Report on

Optimizing Moodle LMS For Improving User Response Time

Submitted in partial fulfilment of the requirements
For Summer Internship 2013

Guided By
Nagesh Karmali,
IIT BOMBAY

Members of Team-
Your Names alphabetically

Contents

Abstract

Our primary goal is to optimize moodle for fast user response time. This can be done by studying the database abstraction and schema of moodle.
(will add more)

Chapter 1

Introduction to Moodle LMS

Moodle is a Virtual Learning Environment (VLE). As of June 2013 it had a user base of 83,008 registered and verified sites, serving 70,696,570 users in 7.5+ million courses with 1.2+ million teachers.[1](https://moodle.org/stats)

1.1 MOODLE

Moodle means Modular Object Oriented Dynamic Learning Environment which was developed to help educators create online courses. It is a free source e-learning. It is also called as Learning Environment(LMS) or Virtual Learning Environment(VLE).It was developed by Martin Dougiamas and is in continual evolution. The first version of moodle was released on 20 August 2002 and the current stable version is Moodle 2.5. It can be installed on any system that supports PHP and a database.

1.2 Features

Moodle has several features considered typical of an e-learning platform, plus some original innovations (like its filtering system). Moodle is very similar to a learning management system. Moodle can be used in many types of environments such as in education, training and development, and business settings.
Some typical features of Moodle are:

- Assignment submission
- Discussion forum
- Files download
- Grading
- Moodle instant messages
- Online calendar
- Online news and announcement (College and course level)
- Online quiz
- Wiki

Developers can extend Moodle's modular construction by creating plugins for specific new functionality. Moodle's infrastructure supports many types of plugins.

- activities (including word and math games)
- resource types
- question types (multiple choice, true and false, fill in the blank, etc.)
- data field types (for the database activity)
- graphical themes
- authentication methods (can require username and password accessibility)
- enrollment methods
- content filters

Many freely available third-party Moodle plugins make use of this infrastructure.[2](modules and

plugins moodle.org)

Moodle users can use PHP to write and contribute new modules. Moodle's development has been assisted by the work of open source programmers.[3](moodle documentation moodle.org)This has contributed towards its rapid development and rapid bug fixes.

By default Moodle includes the TCPDF library that allows the generation of PDF documents from pages.

1.3 Installation

Users can install Moodle from source, but this requires more technical proficiency than other automated approaches such as installing from a Debian package, deploying a ready-to-use Turnkey Moodle appliance,using the Bitnami installer, or using an "one-click install" service such as Installatron.

Some free Moodle hosting providers allow educators to create Moodle-based online classes without installation or server knowledge. Some paid Moodle hosting providers provide value-added services like customization and content development.

Branch	Original release date	Current version	Current version release date	Support Model	Release notes
1.0	20 August 2002	1.0.9	30 May 2003	EOL	
1.1	29 August 2003	1.1.1	11 September 2003	EOL	
1.2	20 March 2004	1.2.1	25 March 2004	EOL	
1.3	25 May 2004	1.3.5	9 September 2004	EOL	
1.4	31 August 2004	1.4.5	7 May 2005	EOL	
1.5	5 June 2005	1.5.4	21 May 2006	EOL	
1.6	20 May 2006	1.6.9	28 January 2009	EOL	
1.7	7 November 2006	1.7.7	28 January 2009	EOL	
1.8	30 March 2007	1.8.14	3 December 2010	EOL	
1.9	3 March 2008	1.9.19	9 July 2012	EOL (Maintained from March 2008 to June 2012. Third-party extended support until December 2013)[24]	
2.0	24 November 2010	2.0.10	9 July 2012	EOL (Maintained from November 2010 to June 2012)	
2.1	1 June 2011	2.1.10	14 January 2013	EOL (Maintained from June 2011 to December 2012)	
2.2	5 December 2011	2.2.10	14 May 2013	**Active** (Maintained from December 2011 to June 2013)	
2.3	25 June 2012	2.3.7	14 May 2013	**Active** (Maintained from June 2012 to December 2013)	
2.4	3 December 2012	2.4.4	14 May 2013	**Active** (Maintained from December 2012 to June 2014)	
2.5	14 May 2013	2.5		**Active** (Maintained from May 2013 to November 2014)	

Legend: Old version | Older version, still supported | **Latest version**

Moodle Versions

2. Database

Moodle is not a single complex application. It is an aggregation of differentplugin’s. The scripts in moodle are written inPHP.So only some plugin’s are object

oriented. There are two layers in moodle to separate presentation from business logic. Outer is theme and controls like visual aspects of moodle interface then the renderer classes which generate html output from the data supplied by transaction script and domain model.Butneither PHP nor moodle architecture doesn't have clear separation of UI layer. Every interaction has logic in it. Transaction script organizesall the script as a single procedure. Later common sub-tasks can be broken into sub-procedures.

2.1 Database Structure

The database structure is defined, edited and upgraded using XMLDB system. XMLDB is the moodle's database abstraction layer and it is the library code that allows moodle to interact and access database. From moodle 1.7 it worked with some more RDBMS. Moodle uses ADODB internally. It is the database abstraction library for PHP. ADO means ActiveX Data Objects. It used ADODB library as the basis of its database abstraction layer.But the issue here is the extra layer of the library code that had a noticeable impact on performance. So from moodle 2.0 it switched to its own abstraction layer which is a thin wrapper around the various PHP database libraries.

It has well defined group of functions to handle all DB structure (DDL) using one neutral description, being able to execute the correct SQL statements required y each RDBMS. All these functions are used exclusively by installation and upgrade processes. To retrieve or modify database content DML functions are used. These functions provide a high level of abstraction and guarantee that database manipulation will work against different RDBMSes.

2.2 Tables

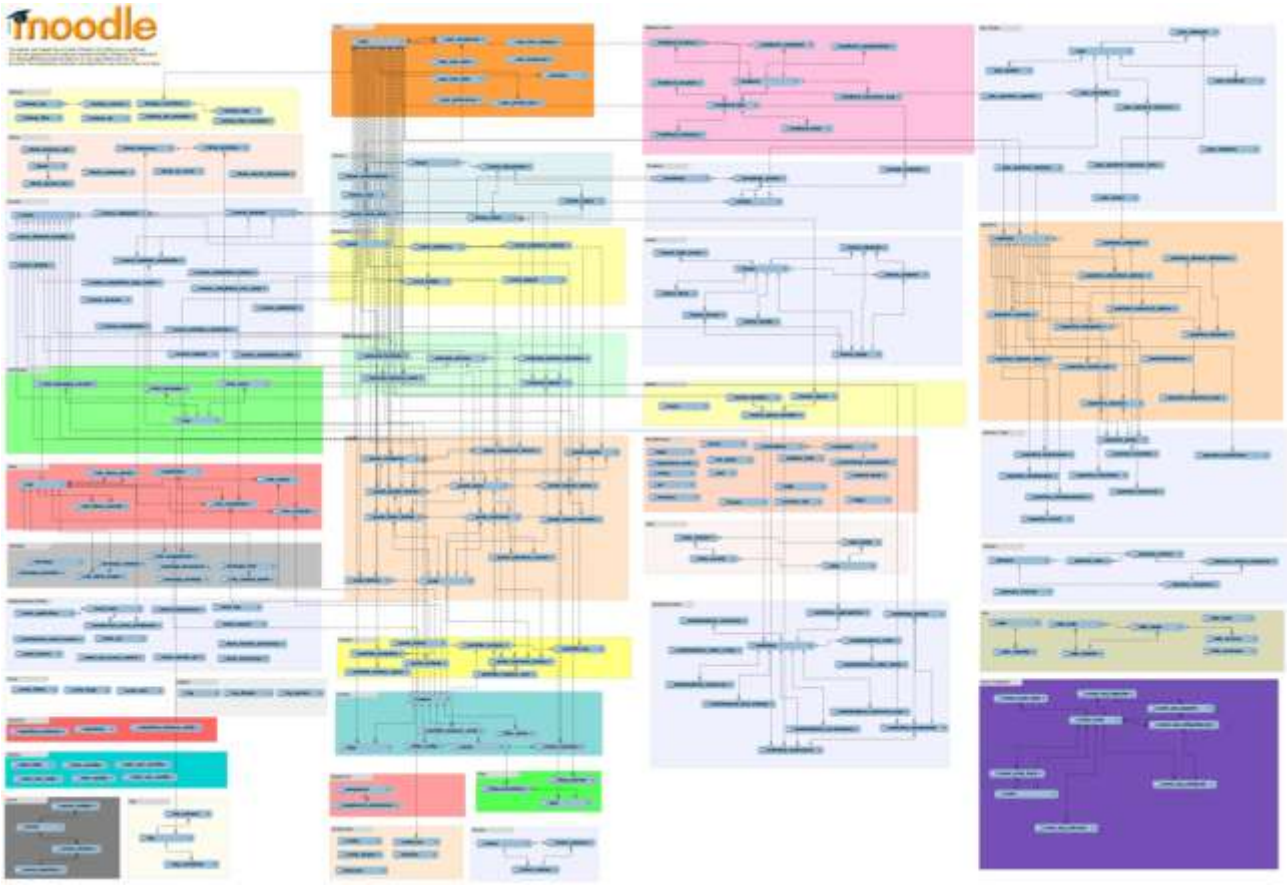
There are 314 tables in Moodle 2.5. They are divided into different categories based on the type of data they store. They are

1. Configuration
2. Users and Profiles
3. Roles and Capabilities System
4. Courses
5. Groups
6. Logging System
7. Blocks System
8. Events
9. Backup and restore
10. Statistics
11. Tags
12. Grade Book
13. Question Bank
14. Messaging System
15. Moodle Network
16. Caching
17. Miscellaneous
18. Activity Modules
19. Blocks
20. Question Types

2.3 Schema

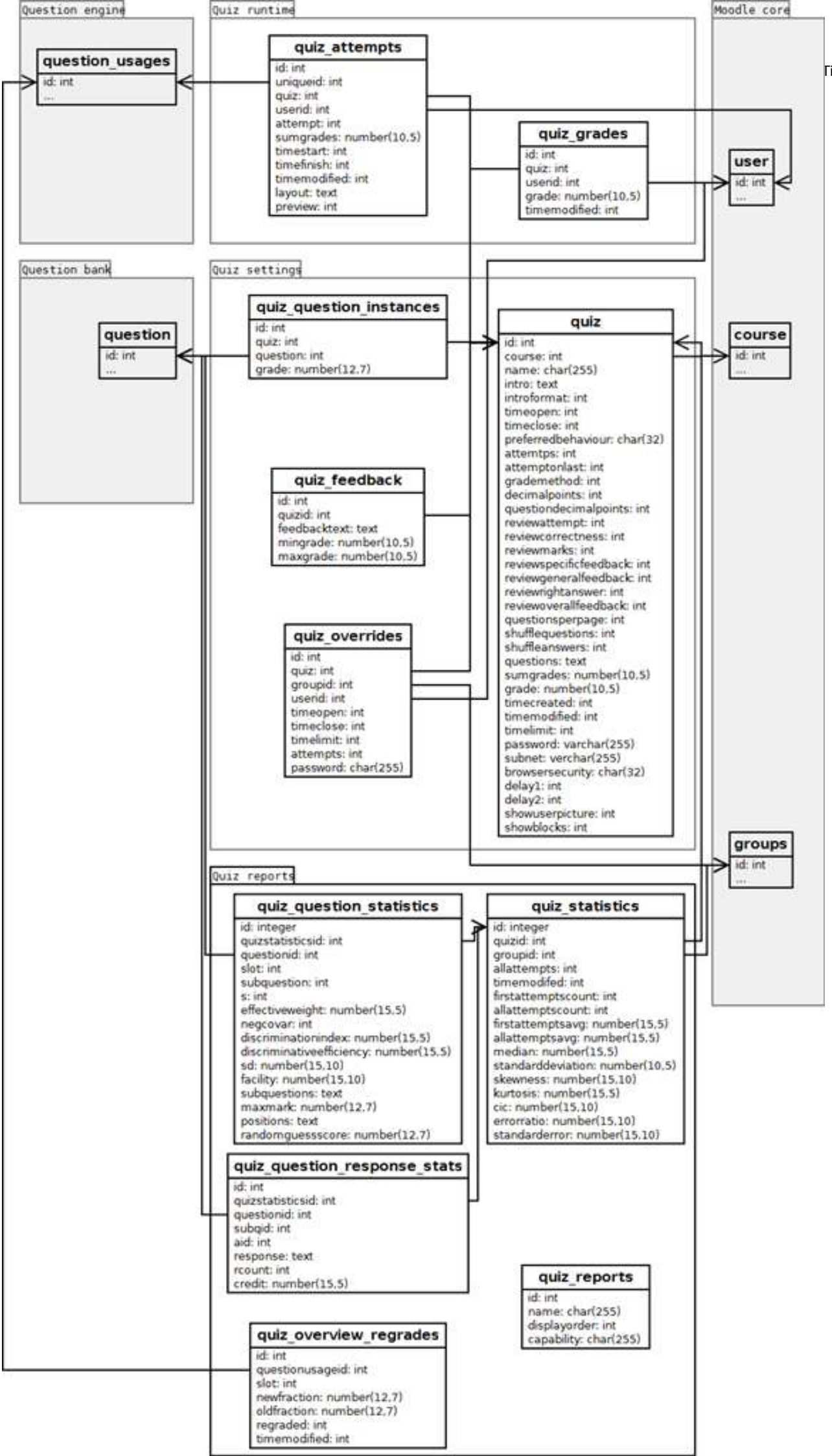
The Moodle 2.2 Database Schema "reverse engineered" from the installed database using MySQL Workbench[4] http://www.mysql.com/products/workbench.
http://docs.moodle.org/dev/Database_schema_introduction

Moodle ER diagram



3. Quiz Module

3.1 Quiz Database


Question engine
Quiz runtime
Moodle core
question_usages
id: int
...
quiz_attempts
id: int
uniqueid: int
quiz: int
userid: int
attempt: int
sumgrades: number(10,5)
timestart: int
timefinish: int
timemodified: int
layout: text
preview: int
quiz_grades
id: int
quiz: int
userid: int
grade: number(10,5)
timemodified: int
user
id: int
...
Question bank
Quiz settings
question
id: int
...
quiz_question_instances
id: int
quiz: int
question: int
grade: number(12,7)
quiz
id: int
course: int
name: char(255)
intro: text
introformat: int
timeopen: int
timeclose: int
preferredbehaviour: char(32)
attempts: int
attemptonlast: int
grademethod: int
decimalpoints: int
questiondecimalpoints: int
reviewattempt: int
reviewcorrectness: int
reviewmarks: int
reviewspecificfeedback: int
reviewgeneralfeedback: int
reviewrightanswer: int
reviewoverallfeedback: int
questionsperpage: int
shufflequestions: int
shuffleanswers: int
questions: text
sumgrades: number(10,5)
grade: number(10,5)
timecreated: int
timemodified: int
timelimit: int
password: varchar(255)
subnet: verchar(255)
browsersecurity: char(32)
delay1: int
delay2: int
showuserpicture: int
showblocks: int
course
id: int
...
quiz_feedback
id: int
quizid: int
feedbacktext: text
mingrade: number(10,5)
maxgrade: number(10,5)
quiz_overrides
id: int
quiz: int
groupid: int
userid: int
timeopen: int
timeclose: int
timelimit: int
attempts: int
password: char(255)
groups
id: int
...
Quiz reports
quiz_question_statistics
id: integer
quizstatisticsid: int
questionid: int
slot: int
subquestion: int
s: int
effectiveweight: number(15,5)
negcovar: int
discriminationindex: number(15,5)
discriminativeefficiency: number(15,5)
sd: number(15,10)
facility: number(15,10)
subquestions: text
maxmark: number(12,7)
positions: text
randomguessscore: number(12,7)
quiz_statistics
id: integer
quizid: int
groupid: int
allattempts: int
timemodfed: int
firstattemptscount: int
allattemptscount: int
firstattemptsavg: number(15,5)
allattemptsavg: number(15,5)
median: number(15,5)
standarddeviation: number(10,5)
skewness: number(15,10)
kurtosis: number(15,5)
cic: number(15,10)
errorratio: number(15,10)
standarderror: number(15,10)
quiz_question_response_stats
id: int
quizstatisticsid: int
questionid: int
subqid: int
aid: int
response: text
rcount: int
credit: number(15,5)
quiz_reports
id: int
name: char(255)
displayorder: int
capability: char(255)
quiz_overview_regrades
id: int
questionusageid: int
slot: int
newfraction: number(12,7)
oldfraction: number(12,7)
regraded: int
timemodified: int

3.2 Question Types

Moodle supports many types of questions.It supports 12 types of questions by default.

1. Calculated

Calculated questions are like numerical questions but with the numbers used selected randomly from a set when the quiz is taken.

2. Calculated Multi choice

Calculated multichoice questions are like multichoice questions which choice elements can include formula results from numeric values that are selected randomly from a set when the quiz is taken.

3. Calculated Simple

A simpler version of calculated questions which are like numerical questions but with the numbers used selected randomly from a set when the quiz is taken.

4. Embedded Answers

Questions of this type are very flexible, but can only be created by entering text containing special codes that create embedded multiple-choice, short answers and numerical questions.

5. Essay

Allows a response of a few sentences or paragraphs. This must then be graded manually.

6. Matching

The answer to each of a number of sub-question must be selected from a list of possibilities.

7. Multiple Choice

Allows the selection of a single or multiple responses from a pre-defined list.

8. Numerical

Allows a numerical response, possibly with units, that is graded by comparing against various model answers, possibly with tolerances.

9. Random Short Answer Matching

Like a Matching question, but created randomly from the short answer questions in a particular category.

10. Short Answer

Allows a response of one or a few words that is graded by comparing against various model answers, which may contain wildcards.

11. True/False

A simple form of multiple choice question with just the two choices 'True' and 'False'.

12. Description

This is not actually a question. Instead it is a way to add some instructions, rubric or other content to the activity. This is similar to the way that labels can be used to add content to the course page.

3.3 SQL Query Log

As moodle uses its own abstraction layer to convert the PHP queries to SQL quries it is quite interesting to know if there is a performance issue in the SQL queries that are generated. So to know what are the queries that get executed on performing a particular action on moodle site first we have to set up a SQL log file . So our first consideration is to get the queries and later check the performance of those queries. We have considered the QUIZ activity here.

Log Set-up

1. To turn MySQL general log we must edit MySQL configuration file located in /etc/mysql/my.cnf.
2. sudo nano /etc/mysql/my.cnf
3. Uncomment the following lines by removing "#".,

```
general_log_file = /var/log/mysql/mysql.log
general_log =1
```

4. Save the changes.
5. Restart mysql server sudo service mysql restart.

Log file will be created in /var/log/mysql .The default name for this file will be mysql.log. Mysql general log is a performance killer so should be enabled only when necessary.You can flush the sql log > /var/log/mysql/mysql.log.

Case Study on LMAX Architecture for Improving the Performance of Moodle

LMAX is a new retail financial trading platform. As a result it has to process many trades with low latency. The system is built on the JVM platform and centers on a Business Logic Processor that can handle 6 million orders per second on a single thread. The Business Logic Processor runs entirely in-memory using event sourcing. The Business Logic Processor is surrounded by Disruptors - a concurrency component that implements a network of queues that operate without needing locks. During the design process the team concluded that recent directions in high-performance concurrency models using queues are fundamentally at odds with modern CPU design.

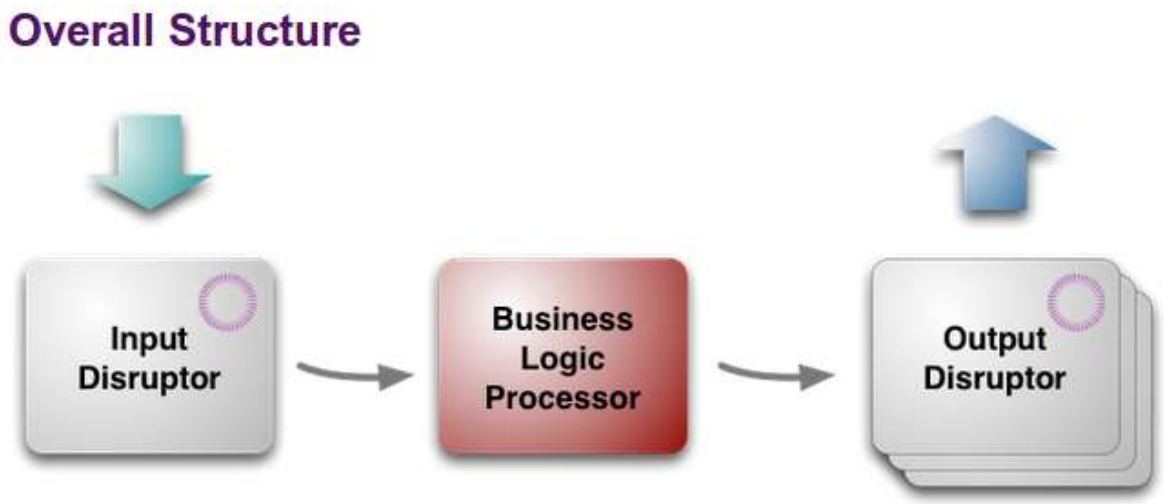


Figure 1: LMAX's architecture in three blobs

LMAX architecture was developed by the team behind the BETFAIR website. BETFAIR is an online betting site, where a few million users login to the site and give another millions of business requests during the time of a sporting event. Considering the current web architecture of BETFFAIR there are a lot of exceptions and drawbacks due to which it may crash during overload. The BETFAIR team has dealt with concurrency in such a way that scalabilty and modifications are difficult and the site lacks a lucid documentation on the architecture and database connectivity. Currently the site uses multiple threads to deal with concurrent users which always poses a problem, escpecially in business processing web architectures.

The LMAX architecture deals with concurrency by using a single thread instead multiple threads. The first and foremost advantage LMAX is that it can handle six million requests on a single thread. This means that there are no issues of multiple-threading with this architecture. Another advantage of using this architecture is that there is no need of changing an already written efficient-business-processing algorithm.

The prime difference which we find in this model is the way in which INPUT & OUTPUT is processed. Here an advanced technology named DISRUPTOR technology is used handle the input and output requests. The input DISRUPTOR sends requests to the 3 core parts of the system:

1. **JOURNALER**
2. **UN-MARSHALLLER**

3. REPLICATOR

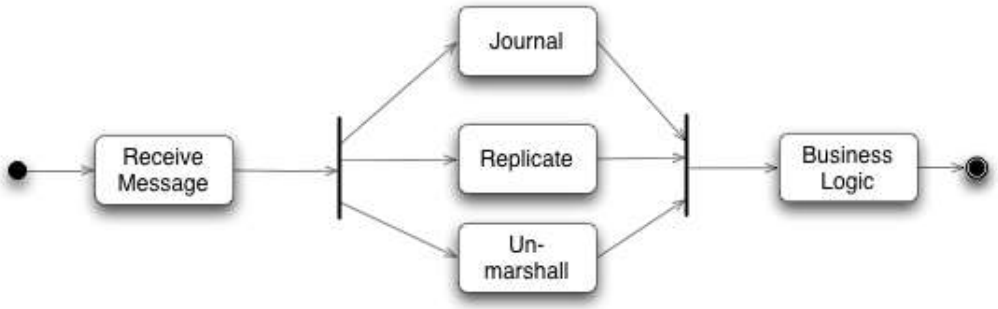


Figure 2: The activities done by the input disruptor (using UML activity diagram notation)

Although the business logic occurs in a single thread, there are number tasks to be done before we can invoke a business object method? The original input for processing comes off the wire in the form of a message, this message needs to be unmarshalled into a form convenient for Business Logic Processor to use. Event Sourcing relies on keeping a durable journal of all the input events, so each input message needs to be journal onto a durable store. Finally the architecture relies on a cluster of Business Logic Processors, so we have to replicate the input messages across this cluster. Similarly on the output side, the output events need to be marshalled for transmission over the network.

Journaler
The journaler records all the activities what a user does, including the changes made to the account and the requests raised, this is done by the method of EVENT-SOURCING. This is done to ensure that recovery of data in the case of any system/session failure is easier. With the help of the journaler the user can continue from where they had encountered an error during the last login.

Replicator
The replicator replicates the input request and sends it to the other servers which are connected to the active-server, which is the only server from which the output is fed to the output disruptor. The system, in which the BLP (Business Logic Processor) is loaded, is actually a combination of 3/more web servers. This is to ensure that in case the active-server fails one of the servers from the rest becomes active and starts generating the output. This solves the issue of a server failure.

UN-Marshaller
The user request details are sent to the BLP by converting it into an object (JAVA Object), and the BLP works on the this encapsulated version of the user request. The BLP produces an O/P which is sent to OUTPUT DISRUPTOR. This part of the disruptor makes the required object.

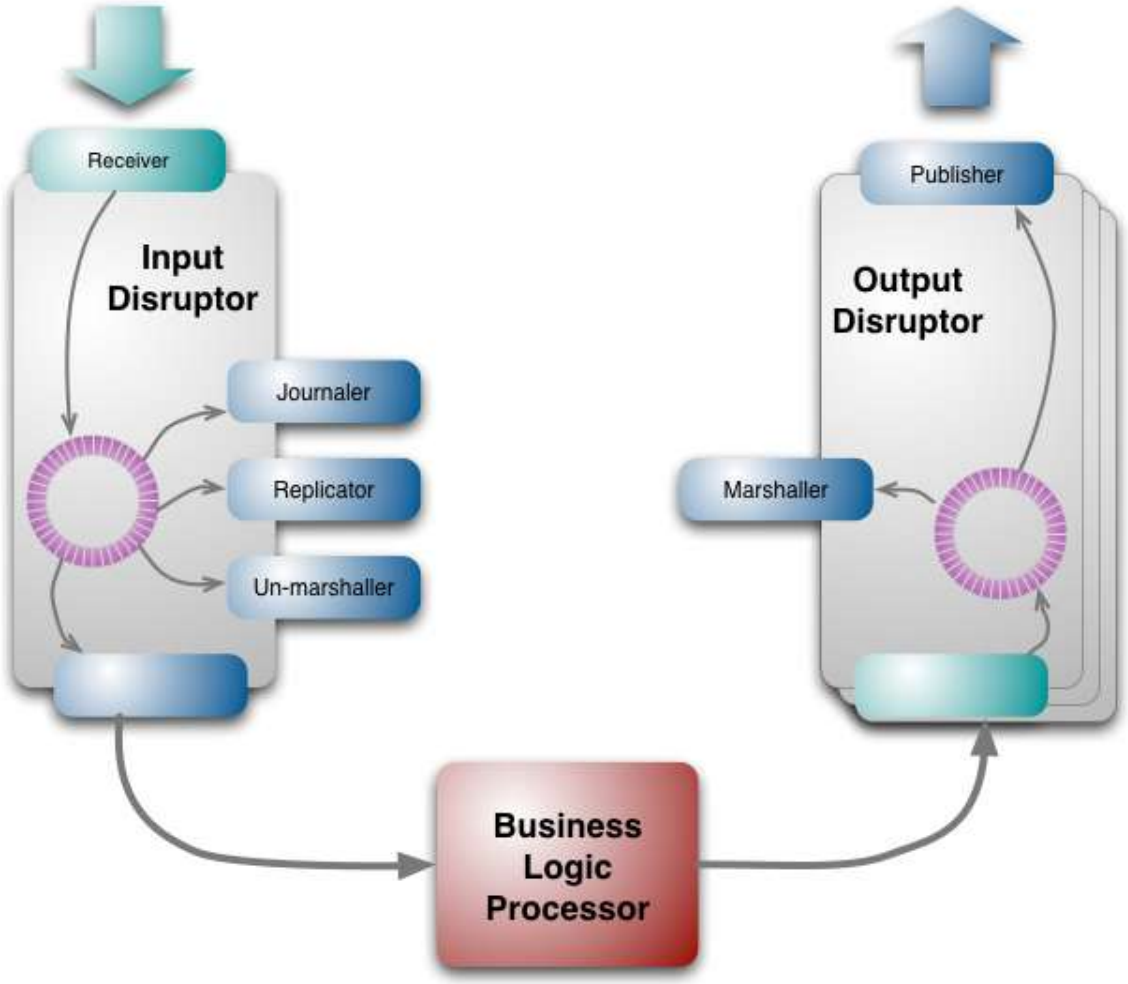


Figure 4: The LMAX architecture with the disruptors expanded

Conclusion

LMAX architecture was designed and tested for a server to process more than 6million requests, this is actually required for a website with a high amount of traffic. This necessity is found in business sites and is absolutely useless in the case of moodle where a maximum of 500 students may give a test. So trying to optimise moodle by changing its architecture to LMAX is not feasible.

SSD v/s HDD for the 'moodle' DATABASE

The moodle 2.5 uses a database named 'moodle'.
All the alterations done in the front end of the moodle get Reflected/stored in the different tables of the database, the data gets Populated under different courses and users in the front end (GUI)
Of the moodle by carrying out a READ operation in the corresponding tables of the above database.

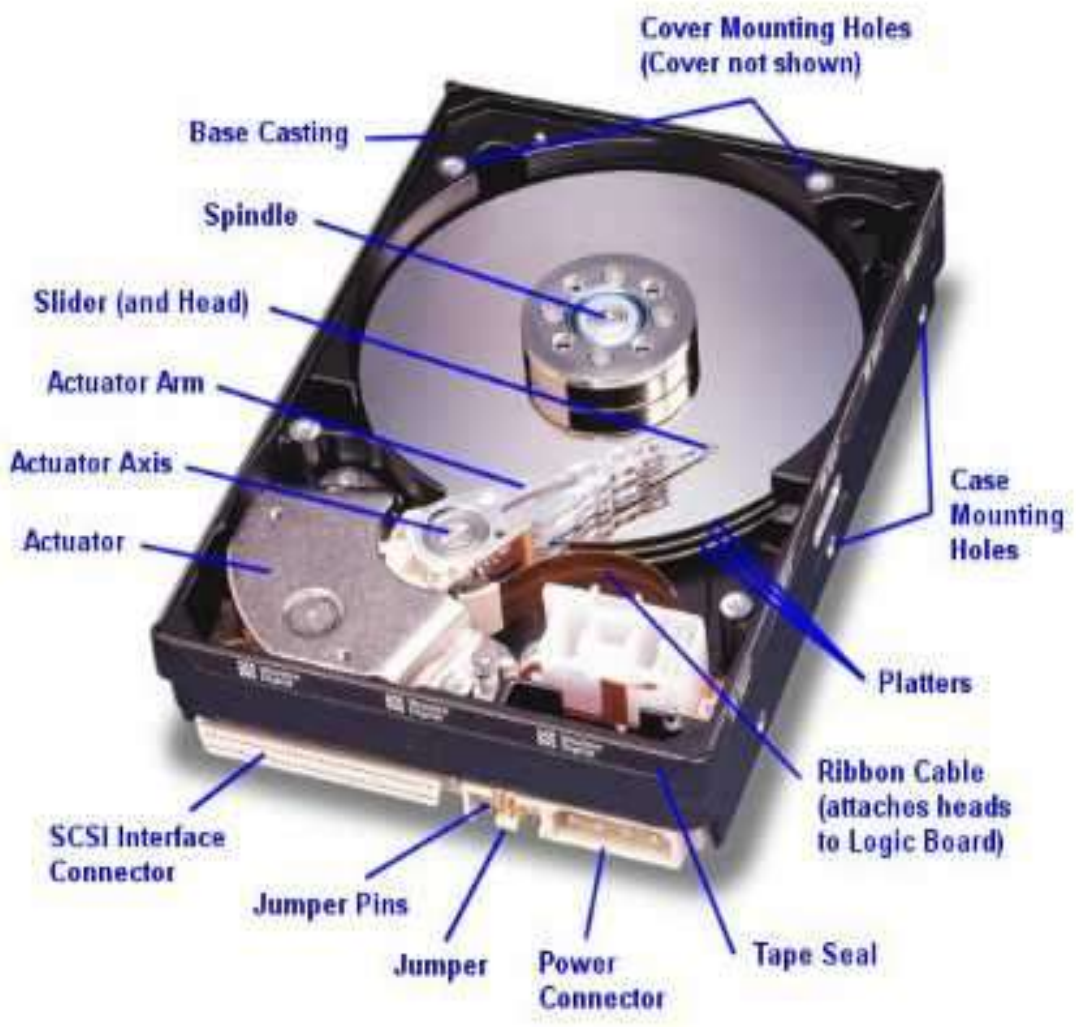
Similarly a WRITE operation (considerately expensive) is encountered in a moodle when bulk users are added to a database.

Fetching (reading) the data from a database is more easy (cost) as compared to updating/inserting (writing) a value. When the size of a database (or any file) is large (say more the 10GB) then Fragmentation occurs, ie the file gets split-up into smaller parts and it gets stored in different areas of the storage disk. This results in random access of data, this is more expensive as compared to Accessing data from contiguous memory locations.

Because of their spiral-like recording surfaces, HDD surfaces work best with larger files that are laid down in contiguous blocks [5]. That way, the drive head can start and end its read in one continuous motion. When hard drivesstart to fill up, large files can become scattered around the disk platter, which is otherwise known as fragmentation.

While read/write algorithms have improved where the effect in minimized, the fact of the matter is that HDDs can become fragmented, while SSDs don't care where the data is stored on its chips, since there's no physical read head.SSDs are inherently faster.

PCIe SSDs



They speed up performance by eliminating the moving parts in traditional hard drives, and improve performance by residing on the PCIe bus closer to the CPU and memory.

The latency of a hard drive (how long it takes to find something) is around 150 times longer than that of an SSD. The transfer rate of a hard drive (how fast it reads once it's found the file) is around 2 to 3 times slower than an SSD for reading, and (very roughly) similar to an SSD for writing.

Once found the data gets stored in the cache, so for further read and write operations it will be more faster as compared to the initial read and write operations.

COMMANDS Executed In The MySQL SERVER

1. SELECT username FROM mdl_user;
HDD : 0.27s , 0.04s , 0.05s
SSD : 0.12s , 0.03s , 0.02s
2. SELECT email FROM mdl_user;
HDD : 0.03s , 0.04s
SSD : 0.01s , 0.01s
3. SELECT email , username FROM mdl_user;
HDD : 0.05s , 0.06s
SSD : 0.04s , 0.04s
4. UPDATE mdl_user SET email='AshwaJith@gmail.com';
HDD : 5.41s , 6.75s , 7.62s
SSD : 4.28s , 4.35s , 3.47s
5. UPDATE mdl_user SET password='helloworld';
HDD : 4.12s , 3.47s
SSD : 1.87s , 1.71s

Conclusion

From the above results it can be confirmed that the executiontimes of moodle-operations especially those which updates/stores
values in the database will get decreased drastically if we load
both the moodle and its assosciated database in an SSD in the
SERVER.

References

[1] LMAX : http://martinfowler.com/articles/lmax.html : figure1
[2] LMAX : http://martinfowler.com/articles/lmax.html : figure2
[3] LMAX : http://martinfowler.com/articles/lmax.html : figure3
[4] LMAX : http://martinfowler.com/articles/lmax.html : figure4
[5] Wikipedia : http://en.wikipedia.org/wiki/Hard_disk_drive
[6] Wikipedia : http://en.wikipedia.org/wiki/Hard_disk_drive
[7] Wikipedia : http://en.wikipedia.org/wiki/Ssd

Optimizing Moodle Lms To Improve User Response Time

Objective

Web Applications are now required in each and every industry, including business, education, tourism, entertainment and many more. The Objective of our project is to optimize Moodle LMS to reduce average user response time by employing numerous front end and back end optimization techniques.

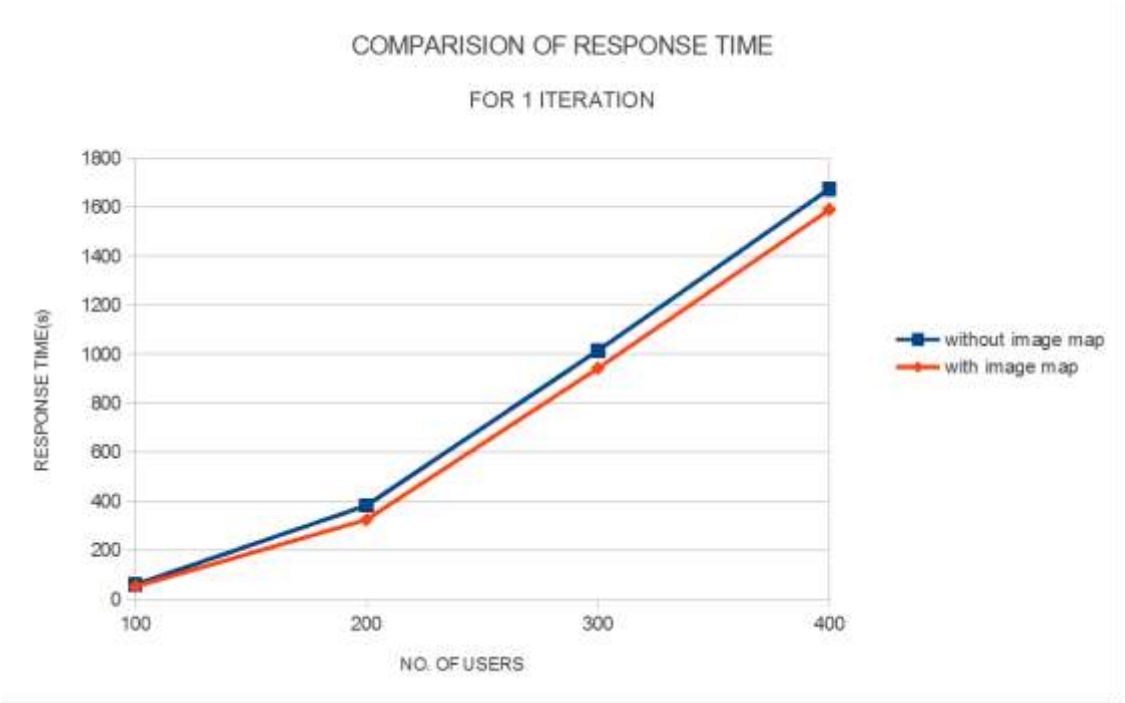
Experiments Conducted

1. Front-End Optimization Using Image Maps

In its simplest form, a hyperlink associates the destination URL with some text. A better alternative is to associate the hyperlink with an image, for example in navigation bars and buttons. If you use multiple hyperlinked images in this way, image maps may be a way to reduce the number of HTTP requests without changing the page's look and feel. An image map allows you to associate multiple URLs with a single image. The destination URL is chosen based on where the user clicks on the image. I have created a set of 10 pages with image map and another set of 10 pages without image map. The results obtained are as follows:

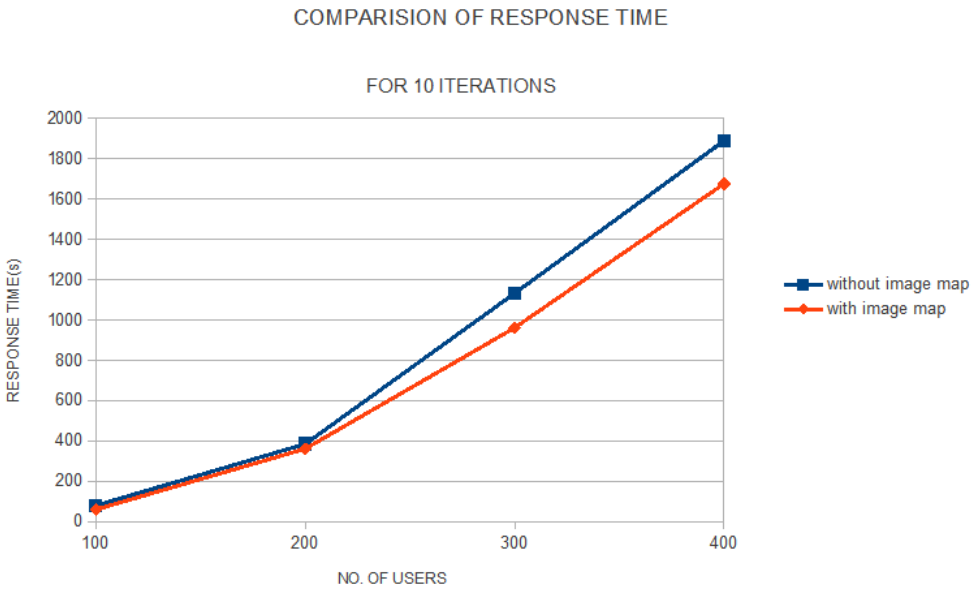
Results for 1 iteration

users	without image map				with image map				
	EXP 1	EXP 2	EXP 3	AVERAGE	EXP 1	EXP 2	EXP 3	AVERAGE	Factor
100	68.5	51.2	61.6	60.43	45.9	53	57.9	52.26	1.15
200	390.3	370.9	388.5	383.2	341.9	329	306.1	325.6	1.17
300	1094.8	991.3	958.9	1015	1041.3	984.3	802.5	942.7	1.07
400	1614.2	1526.6	1883.1	1674.6	1548.1	1367.9	1852.6	1589.5	1.05



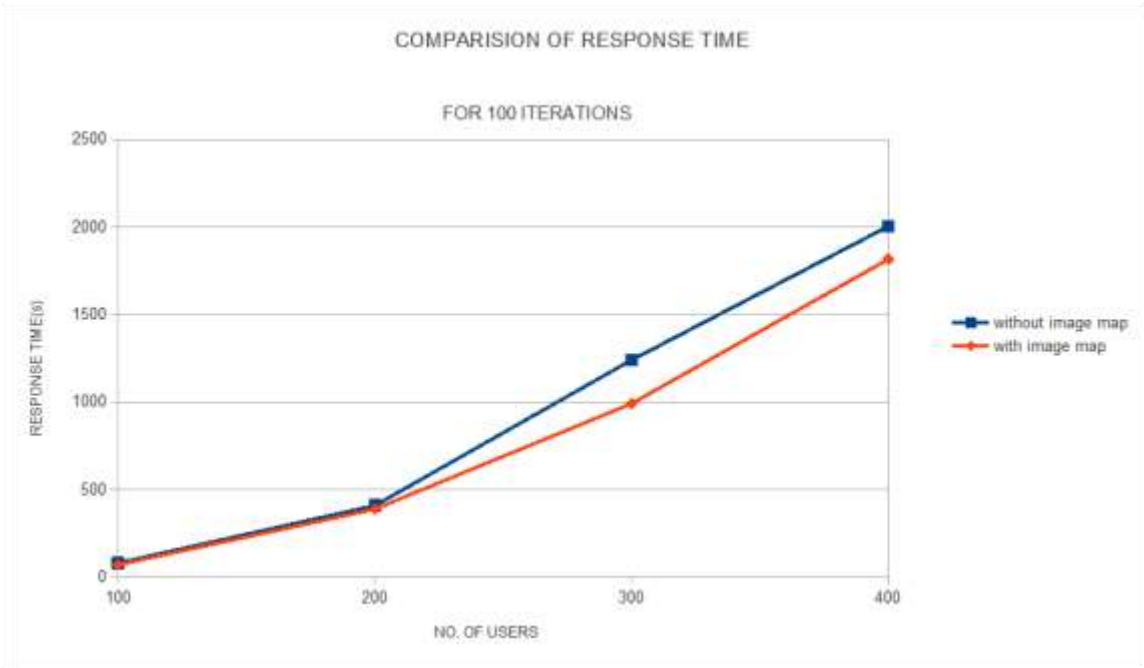
Results for 10 iterations

Users	without image map				with image map				
	EXP 1	EXP 2	EXP 3	AVERAGE	EXP 1	EXP 2	EXP 3	AVERAGE	Factor
100	73.4	82.7	81.6	79.23	67.1	55.2	56.5	59.6	1.3
200	406.7	407	340.4	384.7	379.3	376	329.6	361.33	1.06
300	1061.1	1146.1	1190.4	1132.5	916.9	1036.9	930.5	961.43	1.17
400	1801.9	1785	2080.4	1889.1	1594.7	1482.2	1951.6	1676.1	1.13



Results for 100 iterations

users	without image map				with image map				
	EXP 1	EXP 2	EXP 3	AVERAGE	EXP 1	EXP 2	EXP 3	AVERAGE	Factor
100	74.8	81.2	80	78.67	68.4	65.9	73.9	69.4	1.13
200	425.3	382.9	414.26	407.48	420.9	376.7	365.2	387.6	1.05
300	1148.6	1335.1	1236.4	1240.3	1095.2	918.4	958.6	990.73	1.3
400	1918.6	2016.9	2077.7	2004.4	1647.8	1769.9	2030.9	1816.2	1.1



2. Front End Optimization by Using Far Future Expires Header

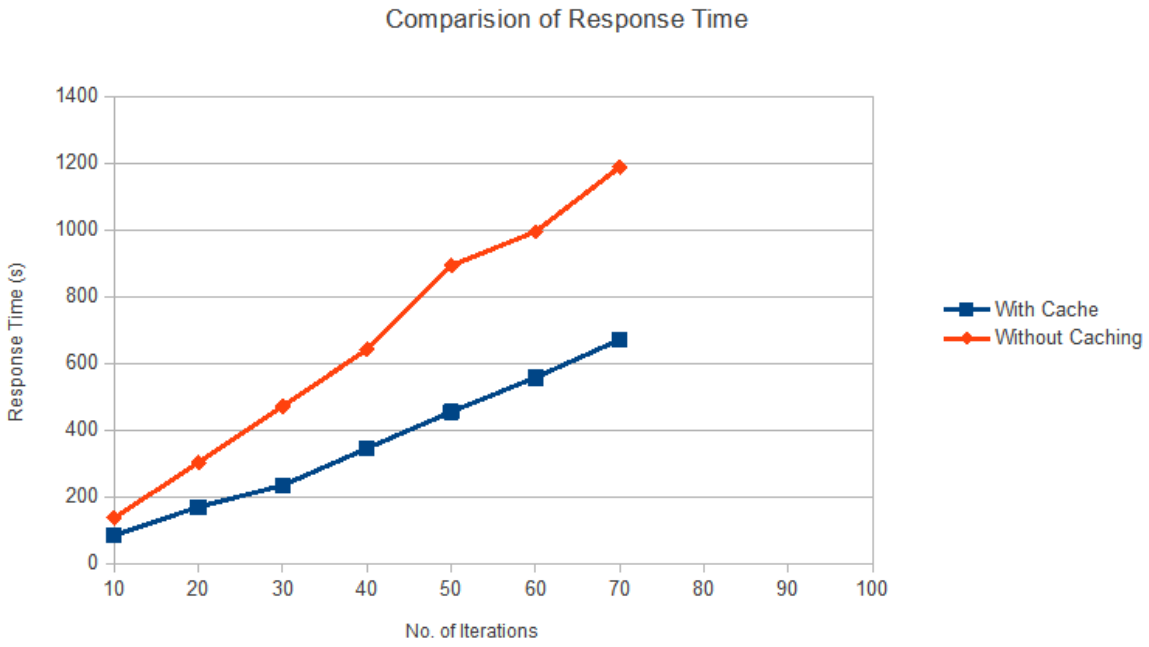
Browsers (and proxies) use a cache to reduce the number of HTTP requests and decrease the size of HTTP responses, thus making web pages load faster. A web server uses the Expires header to tell the web client that it can use the current copy of a component until the specified time.

Moodle sends requests with an Expires Header which is set in past (20th Aug 1969 09:23 GMT). After changing it to 20th Aug 2015 09:23 GMT , the results obtained are:

Results

To View 3 Pages starting from Login Screen:
Login -> Home -> View Course -> Logout

No. of Iterations	With Cache	Without Caching
10	84.075	136.372
20	168.97	302.044
30	233.708	470.641
40	344.575	641.054
50	454.205	892.575
60	557.33	994.313
70	670.668	1188.28



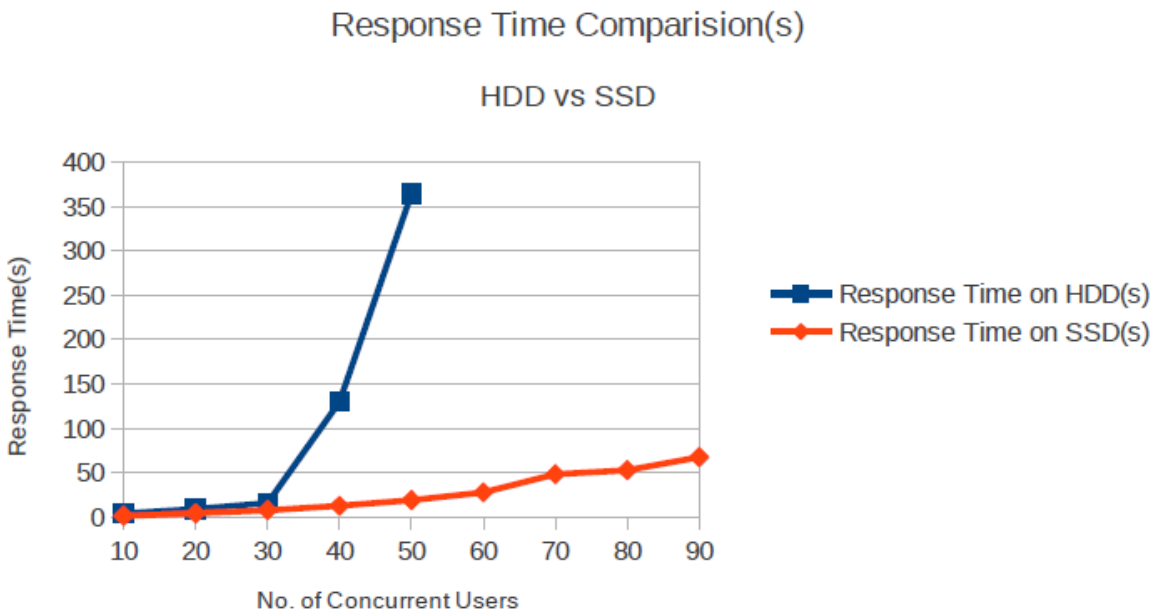
3. Load Testing on SSD vs HDD

Testing was done on two servers with 1GB RAM on Intel® Core™ i5-2310 CPU @2.90GHz × 4 processor , one with HDD and one with SSD. Testing was done for Chat Activity.

Results

No. of Concurrent Users	Response Time on HDD(s)	Response Time on SSD(s)
10	3.671	0.958
20	8.874	3.78
30	15.303	7.373
40	129.786	12.261
50	364.48	18.81
60	System Crashed	27.475
70	System Crashed	48.127
80	System Crashed	52.651
90	System Crashed	67.344

Graph


Response Time Comparision(s)
HDD vs SSD
Response Time(s)
400
350
300
250
200
150
100
50
0
10
20
30
40
50
60
70
80
90
No. of Concurrent Users
Response Time on HDD(s)
Response Time on SSD(s)

Jmeter Testing On Moodle 2.5

Having the Jmeter testplan generation plugin for moodle,various tests were conducted for analysing the performance of different activity modules,mainly quiz.Also analysed how login and logout time varies by increasing the number of concurrent users.

Experimental Setup

All the tests are done in Intel® Core™ i3-3110M CPU,4 GB RAM.

The various steps in doing the experiment are:

1.Generate Jmeter Script

Extract the jmeter script files into [moodle_dir]/admin/report/loadtesting

2.Login to moodle.

Select Settings > Site Administration > Reports > JMeter loadtesting

3. Launch jmeter.

4.Run the script and view the results

The following tests were carried out:

1.Compare different activities

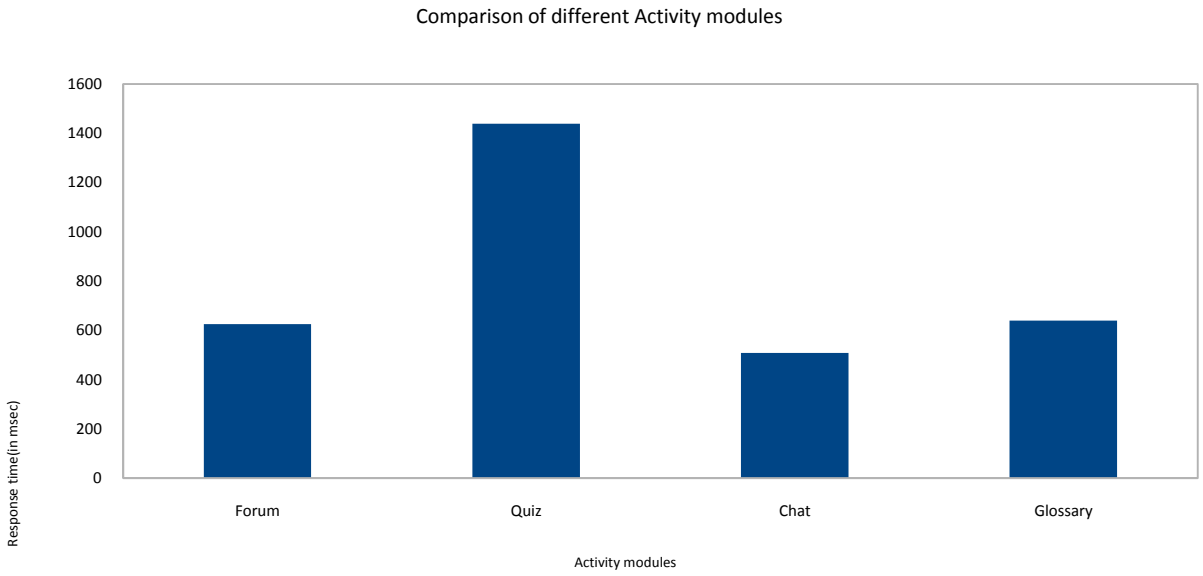
2.Quiz Activity

3.Login time

1.Comparing Different Activities

Each testplan included all steps required to perform the task (logging in, navigating to required page, posting in chat/forum or submitting quiz).The ramp period is kept 1 sec in all test cases.The no of user is also 1.The average page performance results (mean time in milliseconds spent on each page retrieved throughout the testplan) are shown in the table below:

Activity	Response time
Forum	626
Quiz	1439
Chat	509
Glossary	640

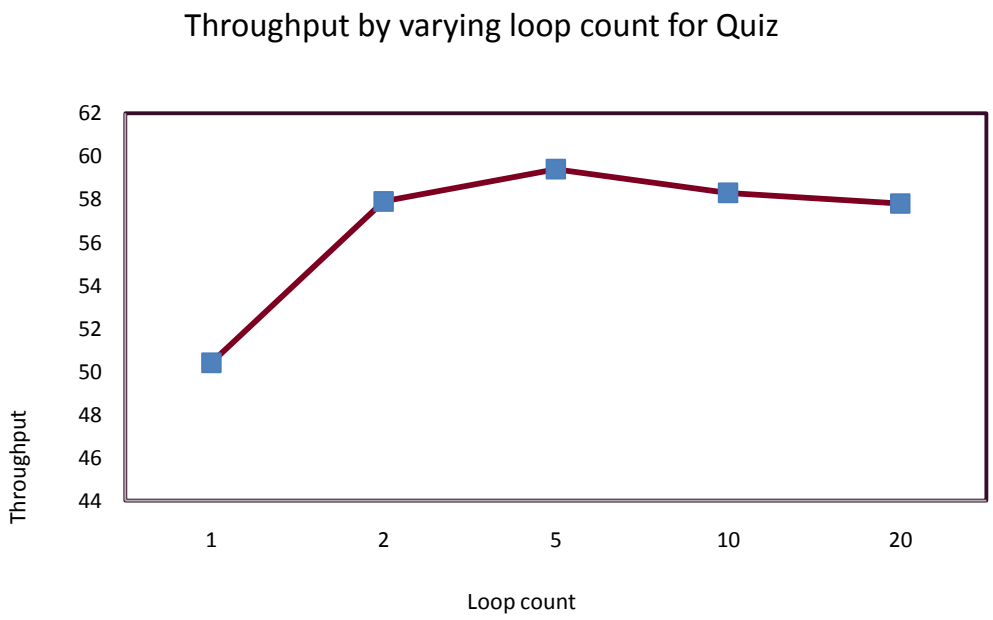


From the graph it is clear that quiz activity take the highest response time.

2.Quiz Activity

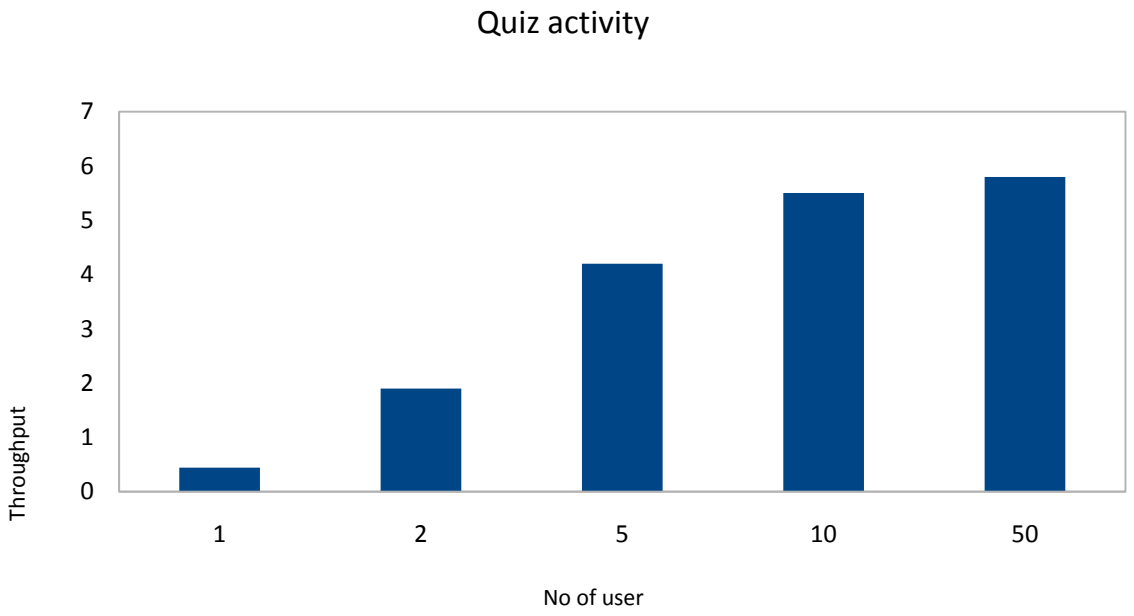
Different tests were carried out in quiz activity to measure the throughput by increasing the loop cont and varying the no. of users.

2.1 Increasing loop count



When loop count is increased,the throughput remains fairly constant.

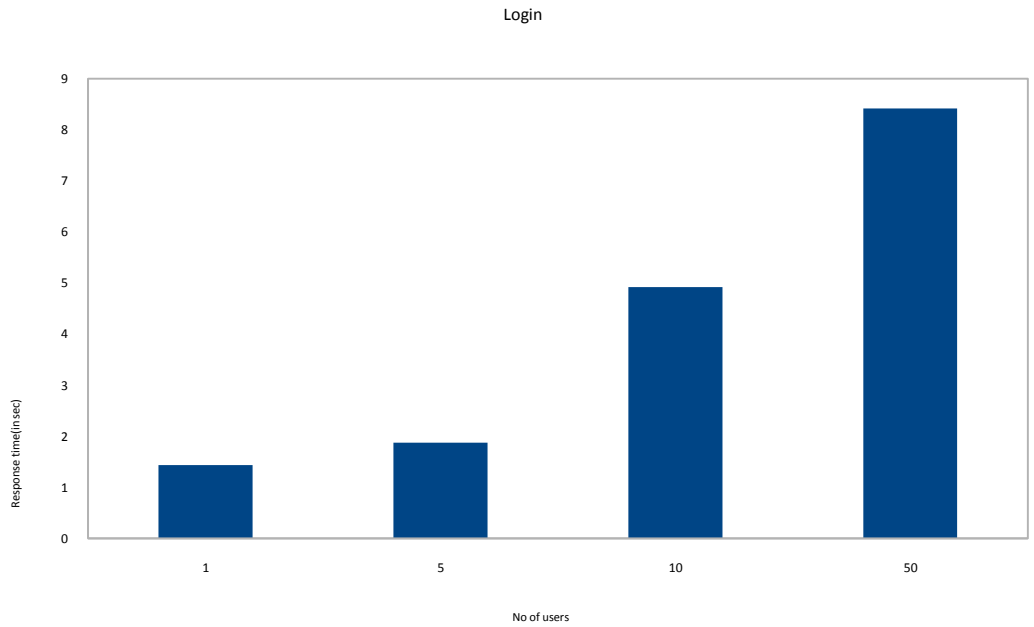
2.2 Increasing the number of users



The test was done keeping loop count=25,ramp up period=1 sec.

After 10 number of threads,throughput increases only slightly.

3.Login



The test was done with ramp up time=1 sec and loop count=2

The graph shows average time required for page retrieving for different number of users.As the number of users is increased,response time also increases as expected.

From the graph,the following conclusion can be drawn

1.Quiz activity takes the maximum response time.

2.Throughput remains fairly constant after increasing the no of users to a certain limit.

3.Response time for login increases with the number of users.

SQL QUERIES GENERATED FOR JMETER

Experimental Setup

1.Setup the SQL log file by editing the configuration file /etc/mysql/my.cnf
Uncomment the lines
general_log_file = /var/log/mysql/mysql.log
general_log =1
2.Restart MySQL service
3.Login to moodle and generate Jmeter Script for Quiz activity
4.Run Jmeter script
5.The queries can viewed in logfile located at /var/log/mysql/mysql.log

The Jmeter script is generated for quiz activity which performs
1.Login to site
2.View Course
3.View Quiz
4Start Attempt
5.Submit Quiz data
6.View Quiz
7.Finish Attempt
8.Logout from site

The script was generated for a single user for attempting a quiz consisting of 2 multiple choice questions.For manually attempting the quiz,**554** queries are generated for just answering the quiz(excluding login,logout,course selection etc).On the other hand,by using Jmeter script for moodle,the number of queries are only **685.**

Introduction

Moodle, by default works on Apache web server. However there are much more powerful & lightweight servers available which may lead to better user response times. Few of these are Lighttpd, Nginx, Cherokee & LiteSpeed.

JMeter was the tool extensively used for testing the performance of Moodle under various test conditions. JMeter is a loadtesting tool which is used to analyse the performance of any web application.

To compare with other servers, extensive tests were done with the default Moodle configuration.

Experimental Setup

- Installation
  - Install LAMP Server
  - Install latest version of moodle.
  - Extract into /var/www
  - Create /var/moodledata and assign appropriate permissions.
  - Disable other applications that make use of ports 80 & 443.
  - Create required databases.
  - Begin Moodle Ins stallation from localhost/moodle
  - Provide details to finish installation.

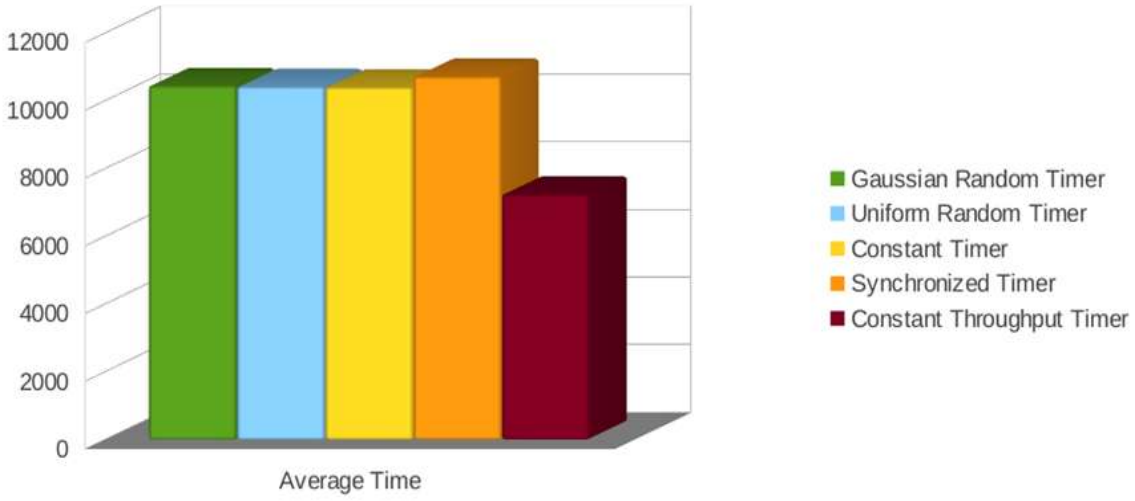
- Testing
  - Add loadtesting plugin to moodle
    - https://github.com/kabalin/moodle-jmeter-script-generator
  - Select testing criterion.
  - Generate Jmeter Script.
  - Load Script onto Jmeter
  - To test different timers, add required timer from menu and provide necessary parameters.
  - To test for different number of users, change number of threads accordingly.
  - To alter the time period within which 'n' concurrent users go live, alter the Ramp-Up Time.
  - Run test.

To test on another server

- Setting up Moodle on Lighttpd
  - Set up LLMP stack.
  - Extract Moodle packages into /var/www.
  - Create database directory moodledata.
  - Change respective directory permissions.
  - Set up database.
  - Install Moodle.

- Testing
  - Same as above

Experimental Results



Comparison Of Various Timers

5. 412 Users

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse... **Log/Display Only:** Errors Successes **Configure**

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
[QUIZ 2] Login to site	1560	15443	776	61340	6827.94	0.00%	5.1/sec	118.11	23635.5
[QUIZ 2] View Course	1533	8411	2164	38069	2030.45	0.00%	5.0/sec	173.58	35251.9
[QUIZ 2] View Quiz	1408	9292	4080	40345	2355.91	0.00%	4.7/sec	158.98	34647.3
[QUIZ 2] Start Attemp...	1260	19100	10386	48040	3240.70	0.00%	4.3/sec	66.89	16019.4
[QUIZ 2] Submit quiz d...	1245	12732	6593	41942	3024.65	0.00%	4.5/sec	65.10	14958.3
[QUIZ 2] Finsih Attempt	1188	18941	12221	46925	4370.86	0.00%	4.5/sec	66.83	15285.8
[QUIZ 2] Logout from ...	1178	5298	3084	14456	1134.48	0.00%	4.8/sec	43.50	9352.3
TOTAL	9372	12668	776	61340	6149.45	0.00%	30.7/sec	664.97	22159.6

6. 413 Users

Summary Report

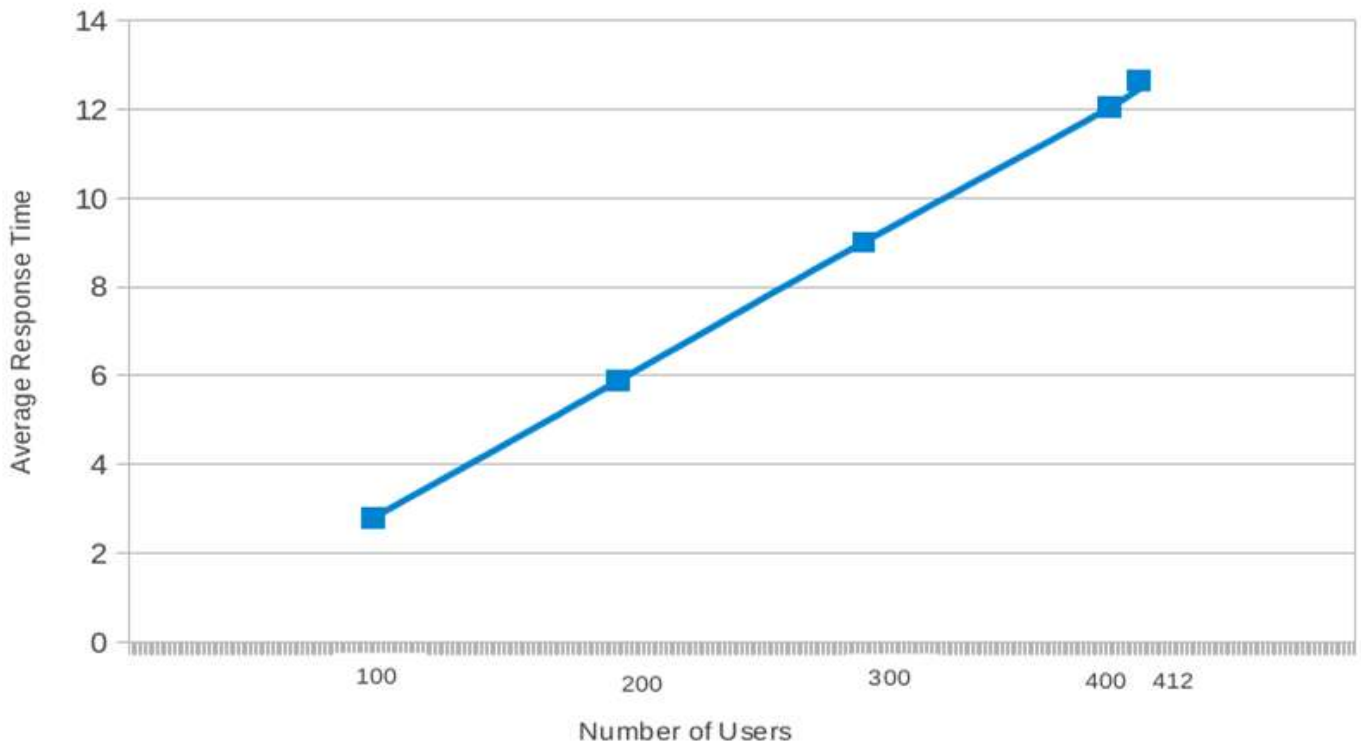
Name: Summary Report

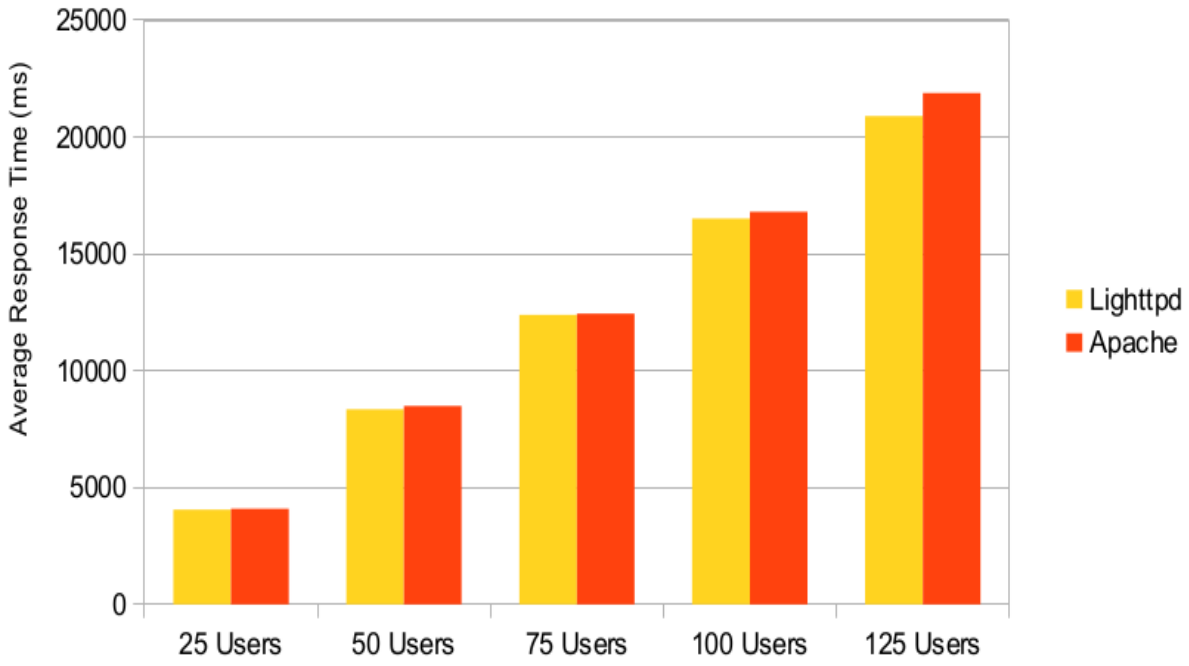
Comments:

Write results to file / Read from file

Filename Browse... **Log/Display Only:** Errors Successes **Configure**

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
[QUIZ 2] Login to site	992	17497	8260	52868	8474.37	0.00%	4.4/sec	100.77	23631.2
[QUIZ 2] View Course	838	9042	6272	34339	2321.15	0.00%	3.8/sec	132.21	35244.0
[QUIZ 2] View Quiz	826	10687	6512	25062	1950.60	0.00%	5.1/sec	196.18	39231.3
[QUIZ 2] Start Attempt - ...	825	21381	14341	44503	2661.83	0.00%	4.2/sec	66.27	16018.5
[QUIZ 2] Submit quiz dat...	824	13669	8054	34625	2361.01	0.00%	5.6/sec	81.36	14957.5
[QUIZ 2] Finsih Attempt	785	19427	12573	63104	8825.63	0.13%	4.8/sec	71.07	15267.2
[QUIZ 2] Logout from site	730	5641	2589	25192	2393.09	0.00%	4.9/sec	44.51	9351.3
TOTAL	5820	14095	2589	63104	7363.51	0.02%	25.6/sec	557.68	22290.9





Conclusions

- Constant Throughput Timer seems to yield lower response times compared to other timers.
- Synchronized Timer seems to give the highest response time and hence must be used for worst case analysis.
- There seems to be a linear relationship between number of users & average response time.
- As per server configurations, the server is capable of handling up to 412 threads for quiz functionality of Moodle without any errors.
- Average Response Time seems to be slightly smaller for Lighttpd than Apache.

Introduction:

Enabling cache for XMLHTTPRequests (AJAX Components) ensures the availability of static structures such as themes and other appearance related aspects at the local client side itself thereby greatly enhancing the response time. This is done by adding a far future expires header for these elements.

Procedure:

- Open /var/www/moodle/lib/outputrendering.php
- Update the Expires header from past to supported far future.
- Record AJAX request response time before and after compiling the changes. (Using iMacros & Firebug)

Result:

For Drag & Drop component:

Without Caching: 309 ms
With Caching: 227 ms

Performance Improvement: 26.5%

Study of various software patterns

A common definition of a pattern is that it is a solution to a problem in a context.For me a pattern is primarily a way to chunk up advice about a topic. Chunking is important because there's such a huge amount of knowledge we need to write software. As a result there needs be ways to divide knowledge up so we don't need to remember it all - what we need is to be able to get at a particular chunk of knowledge when you need it. Only then do we need details.

Moodle Database Configuration/Table indexes and sizes

The Moodle database has around 200 tables, and can be quite daunting at first sight. The good news is that we don't have to understand it all at once. For example, there are eight tables called forum_something. If we are interested in the forum module, then obviously we need to understand these tables, and the places they link into core tables. But if we are not interested in the forum module, we can forget about them. The same is true of each activity module. Once we take out the tables for each activity module in this way, and similarly take out the tables belonging to the enrolment plugins, question types, etc. We are left with about 50 core tables. But the good news is that even here they break down into groups that mostly we can understand together, or ignore. This page lists the core database tables in these groups. Later, it would be good to add more detailed documentation explaining some of these groups.
Table size problem-
mdl_backup_controllers table- It has 22,500 entries and takes up 1.6 Gb of space making it easily the biggest table in the database.
We need a script to automatically delete old records in this table to retrieve space.
Table indexes are needed to speed up searching and processing.

Useful SQL Queries

Queries:-
1.Select users who have not logged in for over 180 days (but not those who have never logged in)
2.Delete users who have not logged in for over 180 days (but not those who have never logged in)
3.Select users Who have NEVER logged in
4.Delete users Who have NEVER logged in
5.Number of views(hits) per student in a course from 2010.
6.Gives a count for resources and activities in a given course
7.Lists all the resources and modules available and makes a count for a given course for those resources or modules that course contains.
8.Number of students who have completed and passed each module at a particular time
9.Show the administrator(s) the ratio between student forum postings and instructor forum postings in each course
10.Show totally opened courses (visible, opened to guets, with no password)
11.Find all information about instructors and the courses they are enrolled in:
12.List all students not enrolled in a course
13.Produce a listing of all active students that do not currently have any enrollments appearing:
14.Return details of your parent and child courses
15.Retrieve all COURSES in ONE CATEGORY
16.Show all students and his courses
17.Find all Teachers in all courses
18.People enrolled in courses and filter by month
19.This query shows that latest last registered user info

20.This query shows that latest last Discussion info
21.Count of all enrolled courses of each users in the db
22.Last access to a course
23.List of all your courses together with how many students are enrolled in each
24.List all the rows in your role assignments table that no longer match to a user

Removal of unused php functions

Possible removal of unused functions in shortanswer/questiontype.php
Unused functions from questionnaire
Remove the grade item for an activity if the grade for that activity is always zero.
Deprecated functions still allowed

Current Work

Implementation of long polling to realtime quiz plugin
Made changes in code
Removal of unused php links during quiz

s