Analysis of Optimization Techniques to Improve User Response Time of Web Applications and Their Implementation for MOODLE

Priyanka Manchanda

Department of Computer Science and Information Technology,
Jaypee Institute of Information Technology,
Sector 128, Noida, UP - 201304, India
pmanchanda1992@gmail.com

Abstract. Analysis of six optimization techniques grouped under three categories (hardware, back-end, and front-end) is done to study the reduction in average user response time for Modular Object Oriented Dynamic Learning Environment (Moodle), a Learning Management System which is scripted in PHP5, runs on Apache web server and utilizes MySQL database software. Before the implementation of these techniques, performance analysis of Moodle is performed for varying number of concurrent users. The results obtained for each optimization technique are then reported in a tabular format. The maximum reduction in end user response time was achieved for hardware optimization which requires Moodle server and database to be installed on solid state disk.

Keywords: Optimization, SSD, HTTP, Moodle, Nginx, caching, DNS.

1 Introduction

The Internet has seen a significant growth of web based applications over the last few years. These have now become an inseparable part of numerous industries like airline, banking, business, computer, education, financial services, healthcare, publishing and telecommunications. They are preferred because of their zero installation time (as they run on a browser), availability of centralised data, their global reach, and their availability (24 hours a day, 7 days a week). According to [1], in June 2011 an average US user spent 74 minutes a day using web applications as compared to 64 minutes a day in June 2010.

In current scenario, improvement in the user response time is the most important issue for enhancing the performance of web applications. With reference to [2], a delay of one second in the performance of web applications can impact customer satisfaction by up to 16%.

Web applications make use of a wide range of technologies including JavaScript, Apache, CSS, HTML, MySQL, PHP and protocols like HTTP headers. Optimizing the way they use these technologies can significantly improve user response time. Furthermore, the browser and hardware capabilities can be employed to reduce the user response time.

Many research groups and authors have addressed this problem and reported their solutions. These include teams such as Yahoo Exceptional Performance Team [3], book authors [4] and research papers [5].

In this contribution, six optimization techniques grouped under three categories are analysed. Further, implementation of these six techniques is done for the Modular Object Oriented Distance Learning Environment (Moodle) [6]. The efficiency of these techniques is studied by comparing the original and the improved average user response time.

2 Performance Analysis of Moodle

Performance Analysis for Varying Number of Concurrent Users

Moodle is a free source Learning Management System (LMS) which is used by thousands of educational institutions around the world to provide an organized interface for e-learning. As of June 2013, it has 83059 currently active sites that have been registered from 236 countries [7]. Moodle LMS is written in PHP and uses XHTML 1.0 Strict, CSS level 2 and JavaScript for its web user interface[5].

With reference to [8], it has been reported that Moodle can support 50 concurrent users for every 1GB RAM. An experiment was performed to verify this result.

Experimental Setup
The experiment was perfomed on a machine with the following specifications:

Hardware: Intel® Core^{TM}i5-2310 CPU @2.90GHz x 4 processor, 8GB Hard disk and 1GB RAM.
Operating system: Ubuntu 12.10
Web server: Apache v2.2.22 and PHP v5.4.6 for Moodle v2.5 for Ubuntu 12.10
Database software: MySQL v5.5.31 for Ubuntu 12.10

- The experiment was performed using Apache JMeter 2.9, an open source load testing tool by the Apache Software Foundation [9].
- The test script was generated by using the JMeter Script Generator plugin for Moodle by James Brisland [10].
- The bandwidth of the network was set to 1024 kbps (1 Mbps) using JMeter.
- The load testing of Moodle was done for a chat activity.
- The sequence of pages visited on Moodle was :

  Login to site -> View Course -> View Chat page -> View Chat window -> Initialize Chat -> Initialize Initial Update

- After initializing chat the following tasks were performed five times for each concurrent user : Post Chat Message -> Initialize Update
- To test the performance of Moodle in the worst case scenario the ramp-up period, that is the amount of time for creating the total number of threads, was set to zero so as to ensure immediate creation of all the threads by JMeter.

Table 1. Average Response Time and Throughput for load testing Moodle on 1GB RAM and 8GB HARD DISK

No. of Concurrent Users	***Average Response Time(s)***	***Throughput (per m)***
10	3.671	147.6
20	8.874	129
30	15.303	99.6
40	129.786	16.8
49	243.469	11.4
50	364.480	7.8
51	Database Overload	Database Overload

While load testing Moodle for 51 concurrent users, it was observed that the connection to the database was aborted due to database overload and the testing process was killed by JMeter.

3 Hardware Optimization

Employing Solid State Disk

The performance of the web applications can be highly enhanced by using a solid state disk drive to reduce the latency of the input and output operations carried out by the server.

A Solid State Disk, or SSD is a high performance plug and play data storage device which uses integrated circuit assemblies as memory to store data persistently [11]. An SSD incorporates solid state flash memory and emulates a hard disk drive to store data [12]. However, unlike the traditional electromechanical disks like hard disk and floppy disks, an SSD is a flash-based and DRAM-based storage device which does not contain any moving parts [13].

An experiment was performed by replacing the Hard Disk Drive(HDD) of the Moodle Server with a 128GB Kingston Solid State Disk Drive.

Experimental Setup

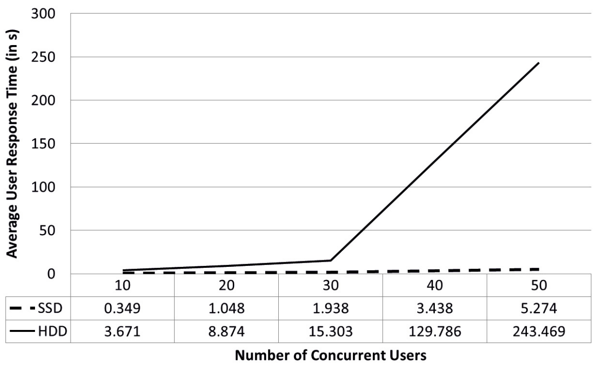
To conform to the experiment performed in section 2 and to compare the performance of Moodle on HDD vs. SSD, the space allocated to Moodle server and database collectively was 8GB of 128GB SSD and the RAM size was limited to 1GB. The experiment was performed on a machine with following specifications:

***Hardware:* Intel® Core$^{\text{TM}}$i5-2310 CPU @2.90GHz x 4 processor, 8GB Solid State disk and 1GB RAM.**
Operating system: Ubuntu 12.10
Web server: Apache v2.2.22 and PHP v5.4.6 for Moodle v2.5 for Ubuntu 12.10
Database software: MySQL v5.5.31 for Ubuntu 12.10
Bandwidth: 1024 Kbps (1 Mbps)

The experiment was performed for the chat activity mentioned in section 2 using Apache JMeter 2.9.

Table 2. Average User Response Time on HDD vs SSD(in s)

No. of concurrent users	Average Response Time on HDD(s)	Average Response Time on SSD (s)	Reduction in Response time %
10	3.671	0.349	90.49
20	8.874	1.048	88.19
30	15.303	1.938	87.34
40	129.786	3.438	97.35
50	364.480	5.274	97.83
60	Database Overload	5.97	-
70	Database Overload	6.492	-
80	Database Overload	8.009	-
90	Database Overload	8.085	-
100	Database Overload	9.797	-
110	Database Overload	13.759	-
120	Database Overload	16.828	-
130	Database Overload	22.991	-
140	Database Overload	30.187	-
150	Database Overload	36.119	-
151	Database Overload	39.141	-
152	Database Overload	Database Overload	-



	10	20	30	40	50
SSD	0.349	1.048	1.938	3.438	5.274
HDD	3.671	8.874	15.303	129.786	243.469

Fig. 1. Average user response time (in s) for Moodle on HDD vs. SSD

From Table 2, it is concluded that the number of concurrent users supported by Moodle installed on SSD for 1 GB RAM is increased to **151** as compared to **50** concurrent users for Moodle installed on HDD with 1 GB RAM. Also, there is a reduction of **87% to 98%** in average user response time after installing Moodle server and database on SSD.

4 Back-End Optimization

Switching to LNMP Stack from LAMP Stack

The Moodle web application runs on the LAMP stack which is a software bundle comprising of Linux based operating system, Apache HTTP server, MySQL database software and PHP object oriented scripting language. LNMP stack is almost similar to LAMP, except the change of web server from Apache to Nginx.

Apache is a process-based server, while nginx is an event-based asynchronous web server and is more scalable than Apache. In Apache, each simultaneous connection requires a thread which incurs significant overhead whereas nginx is event-driven and handles requests in a single (or at least, very few) threads [14].

The performance of Moodle or any web application that runs on Apache and frequently encounters heavy load, can be boosted by replacing Apache by Nginx. An experiment was performed to compare the performance of Moodle installed on Apache vs. Nginx web server.

Experimental Setup
Since it was observed in Section 3 that the performance of Moodle is highly enhanced by installing it on SSD, the experiment was performed on a machine with Moodle installed on 128 GB Solid State Disk and 4GB RAM.

All the other specifications (Operating system, Database software, Web server and Bandwidth) of the machine were kept same as in section 3. The experiment was performed using Apache BenchMark 2.4 [15] for Moodle's login page.

From Table 3, it is observed that there is a reduction of **24% to 34%** in average user response time after installing Moodle on Nginx v1.4.1 web server.

Table 3. Average User Response Time on Apache vs Nginx(in s) Web Server

No. of concurrent users	**Average Response Time on Apache(s)**	**Average Response Time on Nginx(s)**	**Reduction in Response time %**
50	2.209	1.652	25.22
100	4.505	3.359	25.43
150	6.098	4.630	24.07
200	8.192	5.408	33.98
250	10.729	7.156	33.30

5 Front-End Optimization

For any web application, only 10% to 20% of the end user response time is spent downloading the HTML document from the web server to the client's browser. The other 80% to 90% is spent in performing the front end operations, i.e., in downloading the other components of web page [4].

A set of specific rules for speeding up the front end operations carried by a web application is presented in Ref. [4]. Four of the most efficient techniques which showed significant reduction in user response time for Moodle Learning Management System are described in this section.

5.1 Browser Caching by Using Far Future Expires Header

Browsers and proxies use cache to reduce the number and size of the HTTP requests thereby speeding up the web applications. A first-time visitor may have to make several HTTP requests, but by using a Far Future Expires header the developer can significantly improve the performance of web applications for returning visitors. A server uses the Expires header in HTTP response to inform the client that it can use the current copy of a component until the specified time [4].

Moodle sends requests with an Expires Header which is set in past **(20th Aug 1969 09:23 GMT)**. An experiment was performed by changing it to future date of **16th Apr 2015 20:00 GMT**. Also max-age directive was used in Cache control header so as to set the cache expiration window to 10 years in future and the pragma header was unset to enable caching.

Given below are the lines which were added to the headers.conf file of Apache2 Web Server:

```
<FilesMatch".(ico|pdf|flv|jpg|jpeg|png|gif|js|css|swf|php|html)$">
Header set Expires "Thu, 16 Apr 2015 20:00:00 GMT" Header set
Cache-Control " max-age=315360000" Header unset Pragma
</FilesMatch>
```

The experimental setup is the same as section 3 and the experiment was performed using Apache Jmeter 2.9. From Table 4, it is observed that there is a reduction of **70% to 80%** in average user response time after implementing Far Future Expires Header Optimization Technique.

Table 4. Average user response time (in s) with and without caching for 10 iterations

No. of concurrent users	Average Response Time Without Expires Header(s) (no caching)	Average Response Time with Expires Header(s) (caching)	Reduction in Response time %
10	0.625	0.144	76.96
20	1.839	0.408	77.81
40	5.061	1.210	76.09
60	7.086	1.778	74.91
80	8.124	2.426	70.14
100	9.882	3.071	68.92

5.2 Reduce DNS Lookups

The Internet uses IP addresses to find webservers. Before establishing a network connection to a web server, the browser must resolve the hostname of the web server to an IP address by using Domain Name Systems (DNS). The latency introduced due to DNS lookups can be minimized if the DNS resolutions are cached by client's browser [4]. The response time for Moodle's login page of Institutional Moodle websites of 13 universities situated in six continents of the world was recorded for two cases: With DNS Cache and Without DNS Cache. The experiment was performed for a client located in IIT Bombay, India with 128GB SSD, 4GB RAM, Intel® Core^{TM}i5-2310 CPU @2.90GHz x 4 processor and 2 Mbps average download speed.

Table 5. Average user response time (in s) With and Without DNS Cache for 1 user

Continent	**Country**	**University**	**Response time With DNS Cache(s)**	**Response time Without DNS Cache(s)**	**Reduction in Response time(%)**
Asia	India	IIT, Bombay [16]	2.357	1.426	39.50
Asia	India	IIT, Madras [17]	2.516	1.612	35.93
Asia	Singapore	SIM University [18]	1.381	1.055	23.61
Asia	Japan	Sojo University, Kumamoto [19]	6.223	3.116	49.93
Europe	Spain	Graduate School of Management, Barcelona [20]	3.138	1.813	42.22
Europe	UK	University of Nottingham [21]	4.174	2.041	51.10
North America	US	UCLA, California [22]	4.600	3.657	20.50
South America	Argentina	Pontifical Catholic University of Argentina, Buenos Aires [23]	2.534	1.710	32.52
South America	Colombia	University of Grand Colombia, Bogot, D.C. [24]	2.341	1.438	38.57
Africa	Egypt	Oriflame University [25]	5.497	4.288	21.99
Africa	South Africa	Virtual Academy of South Africa [26]	4.936	2.588	47.57
Australia	Australia	Australian National University [27]	4.525	3.559	21.35
Australia	Australia	Monash University [28]	4.947	4.141	16.29

From Table 5 it is concluded that there is a reduction of **16% to 51% depending on the geographical location** of Moodle server, if the resolved hostname for a web page is found in DNS cache.

Another experiment was carried out on the same client to compare the performance of Moodle by changing the number of DNS cache entries, DNS cache expiration period and HTTP keep alive timeout for Mozilla Firefox 21.0 browser. The following three scenarios were tested for 100 iterations of Moodle's login page of Moodle websites of six universities situated in six continents of the world using iMacros 9.0 Firefox extension [29] and HttpFox addon for Firefox [30].

```
Scenario 1 (S1):
DNS Cache Entries = 20
DNS Cache Expiration Period = 60 seconds
HTTP Keep Alive Timeout = 115 seconds

Scenario 2 (S2):
DNS Cache Entries = 512
DNS Cache Expiration Period = 3600 seconds
HTTP Keep Alive Timeout = 115 seconds

Scenario 3 (S3):
DNS Cache Entries = 512
DNS Cache Expiration Period = 3600 seconds
HTTP Keep Alive Timeout = 0 second
```

Table 6. Average user response time (s) for 1 user, 100 iterations for above Scenarios

Continent	**University**	**Response time for S1**	**Response time for S2**	**Difference (s) between S1 & S2**	**Response time for S3**	**Difference(s) between S2 & S3**
North America	UCLA, USA [22]	173.984	**169.284**	4.7	178.69	9.406
Asia	IIT, Madras, India [17]	108.93	**105.677**	3.253	110.548	4.871
Australia	Australian National University [27]	347.361	**344.961**	2.400	354.336	9.375
Africa	Oriflame University, Egypt [25]	244.035	**240.246**	3.789	256.08	15.834
Europe	University of Nottingham, UK [21]	153.71	**150.213**	3.497	156.76	6.547
South America	University of Grand Colombia, Colombia [23]	142.241	**135.908**	6.333	146.763	10.855

From Table 6, it is observed that the end user response time is minimum under Scenario 2. Hence, it can be concluded that the performance of a web application can be enhanced by reducing DNS Lookups, which was achieved by:

- Increasing the number of DNS cache entries,
- Increasing DNS expiration period, and
- Using a Network that supports HTTP keep-alive mechanism

5.3 Gzip Components

Gzip compression of web pages can significantly minimize the latency introduced due to transfer of the web page files from web server to client's browser. Starting with HTTP/1.1, web clients indicate support for compression with the Accept-Encoding header in the HTTP request [4].

```
Accept-Encoding: gzip, deflate
```

After the web server sees this header, it compresses the response using one of the methods listed by the client. The web server uses the Content-Encoding header in the response to inform the client about the compressed response [4].

```
Content-Encoding: gzip
```

An experiment was performed on the client mentioned in section 5.2 for Moodle installed on the machine with specifications as mentioned in section 4 using Web Developer Extension for Mozilla Firefox 21.0 [31].

It is observed that Gzip compression reduces the response size by **75%-77%**.

Table 7. Response Size of Moodle pages with and without compression of components

Moodle Page	**No. of Files Requested**	**Response Size without Compression(KB)**	**Response size with Compression(KB)**	**Reduction in Response Size (%)**
Index	42	926	215	76.78
Login	13	597	138	76.88
View Course	42	804	187	76.74
View Forum (with 1 post)	41	802	187	76.68
View Blog (with 10 posts)	35	889	218	75.48
View Calendar	42	861	207	75.96
View Participants (20 per page)	40	806	188	76.67
1 page quiz with 5 questions	49	847	198	76.62
View Assignments	43	804	187	76.74

5.4 Optimize AJAX

AJAX (Asynchronous JavaScript and XML) is a collection of technologies, primarily JavaScript, CSS, DOM and asynchronous data retreival which is used to exchange data with a server, and update parts of a web page - without reloading the whole page. Though AJAX allows the server to provide instantaneous feedback to the user, it does not guarantee that the user won't have to wait for the asynchronous JavaScript and XML responses. The performance of the web application can be improved by optimizing the AJAX requests. The techniques mentioned in section 5.1, 5.2 and 5.3 are collectively used to optimize the ajax components of Moodle.

The AJAX components are made cacheable by modifying the expires header which is defined in **OutputRenderers.php** file located in **lib** directory of main Moodle directory. An experiment was performed on the client mentioned in section 5.2 for Moodle installed on the machine with specifications as mentioned in section 4 using Firebug Extension 1.11.4 for Mozilla Firefox 21.0 [32].

Modifications made to expires header in OutputRenderers.php file

```
Default:
@header('Expires: Sun, 28 Dec 1997 09:32:45 GMT'); Line 3345

Modified:
@header('Expires: Sun, 28 Dec 2020 09:32:45 GMT'); Line 3345
```

There is a reduction of an average of 23.54% in user response time after optimizing the AJAX components.

Table 8. Average User Response Time (s) before and after optimizing AJAX

Activity	**Response Time before optimizing AJAX(s)**	**Response Time after optimizing AJAX(s)**	**Reduction in Response Time(%)**
Drag and Drop Sections	309	227	26.54
Drag and Drop Activities	2.17	1.62	25.35
Drag and Drop Files	201	175	12.94
Drag and Drop Blocks	440	354	19.55
AJAX Chat Box	36	24	33.33
Average			**23.54**

6 Conclusion

In this presented paper, six methods to optimize web applications have been analysed and tested for Moodle LMS. These methods can be further used to optimize other essential web applications including webmail, online retail sales, online auctions, wikis and e-learning. It is observed that Moodle shows faster response time under heavy traffic, if it is loaded on a solid state disk. This technique can be used to scale high traffic web applications.

The caching mechanism can be used by the client's browser to optimize the front-end operations that can reduce the end-user response time by up to 80%. This mechanism can be used for content that changes infrequently, that is, application's static assets like graphics, style sheets and scripts. In addition to application's static assets, DNS resolutions can be cached by client's browser and can reduce the end-user response time by up to 50%.

The six web optimization techniques discussed in this paper were successfully tested for the Moodle LMS which showed a maximum reduction of 98% in average user response time by using the hardware optimization technique used in (Section 3). These best practices can be further applied to a novel or existing web application to improve its performance by reducing end user response time and thereby increasing the number of concurrent users and throughput.

Acknowledgement. The author would like to thank the members of Department of Computer Science, Indian Institute of Technology, Bombay, India for their kind support and encouragement.

References

1. French, C.N.: Mobile Apps Put the Web in Their Rear-view Mirror (June 20, 2011), from Flurry Blog: `http://blog.flurry.com/bid/63907/Mobile-Apps-Put-the-Web-in-Their-Rear-view-Mirror` (accessed June 4, 2013)
2. Borg, A.: Web Site Performance: When Seconds Count(December 17, 2009), from technewsworld.com: `http://www.techhnewsworld.com/story/68918.html` (accessed June 4, 2013)
3. Yahoo Exceptional Performance Team, `http://developer.yahoo.com/performance`
4. Souders, S.: High Performance Web Sites. O'Reilly Media (2007)
5. Horat, D., Arencibia, A.Q.: Web Applications: A Proposal to Improve Response Time and Its Application to MOODLE. In: Moreno-Díaz, R., Pichler, F., Arencibia, A.Q. (eds.) EUROCAST 2009. LNCS, vol. 5717, pp. 218–225. Springer, Heidelberg (2009)
6. Project Moodle, `http://moodle.org`
7. Moodle Statistiscs, `https://moodle.org/stats/`
8. Joint Information Systems Committee, Regional Support Centre, West Midlands Moodle Wiki, `http://wiki.rscwmsystems.org.uk/index.php/Moodle`
9. Apache JMeter, `http://jmeter.apache.org/`
10. Moodle-jmeter-script-generator, `https://github.com/kabalin/moodle-jmeter-script-generator`

11. Solid State Drive by Wikipedia.org, `http://en.wikipedia.org/wiki/Solid-state_drive`
12. Wong, G.: SSD Market Overview. In: Micheloni, R., Marelli, A., Eshghi, K. (eds.) Inside Solid State Drives (SSDs). Springer Series in Advanced Microelectronics, vol. 37, pp. 1–18. Springer Science+Business Media, Dordrecht (2013)
13. Martin, D.: Is SSD Technology Ready for the Enterprise? (January 14, 2009), from Wikibon.com: `http://wikibon.org/wiki/v/Is_SSD_Technology_Ready_for_the_Enterprise?` (accessed June 11, 2013)
14. Apache vs nginx, `http://www.wikivs.com/wiki/Apache_vs_nginx`
15. ab - Apache HTTP server benchmarking tool, `http://httpd.apache.org/docs/current/programs/ab.html`
16. Indian Institute of Technology, Bombay : Moodle, `https://moodle.iitb.ac.in/login/index.php`
17. Indian Institute of Technology, Madras : Moodle, `http://www.cse.iitm.ac.in/moodle/`
18. Singapore Institute of Management University, Singapore : Moodle, `http://cp.unisim.edu.sg/moodle/`
19. Sojo University, Nishi-ku, Kumamoto, Japan : Moodle, `http://md.ed.sojo-u.ac.jp/`
20. Graduate School of Management, Barcelona, Spain : Moodle, `http://moodle.gsmbarcelona.eu/`
21. University of Nottingham, Nottingham, UK : Moodle, `https://moodle.nottingham.ac.uk/login/index.php`
22. University of California, Los Angeles : Physics and Astronomy Dept. Moodle, `http://reserve.pna.ucla.edu/`
23. Pontifical Catholic University of Argentina, Puerto Madero, Buenos Aires, Argentina : LirWeb Moodle, `http://www.lirweb.com.ar/`
24. University of Grand Colombia, Bogot, D.C., Colombia : Moodle, `http://virtual.ulagrancolombia.edu.co/login/index.php`
25. Oriflame University : Moodle, `http://www.oriflame-eg.com/uni/moodle/`
26. Virtual Academy of South Africa: Moodle, `http://www.virtualacademy.co.za/`
27. Australian National University: Moodle, `http://moodle.anu.edu.au/`
28. Monash University: Moodle, `http://moodle.vle.monash.edu/`
29. iMacros, `http://www.iopus.com/iMacros/`
30. HttpFox Addon for Mozilla Firefox 21.0, `https://addons.mozilla.org/en-us/firefox/addon/httpfox/`
31. Web Developer Extension for Mozilla Firefox 21.0, `https://addons.mozilla.org/en-US/firefox/addon/web-developer/`
32. Firebug 1.11.4 Extension for Mozilla Firefox 21.0, `https://addons.mozilla.org/en-US/firefox/addon/firebug/`