Performance Metrics of a Customized Web Application Developed for Monitoring Sensor Data

Hidam Kumarjit Singh, Member, IEEE
Department of Electronics and Communication Technology
Gauhati University
Guwahati-781014, India
kumarjit_hidam@yahoo.com

Tulshi Bezboruah, Senior Member, IEEE
Department of Electronics and Communication Technology
Gauhati University
Guwahati-781014, India
zbt_gu@yahoo.co.in

***Abstract*—**An interactive .NET web application is designed for use in remote monitoring of sensor response within a Local Area Network zone. Key performance metrics of the web application such as response times, throughput, processor and disk utilization are measured by employing a standard testing tool. The impact of concurrent users' activities on the performance metrics of the web application have been observed by using two test scenarios.

Keywords – Data, Intranet, performance, Sensor, .NET

I. Introduction

Performance metrics of web applications indicate how effectively the web applications running in a web server behave against different workloads and concurrencies [1]. Performance testing is divided into different techniques such as load testing, stress testing, soak testing and spike testing [2]. Every testing technique has its unique objective and requirements. Performance testing enables to find potential bottlenecks that will hamper performance of the web applications when they are subjected to extreme conditions of operation and workloads [3]-[4]. Performance testing are done by using standard testing tools and methods to predict response times, transaction errors, windows resource usage, network utilization, security, availability, scalability and reliability etc. of the web applications [5]-[6]. With the advancement of web technologies, Internet has also been increasingly exploited as flexible communication channel for collecting data from remote sensors and controlling devices, as in [7]-[9] and [11]-[17]. In the present paper, we describe the details of performance testing done for predicting performance of a customized web application. The web application has been designed in ASP.NET and used for monitoring and control of sensor data through Internet in Local Area Network (LAN) environment.

II. Web Application Developed For Monitoring Sensor Data

A. Graphical user interface components

The web application has been designed with Microsoft's Visual Studio 2010 Professional software. The user interface components of the web application comprises of ASP.NET controls like *Text Box*, *Labels*, *Chart*, *buttons*, *Timer* and *Update panels*. There are five web pages in the web application, namely: a Home page, a Log-in page, a Control page, a Data view page and an Unlock page. These web pages act as front end user interface to clients. The Home page provides the hyperlinks for other web pages; Log-in page enables users to get access to the Control page for monitoring sensor data. Data view page enables user to view already recorded sensor data. Unlock page is used by the system administrator for unlocking the Log-in page. **Fig.1** shows the simplified functional blocks of the user interface.

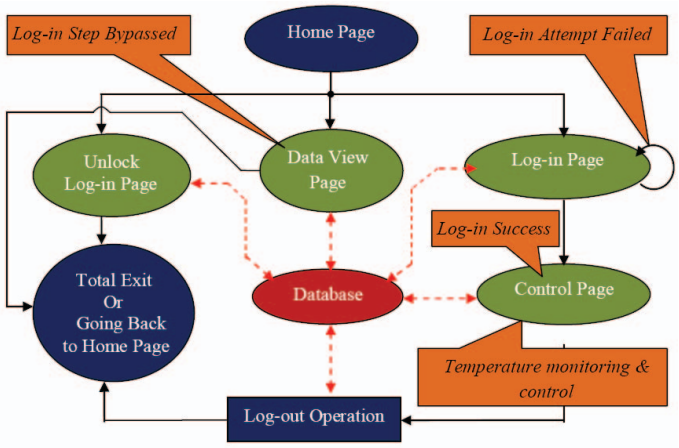

Fig.1. Components of the web application design for monitoring sensor data.

B. Business logic and algorithms of the programs

The fundamental purpose of the web application is to enable monitoring and control of temperature remotely from a client machine in LAN. The control and monitoring tasks are taken care of by a data acquisition (DAQ) program incorporated to server side script of the Control page of the web application. However, the control page is not made accessible to users in concurrent mode, because this web page directly interacts with DAQ hardware. When user activities are allowed to take place in concurrent mode, operational conflict can occur in the DAQ hardware. Hence a predefined variable known as *user flag* is used to control access right of the Control page. The logic status of user flag is asserted to 1 upon

978-1-4799-8349-0/15/$31.00 ©2015 IEEE

successful log-in, and then no other users are allowed to access the Control page as long as *the log-in flag* remains in logic 1. The flag status gets reset to 0 after a successful log-out operation. A system administrator uses Unlock page for overwriting the erroneous logic status of user flag, which can be caused by a previous failed log-out operation. However, concurrent users are allowed to access Data view page without any restriction.

Visual Basic .NET was used for writing the server side program codes of the web application. During the execution of server side codes, the web application communicates with a customized DAQ hardware that is interfaced to the server. The server acquires digitized sensor data from the DAQ hardware through the printer port. Visual Basic.NET performs read and write operations with printer port of the server PC with a dynamic link library (DLL) file, as in [8]-[10]. In the present work, the acquired sensor data corresponds to temperature of a climate chamber. The acquired sensor data are updated on the Control page in the form of alphanumeric as well as graphical plot. At the same time, the sever side code also stores the acquired sensor data to a Microsoft Access database in time stamped manner.

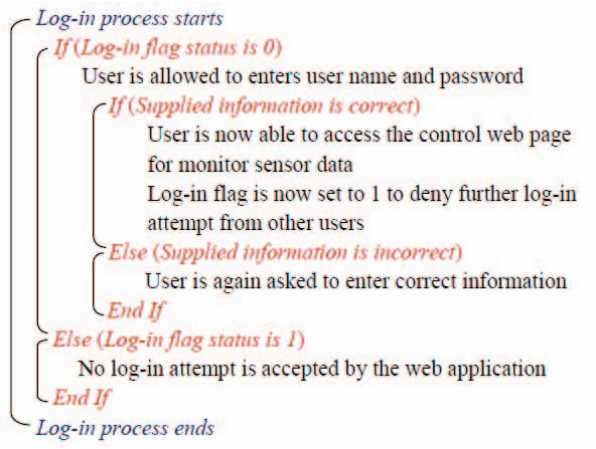

Fig.2. Algorithm for Log-in process employed in the Log-in page.

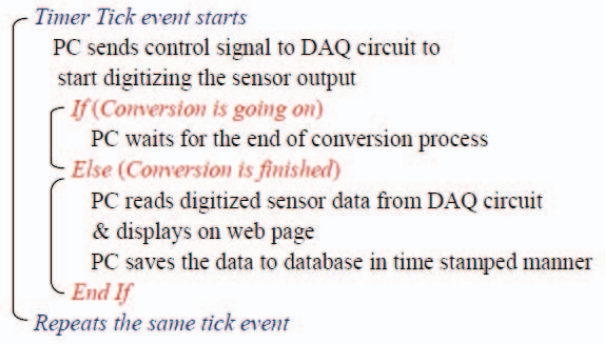

Fig.3. Algorithm for acquiring and displaying sensor data in Control page.

The algorithm of the server side code developed for log-in process is shown in **Fig. 2**. After successful log-in, the user can now start monitoring the sensor data through control page. **Fig. 3** shows the algorithm of the DAQ program developed.

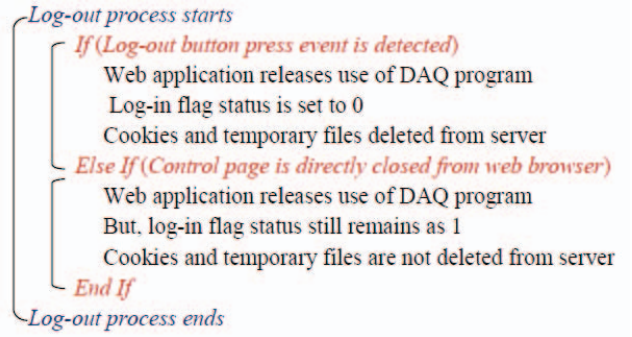

Fig.4. Algorithm for log-out process from the Control page

When the monitoring task is over, the user must log-out from the Control page rather than directly closing the web page from a web browser. The algorithmic description of the log-out process is shown in **Fig.4**. The algorithm of the program used for observing recorded sensor data from Data view page is not included here, because it works as a simple database binding program. **Fig.5** shows the algorithm of the program code used for unlocking Log-in and Control pages.

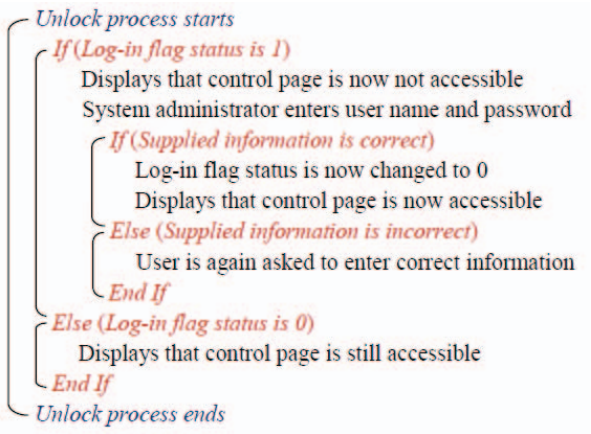

Fig. 5. Algorithm for unlocking Log-in and Control pages.

III. DAQ Circuit Used For Testing The Web Application

The actual snapshot of the DAQ hardware controlled by the web application is shown in **Fig.6**. This circuit board comprises of an oscillator, a digitizer and a temperature control circuit. The oscillator is designed with Schmitt trigger chip HC74LS14 and it acts as a temperature sensor with the help of a thermistor, as in [10]. The digitizer is designed with an 8-bit microcontroller (μC) and it acts as frequency to digital converter (FDC), as in [11]. A miniature heating oven is used for changing temperature of the climate chamber, and a simple ON/OFF switching circuit with a cooling fan add-on is used as temperature controller. The digitizer output is allowed to be read by the web server PC through data lines of printer port. The server PC controls operation of FDC and temperature

controller by transmitting control signals through control lines of printer port.

Fig.6. Actual snapshot showing interfacing of the DAQ circuit to printer port of local web server PC (The server PC is arranged with a normal PC having Windows XP Professional service Pack III operating system).

IV. Testing Of The Web Application

A. Client-server configuration

The web application is run as a website of a local web server machine with Internet Information Service (IIS) 7.5 (Express Edition of Microsoft's web server application software). The server and the client machines are connected to a Gigabit LAN network, as shown in **Fig.7**. The necessary software settings (Windows Firewall and IIS attributes) are applied to the web server for enabling the clients to access the local website.

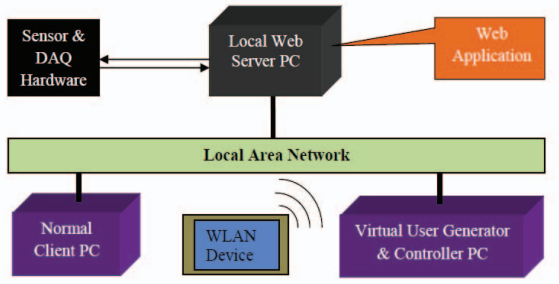

Fig.7. Configuration of clients and server machines in LAN.

Since the entire LAN is controlled by a corporate proxy server (for making connection to outside Internet), proxy server bypass settings are needed to be applied in the web browsers of client machines. So, the default access restriction of the local website, which is imposed by the proxy server on the clients, is avoided. Thus, the clients can access the local website easily for remote monitoring and control purposes.

B. Functional testing

Functional testing of the web application is done to find out presence of functional and logical errors in the application. During functional testing, users interact with the web application by using various control menus, and buttons provided on the web pages. Then, the responses provided by the web application are compared to the expected results. Thus, every web page is observed one after another. If errors are found, necessary modifications and changes are applied to the program codes. The current web application is found to be working without any functional and logical errors. **Fig.8** shows the screenshot of Control page obtained during the functional testing.

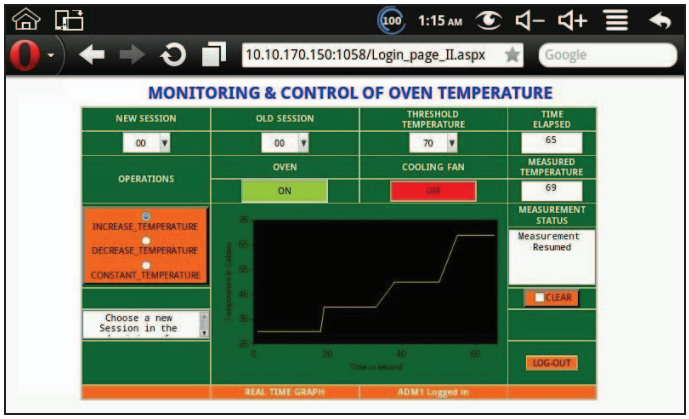

Fig.8. A screenshot of Control page browsed in web browser of an Android tablet (The actual uniform resource locator (URL) link of the web page is kept hidden as security measure).

C. Load testing

The load testing option has been chosen in the present work. There are several standard testing tools (proprietary software packages) for characterizing performance of web application, and Mercury Load Runner (MLR) is one of them. In the present work, MLR (version 8.1) has been used as testing tool. At first, we use an additional PC having MLR to act as virtual user generator cum load controller machine in the client – server network of **Fig.7**. In order to record test scripts, real users are allowed to interact with web application through MLR's script recording engine, and in the process the MLR captures the live actions of the user, as in [2] and [6]. A recorded script consists of program codes that emulate activities of a real user. Scripts are generated by MLR in C language as default. Then, we use the recorded script in load MLR's load controller engine to carry out load testing. The test scenarios used by the controller are given in Table 1.

Table 1. Test scenarios used in the load testing.

Name of the scenario	Characteristics of the scenario
Scenario-1	A group of 60 virtual users are used and the virtual users are allowed to access *Control page* and *Data View page* of the web application in 50:50 user ratio.
Scenario-2	All the virtual users are allowed to access *Control page*, but they are prohibiting from accessing *Data View page* of the web application.

The most important objective of the load testing done in the present work is to observe the impact of virtual user activities on response time of the web application. That is why, in both test scenarios, virtual users are allowed to vary w.r.t time in the manner shown below in Table 2.

Table 2. User pattern and runtime setting

Total number of virtual users	User Ramp up rate	Runtime duration in (minute)	Total testing Time in (minute)	User Ramp down rate
60	Number of user increases by 2 after every 15 seconds	5	12.5	All the users stops simultaneously

We also expect the response time to be affected by size of dataset present in a database (known as sensor database) kept in the web server itself. Sensor data acquired during temperature monitoring and control tasks are stored in the sensor database. So, its dataset size is supposed to increase with time as long as the DAQ program has been accessed. Further, the virtual users are always going to have interaction with this database in both the test scenarios. So, response time of the web application has been recorded by considering different dataset size pattern, as shown in Table 3.

Table 3. Dataset size patterns and their interpretations

Dataset size in sensor database	Interpretation for the dataset size
1 K	One thousand rows and each row has five columns for alphanumeric data
30K	Thirty thousand rows and the same number of columns
70K	Seventy thousand rows and the same number of columns
100K	Hundred thousand rows and the same number of columns

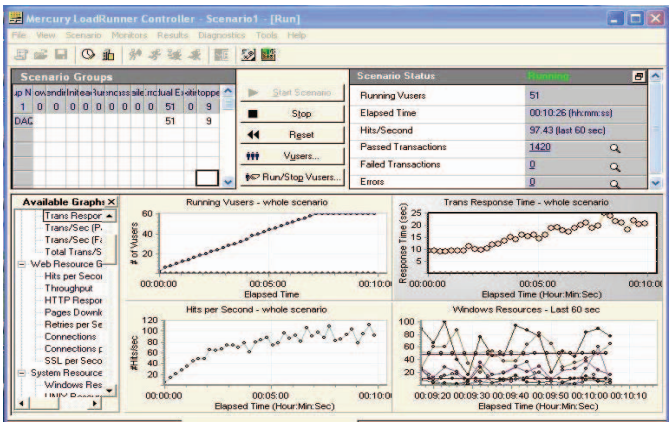

Fig.9. Screenshot of Load Runner showing measured performance metrics during Load testing.

V. EXPERIMENTALLY OBSERVED DATA

A typical screenshot of load testing process with MLR is shown above in **Fig.9**. Measured data are shown in the form of scattered plots in MLR's working window. After the end of a test, these data are automatically saved in hard disk of the load controller. In the present work, we exported the recorded data to Microsoft Excel for drawing customized scattered plots. **Fig.10 – Fig.17** show the scattered plots drawn for various measured parameters such as response time, throughput, server processor time, and disk time.

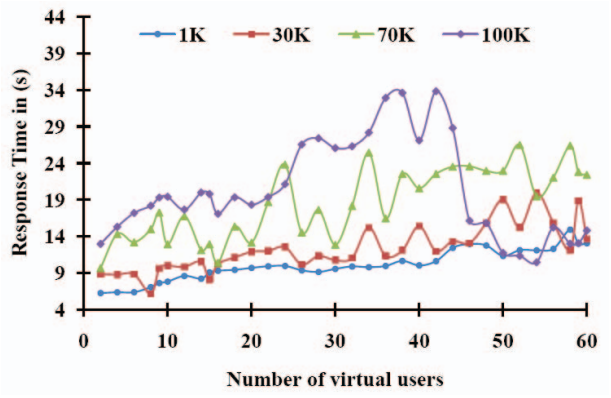

Fig.10: Response times versus virtual users under scenario-1

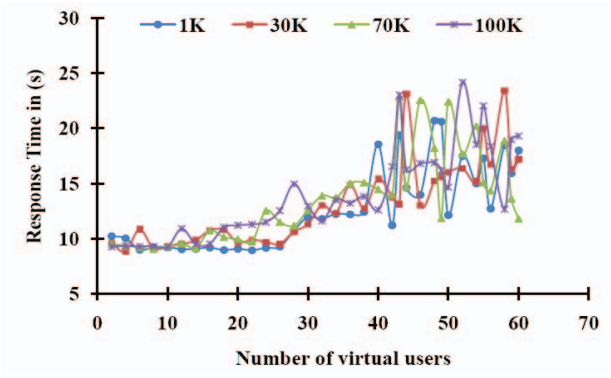

Fig.11.Response times versus virtual users under scenario-2

Fig. 10 and **Fig.11** show variation of response times of the web application obtained with test scenarios 1 and 2 respectively. In both scenarios, response time increases with the increase of virtual users and size of dataset in sensor database. However, two distinct differences can be observed from the above figures. On the average, the response time increases rapidly with virtual user in scenario 1 as compared to scenario 2. And, the maximum response time obtained under scenario 1 is found to be higher than that of scenario 2 by nearly 10 seconds when the size of dataset is 100K.

Typical value of acceptable response time lies in between 3-5 seconds. So, the response times obtained in both the scenarios are comparatively larger than the acceptable bench mark level. The server PC has 2 GB of RAM, 300 GB of hard disk and 1Gbps Ethernet adapter. But, its CPU unit has only two cores. So, the higher response time can also be due to limited hardware resources of the server PC.

Fig. 12 and **Fig.13** show variation of throughput (data bytes per second) obtained with test scenarios 1 and 2 respectively. Maximum throughput is obtained under scenario 1, but it is found to be significantly affected by different size of

dataset. Throughput levels increase much faster in scenario 2, but the effect of dataset size is much reduced as compared to that of scenario 1.

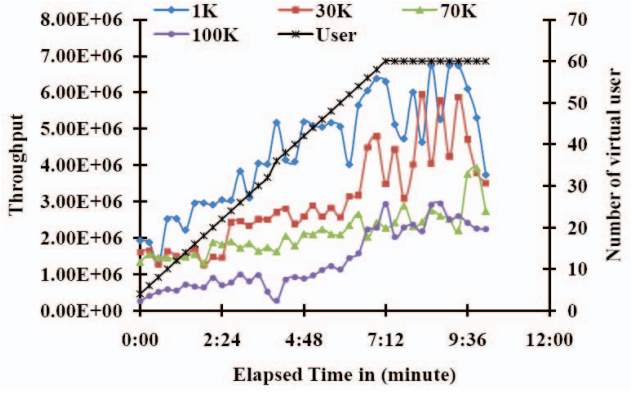

Fig.12. Throughput versus virtual users under scenario-1

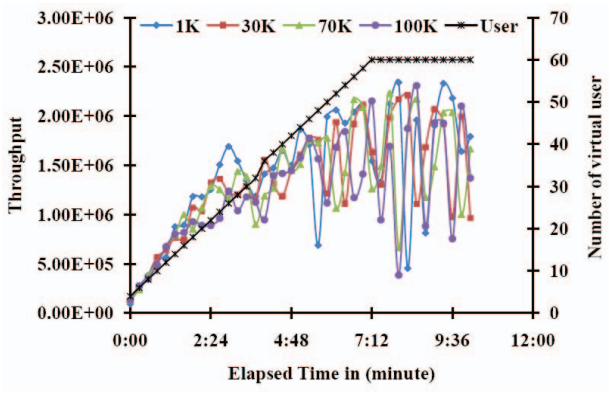

Fig.13. Throughput versus virtual users under scenario-2

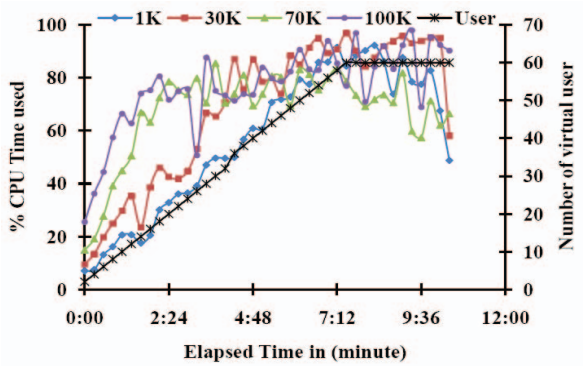

Fig.14. CPU time utilization versus virtual users under Scenario-1.

Fig. 14 and **Fig.15** show variation of CPU time under scenarios 1 and 2 respectively. Maximum usage of CPU time is found in scenario 1 and is found to be significantly affected by size of dataset of sensor database during the ramp up phase of the test. In scenario 2, the CPU time usage is found to be less affected by change in size of dataset of sensor database, as evident from narrow separation between the curves. The maximum CPU usage goes above 80% in scenario 1, whereas it lies below 80% in scenario 2. Typical value of acceptable level of CPU consumption can go as high as 60%, and if it goes beyond this threshold level, CPU usage is considered to be a bottleneck for the web application under test.

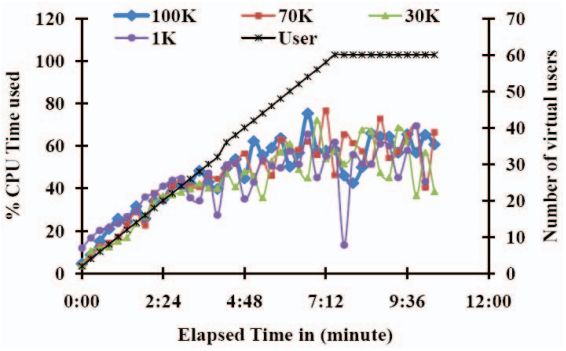

Fig.15. CPU time utilization versus virtual user under scanrio-2

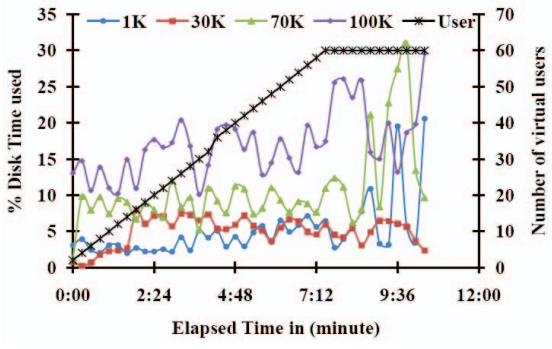

Fig.16. Disk time utilization versus virtual user under scenario-1.

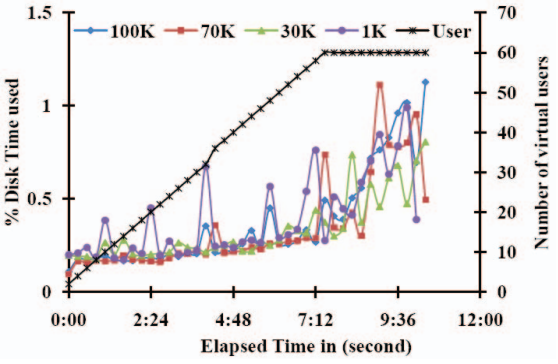

Fig.17. Disk time utilization versus virtual users under scenario-2

Fig. 16 and **Fig.17** show variation of disk usage time under scenarios 1 and 2 respectively. Maximum usage of disk time is found in scenario 1 and is found to be significantly affected by size of dataset of sensor database during the ramp up phase of the test. In scenario 2, the disk usage time is found to be less affected by change in size of dataset of sensor database, as evident from narrow separation between the curves. The maximum disk usage goes above 20% in scenario

1, whereas it lies below 1.5 % in scenario 2. Higher disk usage associated with scenario 1 will be due to larger extent of database operation due to concurrent users. The summarized test results are tabulated in Table 4 and 5 for each test scenario.

Table 4: Average test results under test scenario-1

Scenario-1: Activities types - database browsing + Process control				
Number of data rows in databse	Average Response Time (second)	Average Throughput (bytes/second)	Average %CPU Time used	Average %Disk Time used
1×10^3	10.60	3,666,799.00	56.15%	3.52%
30×10^3	15.45	2,196,622.54	72.45%	4.57%
70×10^3	22.37	1,525,850.00	75.26%	7.06%
100×10^3	29.05	1,292,517.00	76.54%	11.77%

Table 5: Average test results under test scenario-2

Scenario-2: Activities types - Process control only				
Number of data rows in database	Average Response Time (second)	Average Throughput (bytes/second)	Average %CPU Time used	Average %Disk Time used
1×10^3	16.86	1,691,660.27	56.35%	0.54%
30×10^3	16.12	1,710,820.00	54.01 %	0.49%
70×10^3	15.83	1,695,225.66	52.62%	0.68%
100×10^3	16.83	1,669,930.99	53.05%	0.59%

VI. Test Results And Discussion

Table 4 and 5 show the average values of response time, throughput, CPU and disk usage. After after observing these tables, we can conclude that scenario 1 yields poor performance metrics. This can be due to the fact that scenario 1 involves concurrent activities of the virtual user, whereas concurrent user activities are absent in scenario 2. The higher response time and higher CPU usage level of the web application can be improved by using a web server PC having larger memory and multiple number of processors. The number of virtual users used in the current work is quite less as compared to testing of large scale web applications. However, 60 virtual users are sufficient enough for a web application that runs for sensor monitoring task in LAN.

V. Conclusion

Performance metrics of a web application developed for monitoring and control of a customized DAQ system have been measured. The measured data predicts that hardware resources of the server will be the bottleneck.

Acknowledgment

The authors are thankful to the All India Council of Technical Education (AICTE), Govt. of India for financial support towards the work (F. No. 8023/BOR/RID/RPS (NER)-84/2010-2011, 31st March 2011).

References

[1] J.D. Meier, C. Farre, P.Bansode, S.Barber, D. Rea, Performance Testing Guidance for Web Applications - patterns & practices, Microsoft Corporation, 2007.

[2] P.Glavich and Ch. Farrell, .NET Performance Testing and Optimization – The Complete Guide, *Simple Talk Publishing*, ISBN:978-1-906434-40-3, 2010.

[3] R. Jain, The Art of Computer Systems Performance Analysis – Techniques for Experimental Design, Measurement, Simulation and Modeling, Wiley India, 2012.

[4] V.A.F. Almeida and D.A. Menascé, "Capacity Planning: An Essential Tool for Managing Web Services", *IT Pro*, IEEE July - August 2002.

[5] A. Bora, M. K. Bhuyan and T. Bezboruah, " Investigations on Hierarchical Web service based on Java Technique," *Proceedings of the World Congress on Engineering*, Vol II, WCE 2013, July 3 - 5, 2013, London, U.K.

[6] Mercury Load runner tutorial - Mercury Roadrunner Quick Start (avalaible online: https://qageek.files.wordpress.com/2007/05/loadrunner_tutorial.pdf; accessed on 12.11.2014).

[7] S. Frigerio, L. Schenato, G. Bossi, M. Cavalli, M. Mantovani, G. Marcato, A. Pasuto, "A web-based platform for automatic and continuous landslide monitoring: The Rotolon (Eastern Italian Alps) case study," *J. of Computers & Geosciences* 63, 96–105, Elsevier, 2014.

[8] H. K. Singh, T. Bezboruah, "Design of a Remotely Accessible PC based Temperature Monitoring System," *ACEEE Int. J. on Communications*, Vol. 03, No. 01, March 2012.

[9] H. K.Singh, R. Gogoi, and T. Bezboruah. "Design Approach for a Web-Based Distributed Data Acquisition and & Control System," *International Conference on Internet Computing*. 2009.

[10] K.Singh and T.Bezboruah, "Microcontroller Based Frequency to Digital Converter for Interfacing Frequency Output Sensors", *Proc. of International Conference in Electronics Design Computer Automation and Verification* (EDCAV), NIT Meghalaya, Shillong, India, 2015.

[11] A. Z. Alkar and M. A. Karaca, "An Internet-Based Interactive Embedded Data-Acquisition System for Real-Time Applications," *IEEE Transactions on Instrumentation and Measurement*, Vol. 58, No. 3, 2009.

[12] K.Kalaitzakis, E. Koutroulis, V. Vlachos, "Development of a data acquisition system for remote monitoring of renewable energy systems," *Measurement* 34, pp.75–83., 2003.

[13] R. Arcidiacono et.al., "HYPERDAQ – where data acquisition meets the web," proc. of *10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems.* Geneva, 10 - 14 Oct 2005, pp.1-6, 2005.

[14] V. Hrushal *et. al,* "Distributed Web-based Measurement System," proc. of IEEE Workshop on *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pp. 355-358, 5-7 September 2005, Sofia, Bulgaria.

[15] B. R.Kumar, K. Sridharan, and K. Srinivasan, "The Design and Development of a Web-Based Data Acquisition System," *IEEE Transactions on Instrumentation and Measurement*, Vol. 51, No. 3, p.427, 2002.

[16] M. K. Mahmood1, F.M. Al-Naima , "An Internet Based Distributed Control Systems: A Case Study of Oil Refineries," *J. of Energy and Power Engineering*, 3, 310-316, Scientific Research, 2011.

[17] I.-Ch. Shen , Yi-Sh. Huang, M.-Sh. Young & K.-N. Huang, "Implementation of an Internet-Based Data Acquisition System Prototype for Drugs Storage," *Instrumentation Science &Technology*, 35:4, 437-451, Taylor & Francis, 2007.