Some Experiments with the Performance of LAMP Architecture

UV Ramana
Veritas Software
TV Prabhakar
IIT Kanpur
tvp@iitk.ac.in

Abstract

Measurements are very useful to gauge the actual performance of various architectures and their components. In this paper we investigate the performance of the LAMP(Linux, Apache, MySQL, PHP) architecture and MySQL and PHP components. We build a web-site using LAMP and measure the application level performance. We use "measurements as a means" to improve the performance of the website. We then investigate the performance of the application when ported to Windows with running IIS and Apache with MySQL and PHP.

Keywords: LAMP, performance, software architecture

1. Introduction.

A large number of web-sites are built using *PHP* and *MySQL* on the *Linux* platform with *Apache* as the web server. This combination is known as the LAMP architecture. Apache, PHP and MySQL are also available on the Windows platform giving rise to combinations like WAMP (Windows, Apache, MySQL, PHP) and WIMP (where the IIS webserver is used instead of Apache). In this paper we present the results we obtained while investigating some performance issues related to MySQL and PHP and an application built with this architecture.

We first investigate the performance of PHP and MYSQL modules independently. We then check the performance of an application built using these components. We then benchmark the application on WIMP and WAMP platforms.

2. Related Work

Even though LAMP is a very popular architecture there has been little work to characterize and benchmark the architecture, especially at an application level. But, there has been a substantial amount of work done to analyze the performance of some other Web applications. Emmanuel Cecchet et al [1] compared the performance of LAMP architecture with the performance of EJBs and the performance of Java servlets. They performed the comparison based on two benchmarks, RUBiS [2] and RUBBoS [3]. The paper [4] talks about the benchmarks that they developed to perform the comparison. The bottleneck characterization of the work they have done is presented in [5]. The work presented in [6] discusses the performance and scalability of the EJB applications, by performing experiments with different types of EJBs (session, entity etc..) on two different open source EJB application servers (JBOSS and JOnAs).

C. D. Murta, J. M. Almeida and V. A. F. Almeida [7] presented a performance analysis of WWW Server, using WebStone benchmark, in the early days of the inception of the World Wide Web. Arun Iyengar et al. [8] presented a performance analysis of Web server under high CPU loads. In their work Y. Hu, A. Nanda, and Q. Yang [9] analyzed the performance of Apache Web server on a uniprocessor machine and a 4-CPU Symmetric Multi-Processor (SMP) machine. Vsevold V. Panteleenko and Vincent W. Freeh [10] analyzed the Web server performance by simulating Wide Area Network (WAN) conditions.

There has been some work in the area of client emulators. The paper by Gaurav Banga and Peter Drushel [11] describes a method to generate bursty traffic that temporarily exceeds the capacity of the Web server. It also describes the problems that one faces while measuring the Web server capacity. They have implemented a client emulator that can create bursty traffic, by using a *two process* architecture. There are quite a few open source client emulators like HTTPerf, EVE etc. Papers [12] and [13] describe the architecture of these client emulators.

Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology (CIT'05)
0-7695-2432-X/05 $20.00 © 2005 IEEE

Some work is done to compare the performance of a scripting language with that of a programming language. Ousterhout in his article [14] describes some of the results he got while comparing the performance of TCL/TK with C++, Java and MFC. The sqlbench suite provided by MySQL can be used to measure the performance of MySQL. The URL [15] talks about some of the results that are obtained by using the sqlbench suite.

3. PHP vs. C

In this section we present the performance of PHP scripting language in comparison to C. The comparison is done using two different criteria. The first criterion is to compare the total number of instructions that a PHP program can execute per second versus the number of instructions that a C program can execute per second. The second one is to compare the time taken to achieve certain functionality using PHP versus the time taken to achieve the same functionality using C.

3.1 Test Bed

The experiments described here are conducted on three different machines with different processor clock speeds: P-III 600 MHz, P-III 1 GHz and P-IV 2.8 GHz. All these machines had a Linux operating system, with kernel version 2.6.3, PHP version 4.3.4 and *gcc* version 3.3.2. All these systems have a RAM of 256 mb installed in them.

3.2 Findings

F1. C performs about four hundred times faster than PHP in case of compute intensive functions.

Table 1. PHP vs C – number of instructions per second

Computation	2.8 GHz Machine		
	Instructions per second		Ratio
	PHP	C	
Simple Instructions	2811479	1383200824	491
Function call	517227	232404139	449
Recursive Function	626159	230399042	367

From Table 1 we can observe that C is around 400 times faster than PHP in almost all the cases. We can also see that simple instructions are faster than function call instructions by a factor of around five in case of PHP program and by a factor of around six in case of C program.

Table 2. PHP vs C – Time to compute

Computation	2.8 GHz Machine		
	Time Taken		Ratio
	PHP	C	
Fibonacci Number	290.07	0.6806	426.15
File I/O	0.1195	0.0098	12.22
MySQL Access	0.00125	0.00133	0.942

From Table 2 it can be observed that in case of the Fibonacci program, which is computation intensive, C language performance, is nearly 400 times better than that of PHP. In case of File I/O the performance of C still dominates that of PHP, but this time only by a factor of 10. Finally, in case of MySQL data transfer even though C performed slightly better in two cases, we can say that performance of PHP and C are almost equal.

F2. Performance of MySQL can also be substantially improved by eliminating the connection overhead.

It is known that a database access has a connection overhead. The Figure below shows the graph between connection overhead and the processing time.

From the figure we can observe that the connection overhead is far more than the actual processing time. This implies that if we can avoid connection overhead, the performance can be improved substantially.

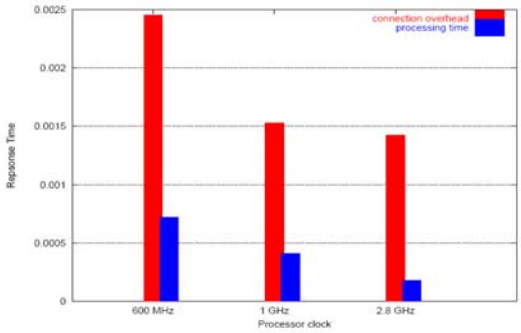

Figure 1.Connection overhead vs. processing time

4. Application benchmark

We investigate the performance of an application built with the LAMP. This section describes the response time and throughput of the *Gita Supersite* under different situations.

Gita Supersite (http://www.gitasupersite.iitk.ac.in) is a Web application developed to display the verses and commentary of *Bhagavadgita*(an ancient text containing the essence of Indian Philosophy) in all Indian languages. It can also display the verses in Roman script. The data(Sanskrit verses) is stored in the database in ISCII{Indian Script Code for Information Interchange} format and is converted into the font code of the desired Indian language to be displayed, when the request arrives.

The architecture of the *Gita Supersite* conforms to the trasitional LAMP architecture, except that it has an extra font conversion module. The font conversion module does the font conversion from the ISCII code to the ISFOC (Intelligence based Script FOnt Code) code of the respective language. It then returns the ISFOC data back to the PHP module, which does some formatting work on the data and returns it to the Apache Web server for sending it to the client.

4.1 Application Performance Measurement

The experimental setup consists of a client emulator which pumps in requests to the server running on a different machine and taking the measurements. Methods like sending requests from multiple machines, running multiple processes to pump the requests, running multiple threads and sending a request each, from all of the threads etc. do not give the desired results. In these methods, we have a fixed number of clients that are sending requests. Each client has to stop after sending a single request, receive the response and then send the request again. This method cannot overload the server, because as soon as the server reaches its limiting capacity, it will delay the processing of the client requests.

Gaurav Banga and Peter Drushel[11] described a way to overcome this problem. With this approach, one can send bursty traffic, but one cannot measure the Web server performance accurately. To avoid this problem a sophisticated client emulator has been designed and implemented. Detailed description can be seen in [16].

4.2 Test Bed

The machine on which the Web server and database server are running, is a P-IV 2.8 GHz processor, 512 MB RAM with Linux kernel version 2.6.3. The Web server used is Apache Web server version 2.0.48 with PHP module version 4.3.4 loaded into it. The database server is MySQL server version 4.0.18. The client emulator is running on another machine with the same configuration as the server.

4.3 Findings

F3. The maximum throughput that can be achieved for *Gita Supersite* using the LAMP architecture is around twenty six connections per second on a P-IV machine with 2.8GHz processor.

As seen is figure 2, the two lines indicate the throughput when there is a limit on the number of server processes and without a limit on the server processes.

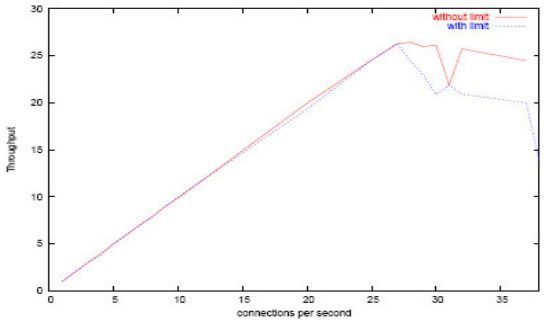

Figure 2. Throughput vs. connections per second

F4. We can improve the throughput of these applications by using a machine with more processor speed, as the CPU is the bottleneck in this case(Fig 3)

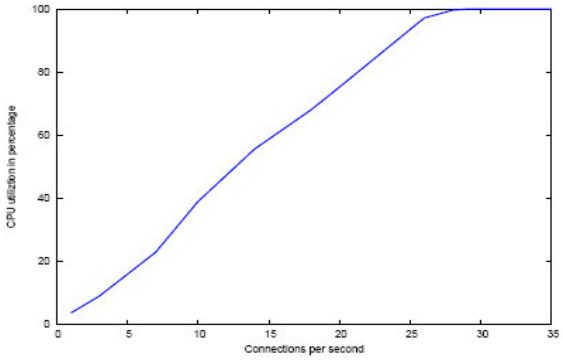

Figure 3. CPU Utilization

F5. The performance of the *Gita Supersite* can be improved by improving the performance of the font conversion module, as it is occupying the maximum portion of the processing time.

Figure 4 gives an indication of where the time is being spent while servicing a request. Most of the time is going into font conversion. That means we could improve the performance by performing the font conversion off-line.

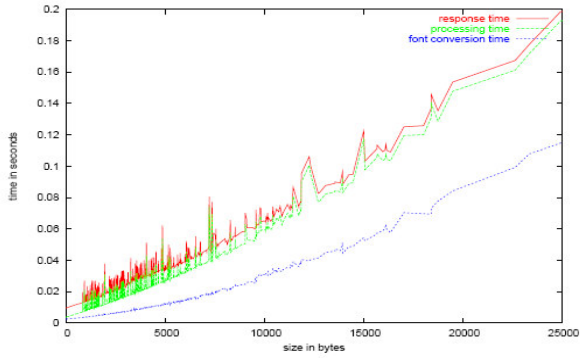

Figure 4. Break up the response time

We then modified the architecture by removing the dynamic font conversion step altogether. All the pages were converted into Devanagari off-line and store in that format in the database. At runtime, the PHP script retrieves the pages and servers it out. In this situation the architecture resembles more closely to a typical LAMP architecture.

F6. The maximum throughput that LAMP architecture can support is around 230 on a P-IV machine with 2.8GHz processor.

Figure 5 indicates the performance of the site as the number of connections goes up. Saturation is reached and the throughput drops around 230.

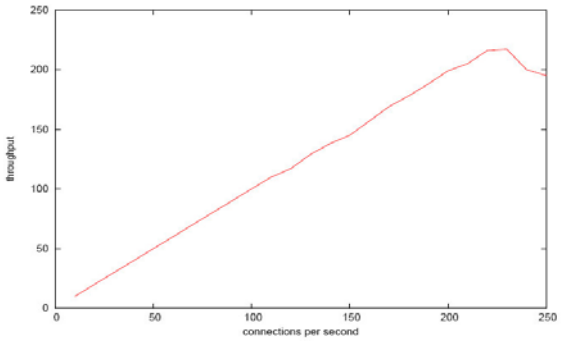

Figure 5. Throughput vs. connections (without font conversion)

F7. The reason for this saturation is the bottleneck in the CPU.

Figure 6 reflects the CPU utilization as the number of connections goes up. CPU reaches saturation level beyond 200 connections resulting in a drop in the throughput.

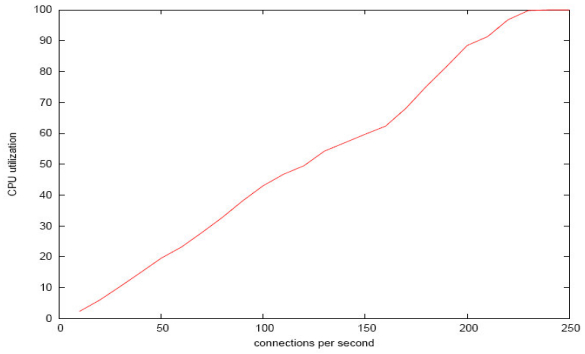

Figure 6. CPU Utilization vs. number of connections

5. LAMP, WAMP and WIMP

We now present the results that we have obtained while running the Gita supersite on the Windows platform with Apache web-server, MYSQL an PHP(WAMP) and the Windows platform with IIS web server, MySQL and PHP(WAMP)

F8. The response time of the WAMP(Windows, Apache, MySQL and PHP) and LAMP architectures are almost equal.

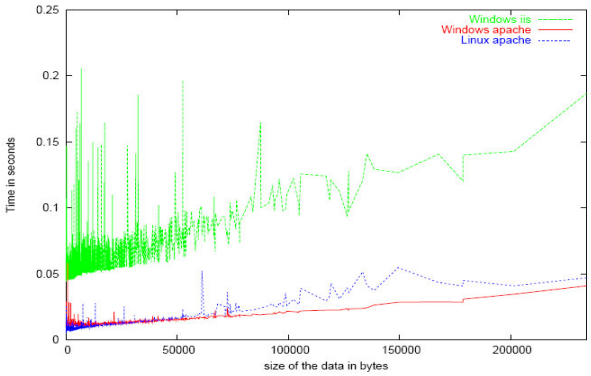

Figure 7. Response times of LAMP, WAMP, WIMP

Figure 7 shows the performance of the three configurations with a single connection. While LAMP and WAMP are almost similar, WIMP is trailing substantially.

6. Conclusions and Further Work

The objective of this work is to come with some numbers that will aid the architect while sizing an application. We could quantify the performance improvements that would one would accrue if one were to write parts of the code in C as against PHP, the impact of indices in MySQL, and the scaling that would occur with the processor speed. We also did some application level benchmarking and compared the performance of the application on Windows and Linux architectures. Linux with Apache very clearly outperforms Windows with IIS when the business logic is in PHP and the persistence is with MySQL. Windows with Apache, PHP and MySQL falls in between. More detailed results are available in[16].

Some of the future work that can be done is to perform this analysis on more number of applications and to increase the validity of the results obtained in this work. The effect of tuning some Linux kernel parameters, on the performance of the LAMP applications, also needs to be studied. The impact of applying architectural tactics [17] on the performance of an application will make an interesting study.

References

[1] Cecchet etal, Performance Comparison of Middleware Architectures for Generating Dynamic Web Content, 4th ACM /IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 16-20, 2003

[2] RUBiS: Rice University Bidding System URL: http://rubis.objectweb.org

[3]UBBoS: Rice University Bulletin Board System URL: http://rubbos.objectweb.org

[4] Amza etal, Specification and Implementation of Dynamic Web Site Benchmarks, IEEE 5th Annual WWC-5, Austin, TX, USA, November 2002.

[5] Amza etal, Bottleneck Characterization of Dynamic Web Site Benchmarks, Technical Report TR02-398, Rice University, January 2002.

[6] Cecchet etal, Performance and scalability of EJB applications, 17th , Oopsla 2002, Seattle, WA, USA, 4-8 November 2002.

[7] C. D. Murta, J. M. Almeida and V. A. F. Almeida, Performance Analysis of a WWW Server, XXII SEMISH, Recife, Brazil, August, 1996

[8] Iyengar etal, An Analysis of Web Server Performance, In Proceedings of the IEEE 1997, GLOBECOM '97, Phoenix, AZ, November 1997.

[9]Y. Hu, Nanda, and Yang., Measurement, analysis and performance improvement of the ApacheWeb server, Proc. of the 18th IEEE IPCCC, (Phoenix/ Scoottsdale, Arizona), pp. 261-267, Feb. 1999.

[10] Panteleenko and Freeh, Web Server Performance in a WAN Environment, Proc. of ICCCN, Dallas, TX, October, 2003.

[11] Banga and Drushel, Measuring the Capacity of a Webserver, USENIX Symposium on Internet Technologies and Systems, Monterey, California, December 1997

[12] Mosberger etal, httperf--A Tool for Measuring Web Server Performance}, Hewlett-Packard Research Labs, http://www.hpl.hp.com/personal/David_Mosberger/httperf.html.

[13] H. Jamjoom and K. Shin, Eve: A Scalable Network Client Emulator, University of Michigan Tech Report, CSE-TR-478-03

[14] Ousterhout, Scripting: Higher Level Programming for 21st Century, IEEE Computer Magazine, March 1998.

[15] MySQL Benchmarks, http:// dev. mysql. com/ tech-resources/benchmarks

[16] UV Ramana, Some experiments with the performance of LAMP architecture, Masters thesis, IIT Kanpur, 2004.

[17] Len Bass, Paul Clements and Rick Kazman, Software Architecture in Practice, Second Edition, SEI Series in Software Engineering, 2003