Volume: 03, Issue: 04, Pages:1233-1239 (2012)

Analysis of Compute Vs Retrieve Intensive Web Applications and Its Impact On The Performance Of A Web Server

Syed Mutahar Aaqib
Department of Computer Science & IT, University of Jammu, Jammu-`180006, J&K (INDIA)
Email: syed.auqib@gmail.com
Dr. Lalitsen Sharma
Department of Computer Science & IT, University of Jammu, Jammu-`180006, J&K (INDIA)
Email: lalitsen.sharma@gmail.com

----------ABSTRACT----------
The World Wide Web (WWW) has undergone remarkable change over the past few years, placing substantially heavy load on Web servers. Today's web servers host web applications that demand high computational resources. Also some applications require heavy database retrieval processing, making server load even more critical. In this paper, performance of Apache web server running compute and retrieve-intensive web workloads is analyzed. Workload files implemented in three dynamic web programming technologies: PERL, PHP and Java Servlets are used with MySQL acting as a data source. Measurements are performed with the intent to analyze the impact of application workloads on the overall performance of the web server and determine which web technology yields better performance on Windows and Linux platforms. Experimental results depict that for both compute and retrieve intensive applications, PHP exhibits better performance than PERL and Java Servlets. A multiple linear regression model was also developed to predict the web server performance and to validate the experimental results. This regression model showed that for compute and retrieve intensive web applications, PHP exhibits better performance than Perl and Java Servlets.

Keywords - Web performance analysis, Web Servers, compute and retrieve intensive

Date of Submission: October 04, 2011 Date of Acceptance: December 02, 2011

1. INTRODUCTION

World Wide Web (WWW) has evolved from a static content-distribution medium into a dynamic and interactive medium, thereby facilitating the generation of dynamic contents "on-the-fly" and giving the ability to personalize web pages. But this advantage comes at a performance cost. Over the past several years there has been a substantial growth in the demand of computational resources for supporting increasingly sophisticated data processing applications. Contemporary applications normally fall in the two categories, compute intensive, which requires lot of computational resources and retrieve intensive, which demand lot of database processing [14,15,16].

In this paper, the impact of compute and retrieve intensive applications on the performance of a web server on Linux and Windows environments is examined. Applications were developed using three dynamic technologies: PHP, PERL and Java Servlets for both Linux and Windows platforms. For retrieve intensive applications, MySQL [13] was chosen data source as it is the most popular relational database management system and acts as a core component for both LAMP and WAMP web application software stack [17]. Apache 2.2 was chosen for this study as it is the most popular web server in the market and is used by most of the sites[1].

2. RELATED LITERATURE

In the recent years substantial amount of work has been done to analyze the performance of web servers on different architectures, but till date no work has been reported in the literature which studied this problem. Ramana et al. [1] performed application level benchmarking under Windows and Linux environments and found out that Apache on Linux yields better

[1] URL:www.news.netcraft.com/archives/category/web-serversurvey as accessed on 01 May 2011

performance than Apache on Windows. Mendes et al. [2] evaluated the performance of CGI, PERL, C++ and ASP on the Jigsaw and IIS running on Microsoft Windows NT. Their results depicted that small dynamic requests do not degrade the performance whereas large dynamic web applications have strong impact on the performance of a web server. Kothari et al. [3] compared the performance of Java Servlets, CGI and FastCGI and concluded that CGI and FastCGI are faster than Servlets. Checchet et al. [4] developed five different EJB applications to study the performance and scalability of JBOSS and JOnAs application servers. Checchet et al. [5] evaluated three dynamic content generation mechanisms: PHP, Java Servlets and EJB. Their results suggested that PHP is more efficient that Java Servlets. Also EJB yielded lower performance than both PHP and Servlets. Apte et al. [7] used stress testing techniques to evaluate various web technologies used for generating web content on dynamic web platforms. Main aim of their work was to determine how a simple and a complex application perform when implemented using different web programming technologies. Swales et al. [8] compared the performance of three dynamic Web programming technologies (JSP, ASP, ASP.NET) for multimedia image distribution. They tested applications that distribute multiple images from an Oracle 9i database unto web clients. Trent et al. [18] compared the performance of JSP and PHP using SPECweb2005benchmark on Apache and Lighttpd web servers. Their results suggested there's only 5-10% difference in the throughput and performance between these two technologies.

The work till date tries to determine which web technology performs better under non-intensive/simpler workloads. However, no work has been done to examine the impact of the web technologies with respect to the class of the workloads (compute/retrieve intensive). The aim of this paper is to fill this void and study the impact of the compute and retrieve intensive workloads on the overall performance of the web server. The results of this paper will help system administrators and web developers choose a web technology that is best suited for compute and retrieve intensive applications.

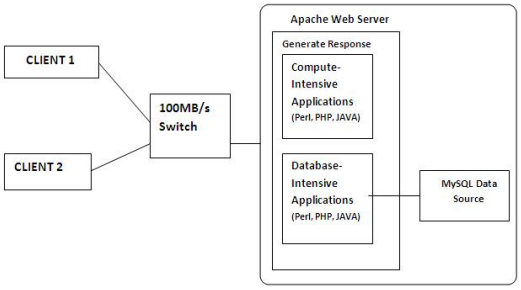
3. EXPERIMENTAL METHODOLOGY

The test-bed setup for the experiments is depicted in Fig. 1. It consists of two clients connected to a server via a 100Mbits/s Ethernet switch. The client machines are running Scientific Linux CERN 5 (2.6.18). Each machine has a single 2.0 GHz Intel processor with 1 GB of RAM and uses "RAM-disk" of 128 MB for collecting measurement statistics. The server machine in our test environment is Intel Pentium I5 machine, with 2 GB of RAM, running Scientific Linux CERN 5 (2.6.18) in case of Linux experiments and Windows 2003 server in case of Windows experiments. The hardware configuration is identical to that of the clients.

3.1. PERFORMANCE TUNING

The number of available file descriptors was increased from 1024 to 32,678 and the limit of the local port range was also increased. TCP TIME_WAIT recycling was enabled to free up sockets in a TIME_WAIT state more quickly, thus allowing clients to generate and sustain high request rate. Also, all the non-essential processes and services on the server as well as client machines were disabled. Also the web server was restarted before and after each experiment.

Figure 1: Test-bed for the experiments.



3.2. CLIENT WORKLOAD GENERATOR

The httperf [9] is an open source benchmark developed by David Mosberger at Hewlett-Packard Research Labs. The httperf benchmark is a flexible HTTP client that requests a file from a web server multiple times and for number of parallel threads and then prints out detailed statistics. Its source code was modified in order to print the server response rate information more frequently. Thus the output of the httperf provides information about TCP connection rate, HTTP request rate and HTTP reply rates after every one second during the experiments.

The server software used in the experiments is Apache 2.2. a public domain web server, running in a stand-alone mode on both Linux and Windows platform. Arlitt [11] and Grottke et al. [12] in their work suggested that the two configuration parameters for Apache web server, MaxRequestsPerChild and MaxClients should be set to 0 and 250 respectively. Based on this insight, in the main experiments the Apache web server was tuned by setting MaxClients MaxRequestsPerChild to these values.

3.3. VALIDATION OF THE TEST ENVIRONMENT

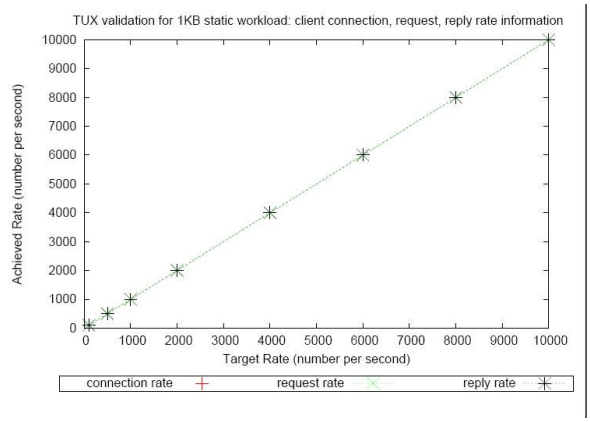
In order to validate the request generation capabilities of the test-bed, TUX web server [10] was used. The purpose of this validation was to show that clients can generate and

sustain high amount of requests rates during the experiments, possibly large enough to saturate the web server, and thus are not a bottleneck.

These tests were performed for a 1KB static file. Figure 2 shows the result for this experiment. This figure has three sets of data plotted which includes average number of TCP connections, average rate of HTTP requests and the average number of HTTP responses.

The results depicted that clients were able to generate and sustain workload of 10,000 requests per second for static 1kb file. Also the server platform and the network could support up to 10,000 responses per second for a static 1 KB file. Thus, the achieved response rates lower than these in the main experiments would indicate a bottleneck related to the particular server software technology being tested.

Figure 2: Experiment involving TUX web-server for validation of test-bed.



3.4. EXPERIMENTAL DESIGN

The metrics viz: TCP connection rate, HTTP request rate, HTTP reply rate, HTTP reply time were chosen in the experiments.

Main experiment was designed for two types of workloads, compute intensive and retrieve intensive. These workload were implemented in PHP, PERL and Java Servlets and were tested on both Windows and Linux environments. In case of compute intensive workloads, two recursive functions, one for Fibonacci series and other for Ackermann function were used with varying values of *n*. For these compute intensive workloads, requests to the server were sent to the server based on the request rate as shown in Table 1.

For retrieve intensive workloads, two test applications were used. One sends MySQL connection establishment 'connect' calls to the server and then tears down the connection and the other one connects to the MySQL server through a "connect" call and then performs a SELECT command on a database tables. These test applications were developed using PERL, PHP and Java Servlet technologies, for both Linux and Windows environments. Request rates for this experiment are shown in TABLE 2.

4. RESULTS AND DISCUSSIONS

This section presents the results of all the main experiments. The first and second part of this section discusses the results of compute intensive workloads and retrieve intensive workloads respectively.

TABLE 1: Factors and levels for compute intensive experiments

Type	Factor	Level	Request rate	Implementation
Compute intensive workload	Fibonacci Number	Small(n) = 10 Large(n) =50	100,200,500,700,1000, 1500,2000,2500,3000 100,200,500,700,1000, 1500,2000,2500,3000	PHP/ PERL/ JAVA
	Ackermann function	Small(n,m) = 1,3 Large(n,m) = 3,5	10,20,40,100,150,200, 500 10,20,40,80,100,200, 250	PHP/ PERL/ JAVA

TABLE 2: Factors and levels for retrieve/database intensive experiments.

Type	Factor	Request rate/sec	Implementation
Database intensive workload	MySQL 'connect' workload	100,200,500,1000,1500,2000,2500,3000	PHP/ PERL/ JAVA
	MySQL 'connect'+ 'SELECT'	100,200,500,1000,1500,2000,2500,3000	PHP/ PERL/ JAVA

In addition, we analyze the performance of PERL, PHP and JAVA server technologies on Linux and Windows platforms for above said workloads.

4.1. COMPUTE INTENSIVE WORKLOADS.

Figure 3 presents the results of the experiment involving a recursive implementation of Fibonacci number series with a considerable small value for *n*. This workload was not so much compute-intensive compared to the workload presented in the next part of this section and doesn't fall in the category of compute intensive function. Nevertheless, results depicted that all the implementations of this workload perform better on Linux than on Windows environment and PERL leads the achieved response rate on Linux platform by 1935.5 response rate/sec for the target rate of 2000 requests followed by 1695.9, 1595.8 responses/sec for PHP and Java respectively. On windows platform, Perl also performs better than PHP and Java. Similar results were found for Fibonacci number series with large value of *n*. Figure 4 shows the results; here also Perl performs better than PHP and Java on both platforms. So for web applications where the demand of computation is not much, Perl can be a good option. Next experiment

involved a compute intensive recursive implementation of ackermann function A(m, n), Figure depicts the results for considerable small values of (*m, n*). As ackermann function[6] is a total computable function than fibonacci series, better insights can be gained from this experiment.

$$A(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1, 1) & \text{if } m>0 \text{ and } n=0 \\ A(m-1, A(m, n-1)) & \text{if } m>0 \text{ and } n>0 \end{cases}$$

Results of this experiment as shown in Fig 4,depict that PHP on Linux followed by PHP on Windows exhibit better performance that Java and Perl on both platforms. Also Java performs better than Perl in both environments.

Figure 3: Results of experiments for Fibonacci number workload Small(n)

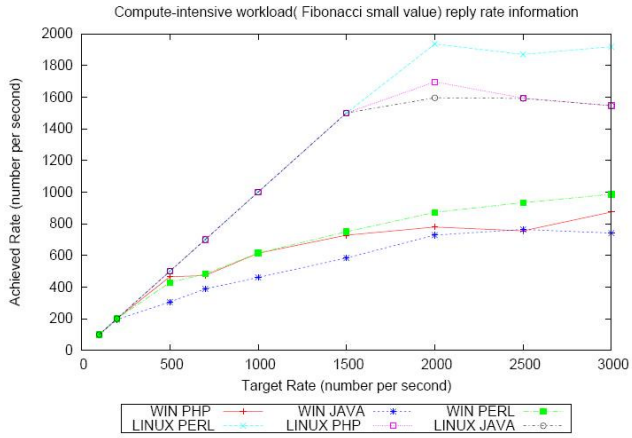


Figure 4: Results of experiments for Fibonacci number workload Large(n)

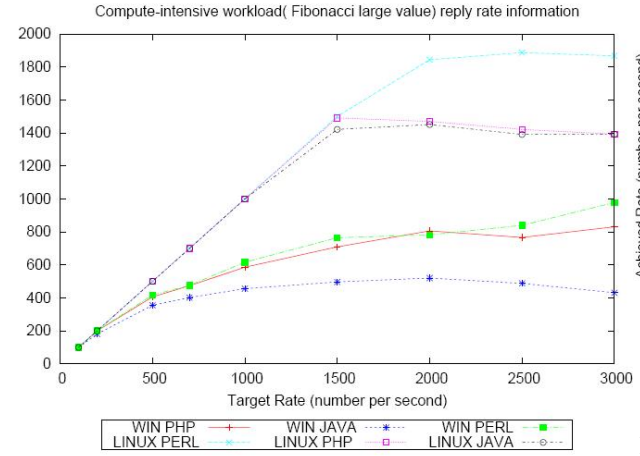


Figure 5 depicts the results for considerable large values of (*m, n*) for Ackermann function. Here as the value of (*m, n*) was somewhat large than that of the previous experiment, CPU became a bottleneck as the all the implementations of this workload were caught in deep recursions. In this experiment too, PHP exhibited better performance than the other technologies.

4.2 RETRIEVE INTENSIVE WORKLOADS

These set of experiments were performed to effectively analyze the performance of retrieve intensive workloads and to determine the affinity of various web technologies with Apache web server. Data source used is MySQL database on both Linux and Windows environments. Two set of experiments were conducted for this workload type. One set comprised of test application used to examine the impact of simultaneous "connect" calls on the MySQL database and other one performed connection establishment "connect" call followed by a SELECT operation on a predetermined set of tables.

Figure 4: Results of experiments for Ackermann series workload-small(*n,m*).

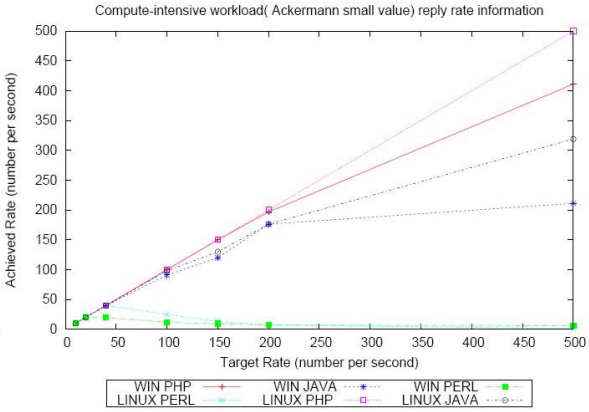
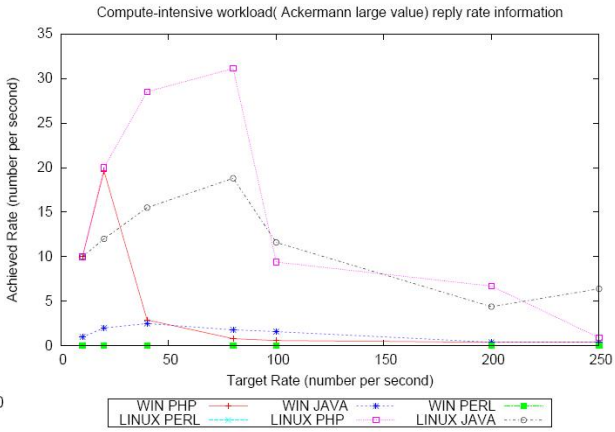


Figure 5: Results of experiments for Ackermann series workload-large (*n,m*).



Results of the first set of experiments are shown in Figure 6, Here also, PHP on Linux exhibited better performance than all of the other web technologies. Next, PHP on windows platform performed better than Java and Perl. Interestingly Perl which depicted highest performance in

case of Fibonacci number series exhibits lowest performance in case of both compute and retrieve intensive applications. In all of the above set of experiments, it was found that Apache web server performs better on Linux platform than on Windows. Ramana et al. [1] has reached to the same conclusion that overall Linux performs better than Windows. It was also found that the affinity of PHP with Apache web server on both the platforms, is better than Java and Perl for both compute and retrieve intensive applications.

5. PERFORMANCE MODEL

This section presents a multiple linear regression model [19] developed using the results obtained in the previous section to predict the performance of a web server for different workload types. A multiple linear regression model enables one to predict a response variable *y* as a function of *k* predictor variables x_1, x_2,…,x_k, using a linear model of the following form:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k + e$$

Where, b_0, b_1, b_2 ,…, b_k are k+1 fixed parameters and e is the error term.

In vector notation, the model is represented as:

$$y = Xb+e,$$

where, *y* is a column vector of *n* observed values of y,

X is a (*n*,*k*+*1*) matrix whose (i, j+1)th element $X_{i,j+1} = 1$ if j=0 else x_{ij},

b is a column vector with k+1 elements and

e is also a column vector of *n* error terms.

This model is based on 7 quantitative and categorical predictors: Target request rate, TCP connection rate, achieved request rate and achieved reply rate are quantitative predictors where as workload type, operating system and dynamic web technology (i.e. PHP, Perl, JAVA) are categorical predictors. The categorical predictor variable for workload-type takes two values: compute intensive, retrieve intensive, for operating system it also takes two values: SLC Linux and Windows and finally for dynamic web technology it takes three values: PHP, Perl and Java. Based on the multi linear regression model in [19], the coefficient of determination for this model is 0.9776. Thus the regression explains 97.76% of the variance of the reply rate in our model.

Figure 8 and Figure 9 depicts the measured and modeled reply rates using the multiple linear regression model for LAMP and WAMP experiments. Each graph in figure 8 and 9 represents the peak reply rates achieved for compute and retrieve intensive applications for a particular web technology i.e. PHP, Perl and Java. Figure 8 shows the comparative results of measured and modeled reply rates for experiments involving Linux platform, whereas Figure 9 shows the comparative results of measured and modeled reply rates for experiments involving Windows platform. These results show that our model predicts the performance of compute and retrieve intensive application for all three technologies i.e. PHP. Perl and JAVA within 8- 10% of the measured values. It is thus assumed that differences of 8-10% in predictions made by the multiple linear regression model are significant. The model thus validates the outcome of the experiments, which clearly shows that, for compute and retrieve intensive application PHP is a better choice compared to Perl and Java.

Figure 6: Results of experiments for first retrieve intensive workload.

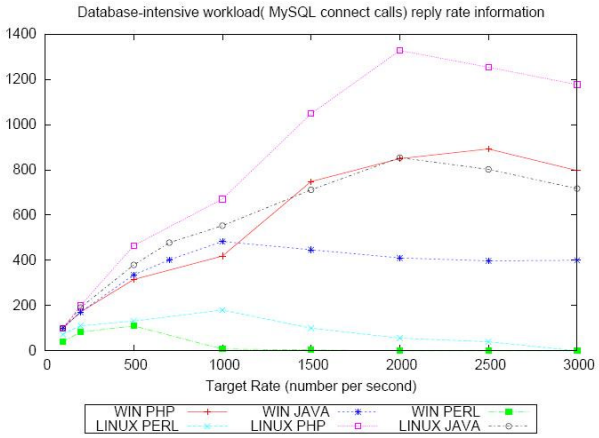
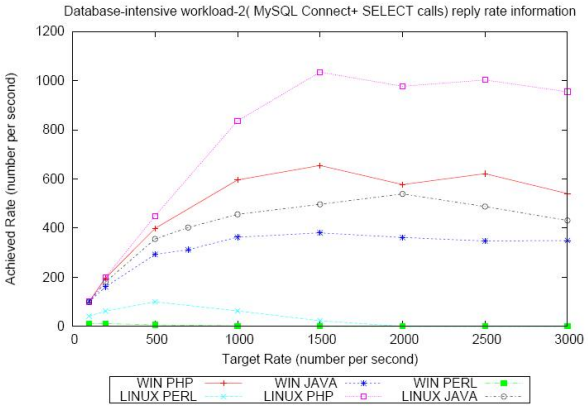


Figure 7: Results of experiments for second retrieve intensive workload.



6. CONCLUSION

In this paper, the impact of compute and retrieve intensive applications on the performance of a web server on Linux and Windows platforms is analyzed. Three web technologies viz. PHP, PERL and JAVA were used

to develop the test applications for both classes of experiments. Results were validated by repeating the same experiment under same environment multiple times and taking the mean values. Results revealed that for trivial applications with no large computational requirements, PERL performs better than the other two technologies. But in case of compute and retrieve intensive applications, it was found that PHP on Linux exhibits better performance than Java and Perl. A multiple linear regression model was also developed to predict the web server performance and to validate the experimental results. In other words, as request rates increases, applications developed using PHP are more efficient because its affinity with MySQL is more as compared to PERL and JAVA. This regression model also showed that for compute and retrieve intensive web applications, the performance of PHP is better than that of Perl and Java Servlets. Finally it was concluded that LAMP is a reliable and effective platform than WAMP in terms of performance.

Figure 8: Target rate/Achieved reply rate. Performance predicted by our multiple linear regression model for experiments involving Linux platform.

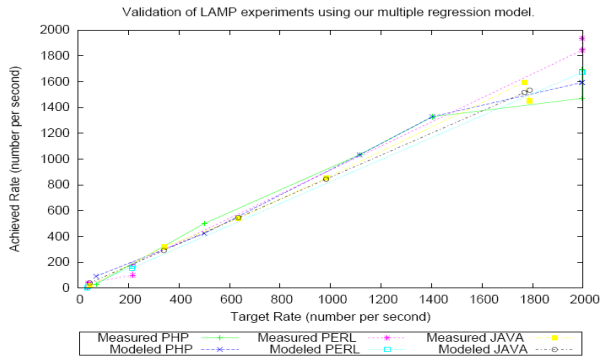
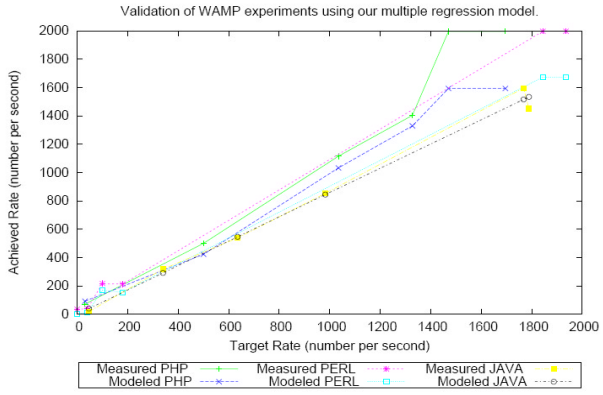


Figure 9: Target rate/Achieved reply rate. Performance predicted by our multiple linear regression model for experiments involving Windows platform.



ACKNOWLEDGMENTS

The authors are thankful to Prof. Devanand, Head, Department of Computer Science and IT, University of Jammu, for his kind support.

REFERENCES

[1] Ramana U. V., Prabhakar, T. V., Some Experiments with the Performance of LAMP Architecture. *Fifth International Conference on Computer and Information Technology*, 2005, 916-920.

[2] Mendes M. A., Almeida V. A., Analyzing the impact of dynamic pages on the performance of web servers. *In Proceedings of the Computer Measurement Group Conference,* Anaheim, CA, 1998, 539-547.

[3] Kothari B., Claypool M., Dynamic Web pages: performance impact on Web servers. *Internet Research, 11*(1) 2001, 18–25.

[4] Checchet E., Marguerite J. and Zwanepoel W., Performance and scalability of EJB applications. *Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2002.

[5] Checchet E., Chanda A., Elnikety S., Marguerite J., and Zwaenepoel W., Performance comparison of middleware architectures for generating dynamic Web content. *In Proceedings of the ACM/IFIP/ USENIX International Middleware Conference* Brazil, 2003, 16-20.

[6] Wichmann B. A.: How to call procedures, or second thoughts on Ackermann's function. *Software: Practice and experience 7(3)*, 1977, 317–329.

[7] Apte V., Hansen T., and Reeser, P., Performance comparison of dynamic web platforms. *Computer Communications 26*(8), 2003, 888–898.

[8] Swales D., Sewry D., and Terzoli A., A Performance Comparison of Web Development Technologies to Distribute Multimedia across an Intranet. *In Proceedings of Southern African Tele communication Networks and Applications Conference (SATNAC)*, 2003.

[9] Mosberger D. Jin T., httperf: A Tool for Measuring Web Server Performance. *The First Workshop on Internet Server Performance,* Madison, WI, 1998, 59-67.

[10] Lever C., Eriksen M., and Molloy S.: An analysis of the TUX web server. *Technical report, University of Michigan,* 2000, 00-8

[11] Grottke M., Li L., Vaidyanathan K., and Trivedi K.S.: Analysis of software aging in a web server. *IEEE Transactions on Reliability, 55*(3) 2006, 411-420.

[12] Arlitt M., Williamson C., Understanding Web server configuration issues. *Software: Practice and Experience. 34*(2), 2004, 163–186.

[13] MySQL AB. MySQL: The World's Most Popular Open Source Database. www.mysql.org. 2005.

[14] Che S., Li J., Lach J., and Skadron K., Accelerating compute intensive applications with GPUs and FPGAs. *Proc. of the 6th IEEE Symposium on Application Specific Processors*, 2008.

[15] Gorton, Greenfield P., Szalay A., and Williams R., Data-Intensive Computing in the 21st Century. *IEEE Computer, (41)* 4, 2008, 30–32.

[16] Apparao, P., Iyer, R., Zhang, X., Newell, D., Adelmeyer T., Characterization & analysis of a server consolidation benchmark. *Proceedings of the fourth ACM SIGPLAN/SIGOPSI international Conference on Virtual Execution Environments*, USA, 2008, 21–30.

[17] Dougherty D., LAMP: The Open Source Web Platform. www.onlamp.com/pub/a/onlamp/2001/01/25/lamp.html, 2001 as accessed on May 2011.

[18] Trent S., Tatsubori M., Suzumura T., Tozawa A., and Onodera T., Performance comparison of PHP and JSP as server-side scripting languages. *In Proc. 9th International Middleware Conference. ACM/IFIP/USENIX, Springer, 1*(5), Belgium, 2008, 164–182.

[19] Jain R., *The Art of Computer System Performance Analysis*, (John Wiley & Sons Inc. 1991).

Authors Biography

Syed Mutahar Aaqib is a research scholar in the Department of Computer Science & IT, University of Jammu. His research interests are in the field of performance and scalability analysis of web servers.

Dr. Lalitsen Sharma obtained his PhD degree from Guru Nanak Dev University, Amritsar. He is working as an Associate Professor in the Department of Computer Science & IT, University of Jammu. His research interests are in the field of web services and 'Network Applications' Security.