space requirement to capture those extra fields as well as system will take longer time for disk I/O to write those added fields to the logs. This may again cause performance hindrance problem for an application performing hundreds and thousands of transactions per second. Therefore it is a common practice for a system administrator to collect bare minimum information in CLF format prescribed by the W3C. In such situation, extracting knowledge pertaining to web-pages and their corresponding components using minimal information from access-log data remains an open challenge.

III. MODEL

In this section, we present a model to estimate server response time by segregating individual http requests from access-log data and tagging each request to parent web-page requests. Without any prior knowledge of an application, unsupervised learning approach using clustering technique is a natural choice for this problem. Group distinct requested web-resources into one logical cluster representing one user initiated request for a web-page. However, there exist some fundamental difficulties in clustering of http requests compare to clustering in traditional applications. First, access-log data contains mixed (both numerical and non-numerical) variables and also pattern evolving from such data is non-numerical. Second, due to caching effect (proxy as well as browser caching) likelihood of incomplete data is higher than traditional applications. Third, growing volume of web-application transactions and chaotic access behavior of end-users increase complexity for analysis. Last but the most importantly, the problem itself introduces inherent uncertainty in analysis as one web-resource may be part of more than one cluster. Fuzzy logic provides a natural framework for process involving non-random uncertainty and impreciseness. Initially proposed by Zadeh [13], Fuzzy set theory allows uncertainty of belonging – one entity can be a member of multiple groups (clusters) with varied degree of membership. Usually, membership of an entity for a cluster is calculated based on the distance from the cluster center. Based on the concept of fuzzy c-partitions, introduced by Ruspini [14], Fuzzy c-Mean (FCM) algorithm, proposed by Bezdek [15], [16], is one of the most well known algorithm in literature on clustering analysis and fuzzy clustering. FCM and other variants of FCM algorithm available in literature [17]-[21] (just to list a few) have been proposed for wide range of problems from various domains. Here in this section, we will describe this approach in three steps: data pre-processing, clustering and estimation of response time.

A. Data Preprocessing

Data pre-processing tasks include data filtering, transformation, grouping and sorting. The first goal of analysis is to identify parent web-pages and tag all other http requests for other web-resources from access-log data to appropriate parent web-pages. To achieve this goal, we filter out impertinent entries from access-log data. The log entries with erroneous status code 4xx and 5xx are excluded from analysis. As well as we filter out the entries with http requests other than get, put and post. Then convert access-log data into a Dataset $X \in \{x_{ij} \mid 1 \le i \le N;\ 1 \le j \le V\}$ containing N number of data-points (http requests) and V number of measurements (variables) for each data-point. Here, we have used two non-numerical data fields - information pertaining to client IP and component or resource requested (URI request); and one numerical attribute that is time for the request (time-stamp). Although this algorithm was designed around the central idea of CLF format, if a system is capturing additional custom fields and if those parameters could be useful for the targeted analysis, one can include these parameters by specifying their data-type (i.e. numerical or non-numerical).

After filtering relevant data, we identify all http requests corresponding to parent web-pages. Next we group entries coming from same IP, preserving order appeared in access-logs and sort in ascending order of total number of

parent web-page requests coming from same IP. Although one can use entire data-set in one pass for subsequent phase, we strongly recommend using incremental learning approach - analyzing requests coming from one IP at a time, starting with an IP with least number of web-page calls. One can further subdivide data by specifying upper limit on transaction duration. If difference in timestamp entries for two successive http requests coming from one IP is greater than this pre-specified limit, algorithm marks the first entry as end of one logical session the second entry as beginning of new and divides data into two subsets for further analysis. We recommend value for this parameter as twice the value for maximum time permitted by the SLA. The obvious advantages of adopting incremental learning approach are: a) smaller the size of data to be analyzed in one step, faster the convergence of iterative clustering algorithm presented below and b) higher confidence in membership matrix (output of clustering step described below) generated. Finally, for each non-numerical variable j, first we prepare a master list of all possible values $(J_1, J_2, \cdots J_{v(j)})$ and each numerical variable is unit-normalize as shown in (1). Here J_{min} and J_{max} are minimum and maximum values for the numerical variable j respectively.

$$x_{ij} = \frac{x_{ij} - J_{min}}{J_{max} - J_{min}} \tag{1}$$

B. Clustering for Access-Log Data

Here we have used a variant of traditional FCM algorithm which can accommodate both numerical as well as non-numerical variables. This is an iterative algorithm and it requires five input parameters: Dataset X representing pre-processed entries from access-log data; number of clusters C sought (an integer $2 \leq C \leq N$); fuzziness factor ϕ (a real value $1 \leq \phi < \infty$); tolerance limit ε (a small positive real value); and maximum number of iterations *max_itr* (a large positive integer).

Initialization phase of this algorithm involves generation of one random membership matrix $M \in \{m_{ik} \mid 1 \leq i \leq N;\ 1 \leq k \leq C\}$ such that $0 \leq m_{ik} \leq 1$ and $\sum_{k=1}^{C} m_{ik} = 1$. Here m_{ik} represents membership value of i^{th} data-point in k^{th} cluster. Instead of generating membership matrix completely in random fashion, we can initialize this matrix as per frequency of each requested resource appearing in log data, which will help in achieving faster convergence of the following iterative steps.

1) From the membership matrix M, calculate cluster centers matrix $CC \in \{c_{kj} \mid 1 \leq k \leq C\ 1 \leq j \leq V\}$ as follows:
   a) *For numerical variables:* Cluster center is calculated as a fixed point as per (2).

$$c_{kj} = \frac{\sum_{i=1}^{N} (m_{ik})^{\phi} x_{ij}}{\sum_{i=1}^{N} (m_{ik})^{\phi}} \tag{2}$$

   b) *For non-numerical variables:* Cluster center is calculated as an influence vector given by (3)

$$c_{kj} = \{\lambda_{kj1}, \lambda_{kj2}, \ldots, \lambda_{kjv(j)}\} \tag{3}$$

Here λ_{kjl} represents influence value of a particular value J_l of variable j, while determining the cluster center c_{kj} such that $0 \leq \lambda_{kjl}\ (\forall l \in [1, v(j)]) \leq 1$ and $\sum_{l=1}^{v(j)} \lambda_{kjl} = 1$. The influence value λ_{kjl} is calculated as per (4).

$$\left.\begin{aligned} \lambda_{kjl} &= \frac{\sum_{i=1}^{N} (m_{ik})^{\phi} \pi_{il}}{\sum_{i=1}^{N} (m_{ik})^{\phi}} \\ &where \\ \pi_{il} &= \begin{cases} 1 & if\ x_{ij} = J_l \\ 0 & otherwise \end{cases} \end{aligned}\right\} \tag{4}$$

2) Based on the matrix CC, calculate dissimilarity matrix $D \in \{d_{ik} \mid 1 \leq i \leq N;\ 1 \leq k \leq C\}$ as per (5) where W_j represents weight assigned to variable j.

$$d_{ik} = \sum_{j\ \epsilon\ non-numerical} W_j \times \left(\sum_{l=1, x_{ij} \neq J_l}^{v(j)} \lambda_{kjl}\right)^2 + \sum_{j\ \epsilon\ numerical} W_j \times \left(x_{ij} - c_{kj}\right)^2 \tag{5}$$

3) Based on this dissimilarity matrix D, re-compute the membership matrix M as per (6):

$$m_{ik} = \frac{1}{\sum_{q=1}^{C} \left(\frac{d_{ik}}{d_{iq}}\right)^{\left(\frac{2}{\phi - 1}\right)}} \tag{6}$$

These steps are repeated until algorithm achieves convergences ($|M_{itr} - M_{itr-1}| \leq \varepsilon$) or maximum number of iteration is reached. At the end of iterative phase, each entry from access-log data will have a membership vector representing probability of this web-component belonging to each parent web-page call appearing in access-log. Here we have just described the mechanics of FCM algorithm. For the mathematics of objective optimization and derivation of these equations, interested readers are requested to refer [16].

C. Response Time Estimation (RTE)

Based on the membership matrix generated at the end of clustering phase, we tag each http request for a web-component to a parent web-page call having the highest membership value for that web-page cluster. At this point, we perform a validation check – no two http requests for a particular web-component can get tagged to the same parent page. In multiuser scenario, it is often possible that concurrent users accessing the same web-page virtually at the same time. Due to which, access-log will have multiple entries for the same parent page and its constituent components having almost same timestamp. In such cases, algorithm may give same probability to multiple requests for a particular web-component. So here we assign first request to the cluster having the highest probability, second request to the cluster having second highest probability and so on. Once we have tagging information, we can look at each web-page request cluster and then estimate response time as difference between timestamps of the earliest and the latest entries for that particular cluster given by (7).

$$RT_k = TimeStamp_{Last} - TimeStamp_{First} + 1 \tag{7}$$

This approach for RTE is completely non intrusive as it does not stipulate any alteration to existing system configuration or application-code. Moreover since there is no need to perform this analysis on the same web-server, log

files can be offloaded and analysis can be done offline. Therefore it does not impose any additional load on system or impact performance of production-setup.

TABLE II: VALIDATION FOR PAGE-TAGGING

	Dataset 1	Dataset 2	Dataset 3 (In-house Exp)
Accuracy	98.66 %	98.62 %	98.34 %

IV. EXPERIMENTAL ANALYSIS

The proposed algorithm is validated in two steps: first step was to validate the page tagging algorithm and second was to assess accuracy of response time estimation. We have collected access-log data from three different sources. Two datasets were obtained from two different project teams maintaining live applications in production setup and third access-log dataset was generated by conducting an in-house experiment. For this experiment three applications were hosted on a common apache tomcat web-server. Three applications deployed were: a) JPetStoreApp [22]: It is a Pet Store application based completely on open source freeware (struts and others); b) NxGCel: It is a telecom reporting application on mobile usage; and c) VINS: Vehicle Insurance System is an in-house application. Lab members were requested to access these applications concurrently for few hours to generate enough entries in access-log.

Although the page-tagging algorithm is designed for CLF format, we have customized it to capture *Referrer* – URL containing link (parent web-page) to resource (child component) being requested. This additional field is used only for validation purpose and has not been used in clustering phase described above. All experiments were conducted using following values for input parameters: number of clusters *C* as number of page calls, fuzziness factor ϕ as 1.5, tolerance limit ε as 10^{-6} and maximum iterations of 100. Table II describes accuracy of the page-tagging algorithm.

To validate response-time estimation, we conducted another set of experiments. For this purpose we have used an open source load testing framework, Grinder [23] and JPetStoreApp running on Apache Tomcat web-server. To vary degree of concurrency we have used 30 users having an inter-arrival time (IAT) of 30, 20 and 10 seconds. Fig. 1 shows average response-time of eight web-pages of JPetStoreApp in load-testing experiments. Fig. 1 also compares these results with estimated average response time by analyzing access-log data generated during load testing runs. Table III lists absolute error in response time estimation compared to value measured by Grinder.

TABLE III: ABSOLUTE ERROR

	30s IAT	20s IAT	10s IAT
Page 1	0. 67 %	11.97 %	19.41 %
Page 2	11.15 %	9.23 %	2.65 %
Page 3	14.20 %	7.21 %	10.57 %
Page 4	15.61 %	3.90 %	1.82 %
Page 5	7.15 %	10.38 %	16.86 %
Page 6	16.74 %	0.27 %	10.70 %
Page 7	4.20 %	4.42 %	0.43 %
Page 8	3.64 %	7.25 %	12.82 %

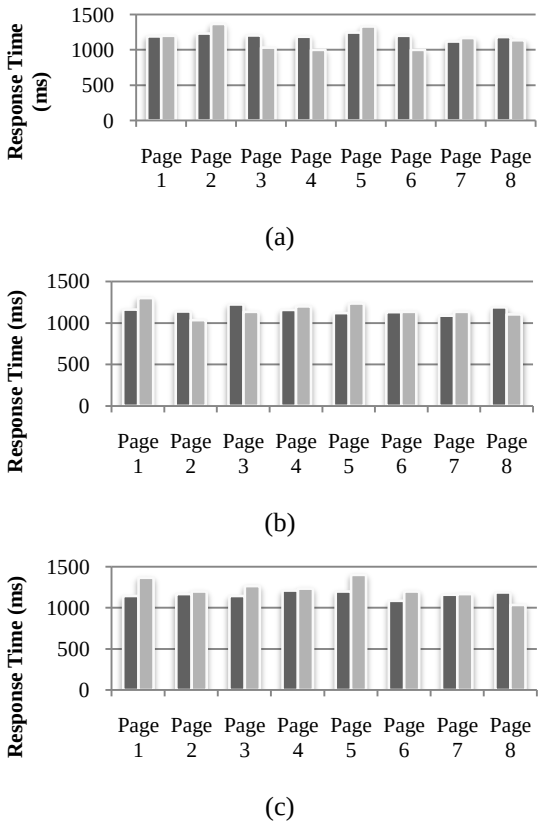

Fig. 1. Response time estimation for grinder experiment. (■ : measured; □ : estimated) (a) IAT 30s; (b) IAT 20s; (c) IAT 10s

One should note that Grinder reports end-to-end response time. It represents cumulative value for response time in all three phases of one http request: connection phase, server processing phase and network transmission phase, whereas proposed algorithm estimates only server response time. With the assumption that network delay is almost negligible in LAN, time measured by Grinder equals to server response time. From table III one can ascertain that upper bound on error in response time estimation is about 20%. These results are not surprising. First, difference in precision for time measurement (Grinder can measure response time with millisecond level precision whereas CLF format provides timestamp data with precision of seconds only) and the fact that average response time of a web page, measured by Grinder in these experiments, is between 1 to 1.4 seconds which is very close to the smallest value given by (7) (i.e. 1 second), attributes to higher rate of error in response time estimation obtained from access-log analysis. Second, log generated using Grinder represents robotic behavior compared to actual human behavior in real world application. This robotic behavior is also another important dimension in contributing towards error in response time estimation. But as mentioned earlier, for many applications running under performance constrained environment, where no monitoring data and minimal amount of logging data is available, this estimation can serve as an indicator for application performance and can help in gaining insight knowledge about server status.

V. LIMITATIONS

One assumption made in this analysis is presence of sticky

sessions - all communication between a user and server in one session is over same path. For many applications, it is true, however one may consider as a shortcoming of this approach in today's dynamic world. Other assumption made in (7) is that server will take some finite amount of time in responding each user request for a web-page. Since CLF format records timestamp with precision of seconds only, we often get $TimeStamp_{Last} = TimeStamp_{First}$. Therefore we have here assumed that the lowest amount of time taken by server to respond is 1 second. One can derive at a conclusion that this approach can provide only an estimation and not actual server response time.

VI. Conclusion

In absence of knowledge pertaining to web-pages and their constituting components, one can derive these pieces of information and estimate server response time just by analyzing information available in server access-log data using the model described in this paper. This approach is completely non intrusive as it does not stipulate any alteration to existing system configuration or application-code. Moreover since there is no need to perform this analysis on the same web-server, log files can be offloaded and analysis can be done offline. Therefore it does not impose any additional load on system or impact performance of production-setup. For many applications running under performance constrained environment, where no monitoring data and minimal amount of logging data is available, estimation approach presented here can be very useful as it can serve as an indicator for application performance and can help in gaining insight knowledge about server status. In future, we are planning to cross-relation with other infrastructure log-data like system-resource usage data to perform workload characterization, resource demand estimation and availability and reliability analysis for system in production just by analyzing system-log data.

Acknowledgment

I would like to thank each and every member of Innovation Labs for their kind assistance, supporting this work with ideas and criticism, and for their extended help to make this work successful.

References

[1] Wikipedia – List of web analytics software. [Online]. Available: http://en.wikipedia.org/ wiki/List_of_web_analytics_software
[2] E. Peterson. Web Analytics Demystified: A Marketer's Guide to Understanding How Your Web Site Affects Your Business. [Online]. Available: http://www.webanalyticsdemystified.com
[3] A. Savoia, "Web Page Response Time 101: Understanding and measuring performance test results," *STQE Magazine*, pp. 48–53. 2001.
[4] D. Shin, K. Koh, and Y. Won, "Measurement and Analysis of Web Server Response Time," in *Proc. of Sixth APCC*, Seoul, Korea, Oct. 2000.
[5] D. P. Olshefski, J. Nieh, and D. Agrawal, "Inferring Client Response Time at the Web Server," in *Proc. ACM SIGMETRICS ICMMCS*, NY, USA, Jan. 2002, pp. 160-171.
[6] J. Wei and C. Xu, "Measuring Client-Perceived Pageview Response Time of Internet Services," *IEEE Trans on Parallel and Distributed Systems*, May 2011, vol. 22, no. 5, pp. 773-785.
[7] T. Chiew and K. Renaud, "Disassembling Web Site Response Time," in *Proc. Twelfth ECITE*, Turku, Finland, September 2005, pp. 137-146.
[8] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, NY, USA, vol. 55, no. 2, February 2012, pp. 55-61.
[9] W3C Common Log Format (CLF) Description. [Online]. Available: http://www.w3.org/ Daemon/User/Config/Logging.html#AccessLog
[10] Combined Log Format. [Online]. Available: http://httpd.apache.org/docs/2.4/mod/ mod_log_config.html#formats
[11] W3C Extended Log Format. [Online]. Available: http://www.w3.org/TR/WD-logfile.html
[12] Network Address Translation. [Online]. Available: http://en.wikipedia.org/wiki/ Network_Address_and_Port_Translation
[13] L. A. Zadeh, "Fuzzy Sets," *Inf. and Ctr*. vol. 8, no. 3, 1965, pp. 338-353.
[14] E. Ruspini, "A New Approach to Clustering," *Inf. and Ctr.*, vol. 15, no. 1, 1969, pp. 22-32.
[15] J. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms*," Kluwer Academic Publishers Norwell*, MA, USA, 1981.
[16] J. Bezdek, Robert Ehrlich, and William Full, "FCM: The Fuzzy c-Means Clustering Algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, 1984, pp. 191-203.
[17] S. Chatzis, "A fuzzy c-means-type algorithm for clustering of data with mixed numeric and categorical attributes employing a probabilistic dissimilarity functional," *Expert Systems with Applications*, vol. 38, 2011, pp. 8684–8689
[18] M. Yanga, W. Hungb, and F. Cheng, "Mixed-variable fuzzy clustering approach to part family and machine cell formation for GT applications," *Int. J. of Prod. Economics*, vol. 103, 2006, pp. 185–198.
[19] S. Nascimento, B. Mirkin, and F. Moura-Pires, "A Fuzzy Clustering Model of Data and Fuzzy c-Means," in *Proc. Ninth IEEE International Conference on Fuzzy Systems*, 2000.
[20] D. Kim, K. H. Lee, and D. Lee, "Fuzzy clustering of categorical data using fuzzy centroids," *Pattern Recognition Letters*, vol. 25, no. 11, 2004.
[21] M. Yang, P. Hwang, and D. Chen, "Fuzzy clustering algorithms for mixed feature variables," *Fuzzy Sets and Systems*, vol. 14, 2004, pp. 301–317.
[22] JPetStoreApp: based on J2EE Pet Store application. [Online]. Available: http://sourceforge.net/ projects/ibatisjpetstore/
[23] Grinder: an open source load testing framework. [Online]. Available: http://grinder.sourceforge.net/

Mehul Vora has received B.E. (Chemical Engineering) degree from University of Mumbai (India), in 2000, and M.S. (Computer Science) & Ph.D. degrees from the Clarkson University, NY (USA) in 2003 and 2007, respectively. Since 2008, Mehul has been associated with Tata Consultancy Services (TCS) Limited, Mumbai, India. He has served diverse roles like Measurement Science Analyst and Business Analyst. In his current role as R&D Scientist at Innovation Labs, TCS, he is associated with performance engineering research center.