Project Report

Optimizing Moodle LMS to Improve User Response Time

Principal Investigator
Prof. D.B. Phatak

Project In-Charge
Mr. Nagesh Karmali

Project Mentors
Mr. Anoop Dobhal
Mr. Dhanajay Ambekar
Miss. Firuza Aibara (PMO)
Mr. Nilesh Birari
Mr. Raju.B
Miss. Silpa T.

Project Team Members
Miss. Akshaya Muralidharan
Mr. Mevin Dominic
Mr. Pawan Deshmukh
Miss. Priyanka Manchanda
Miss. Shabna T. R
Mr. Surya Pratap Singh
Miss. Vintha Slipa Reddy

Last Updated: July 2, 2013
The project entitled "Optimizing Moodle LMS to Improve User Response Time" submitted by Miss. Akshaya Muralidharan, Mr. Mevin Dominic, Mr. Pawan Deshmukh, Miss. Priyanka Manchanda, Miss. Shabna T. R, Mr. Surya Pratap Singh, and Miss. Vintha Slipa Reddy is approved for Summer Internship 2013 programme from 9th May 2013 to 6th July 2013, at Department of Computer Science and Engineering, IIT Bombay.

Prof. Deepak B. Phatak
Dept of CSE, IITB
Principal Investigator

Mr. Nagesh Karmali
Dept of CSE, IITB
Project In-charge

Mr. D.B. Sathe
External Examiner

Place: IIT Bombay, Mumbai
Date: 3rd July 2013
Declaration

I declare that this written submission represents my ideas in my own words and where others’ ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Miss. Akshaya Muralidharan
National Institute of Technology, Surathkal

Mr. Mevin Dominic
National Institute of Technology, Calicut

Mr. Pawan Deshmukh
Indian School of Mines, Dhanbad

Miss. Priyanka Manchanda
Jaypee Institute Of Information Technology, Noida

Miss. Shabna T. R
Model Engineering College, Kerala

Mr. Surya Pratap Singh
National Institute of Technology, Trichy

Miss. Vintha Slipa Reddy
Lakireddy Bali Reddy College Of Engineering, Mylavaram

Date: ______________________
Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to Prof D.B. Pathak, Mr. Nagesh Karmali, Mr. Anoop Dobhal, Mr. Dhanajay Ambekar, Miss Firuza Aibara, Mr. Nilesh Birari, Mr Raju.B and Miss Silpa T. for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We would like to express my gratitude towards the members of IIT Bombay, for their kind co-operation and encouragement which help me in completion of this project.

Our thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.
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.

Moodle performance optimization can be achieved through different ways to reduce the user response time. One can change in the Moodle architecture or in the hardware/software used.

The current Moodle system faces a lot of performance issues. The database schema of Moodle 2.5 contains as many as 314 tables. The major concern in this project is the high response time of Moodle database.

To study the complex Moodle architecture and its response time for different activities of the user, different tools/softwares/hardware like JMeter, Apache and Lighttpd, SSD (Solid State Drive), various machines (both real and virtual) etc. were used.

Testing can be done mainly in two ways for Moodle, the first one is database testing (performed by running a script) for improving the latency that is incurred at the database server and the next one is by using JMeter, which helps to compare the user response times for different hardware and database tuning. Both of these were exploited to the boundary conditions in this project. Plenty of tests/experiments were done as a part of this project so as to determine the maximum threshold values of different parameters with different setups of hardware/software.

Another objective which has been taken up in this project is to try optimization by adding/writing useful plug-ins so that they can be effective in performance optimization. We have been successful to an extent in accomplishing the objective by implementing the techniques and experiments mentioned above.
# List of Figures

1.1 Moodle Versions .................................................. 3
1.2 Moodle Download Page[4] ........................................ 4

3.1 Users and Profile .................................................. 13
3.2 Users .............................................................. 14
3.3 Groups ............................................................ 15
3.4 Blocks .............................................................. 16
3.5 GradeBook ........................................................ 17

5.1 Average User Response Time (for SSD vs HDD) ............... 29
5.2 Average User Response Time for 1 iteration .................. 31
5.3 Average User Response Time for 10 iterations ............... 32
5.4 Average User Response Time for 100 iterations .............. 32
5.5 Average User Response Time for 1 user different iterations 34
5.6 Average User Response Time for 10 iterations with different number of users 35

6.1 Moodle Quiz Database ............................................. 41

7.1 Hard Disk[12] ....................................................... 46
7.3 Code Snippet for executing Queries from Query Log ........ 54
7.4 Tests in two systems SSD vs HDD - SSD ...................... 58
7.5 Tests in two systems SSD vs HDD - HDD ...................... 59
7.6 Results(in sec):SSD v/s HDD Test on Different Systems .... 62
7.7 SSD i5 Memory Configuration .................................... 62
7.8 HDD i5 Memory Configuration ................................. 62
7.9 HDD i3 Memory Configuration ................................... 63
7.10 clean_up.sh ....................................................... 64
7.11 testcode1on6.py ................................................. 65
7.12 code.py .......................................................... 66
7.13 testcode.py ...................................................... 67
7.15 System 1 Memory Configuration ............................... 68
7.16 System 1 CPU Information ...................................... 68
7.17 System 1 Result Execution Time .............................. 69
7.18 System 2 Processor Configuration ............................ 70
7.19 System 2 Memory Configuration .............................. 70
7.20 System 2 CPU Information ...................................... 71
LIST OF FIGURES

7.21 System 2 Result Execution Time ........................................ 72
7.22 System 3 Processor Configuration ........................................ 73
7.23 System 3 Memory Configuration ........................................... 73
7.24 System 3 CPU Information ............................................... 74
7.25 System 3 Result Execution Time ........................................ 75
7.26 System 4 Processor Configuration ........................................ 76
7.27 System 4 Memory Configuration ........................................... 76
7.28 System 4 CPU Information ............................................... 77
7.29 System 4 Result Execution Time ........................................ 78
7.30 Initial Configuration ....................................................... 79
7.31 Final Configuration ....................................................... 79
7.32 Results Using the Same System ......................................... 80
7.33 Modified Code for 100 Users ........................................... 82
7.34 HDD ............................................................................. 82
7.35 SSD ............................................................................. 82
8.1 Average Response Time (in ms) vs. Type of Timer .................... 87
8.2 Average Response Time (in ms) for 412 users ......................... 88
8.3 Average Response Time (in ms) for 413 users ......................... 89
8.4 Average Response Time (in ms) vs Number of Users .............. 89
8.5 Average Response Time (in ms) for Lighttpd vs Apache .......... 90
8.6 Average Response Time (in ms) for Lighttpd vs Apache .......... 91
8.7 Generate JMeter Script ...................................................... 92
8.8 Open JMeter Script ......................................................... 93
8.9 Summary Report .............................................................. 93
8.10 JMeter Testing For Real Time Quiz Plugin Fig1 ....................... 97
8.11 JMeter Testing For Real Time Quiz Plugin Fig2 ....................... 97
8.12 JMeter Testing For Real Time Quiz Plugin Fig3 ....................... 98
8.13 JMeter Testing For Real Time Quiz Plugin Fig4 ....................... 98
8.14 JMeter Testing For Real Time Quiz Plugin Fig5 ....................... 99
9.1 Working of Long Polling .................................................... 102
9.2 Snippet of quizdata.php ..................................................... 104
9.3 Snippet of view_student.js ............................................... 104
9.4 Snippet of quizdata.php ..................................................... 105
9.5 In view_student.js in line 502 ............................................ 105
9.6 line no 365,quizdata.php .................................................. 105
9.7 line no 406,quizdata.php .................................................. 106
9.8 Three questions have been set up by the teacher .................. 106
9.9 Specify a name for the quiz and start it to enable the students to take it .................................................. 107
9.10 After the student connects, it waits for the question to be sent ... 107
9.11 The teacher waits until all the students are connected .......... 108
9.12 When the teacher presses the next button, it becomes visible to the student and he has 30 seconds to answer it .... 108
9.13 The question is not sent until the teacher presses the next button. Long polling goes on in this period ............... 109
9.14 After pressing next button the question gets displayed to student within next 3 seconds .................................. 109
9.15 The first question gets displayed and the student has 30 seconds to answer it, no polling goes on in this period. 110
9.16 After answering all questions the student gets to know his performance. 110
9.17 Similarly all questions are sent by the teacher and answered by the student. We can view all responses by clicking on the view response tab. 111
9.18 90 users: default method. 111
9.19 90 users: long-polling. 112
10.1 60 users: default quiz. 115
10.2 60 users: Random batch-mode connectivity algorithm. 115
10.3 75 users: Random batch-mode connectivity algorithm. 116
11.1 Create Policy 120
11.2 Add Context 121
11.3 Edit Policy 122
11.4 Resquence 122
11.5 Database Schema Screenshot 1 123
11.6 Database Schema Screenshot 2 124
11.7 Database Schema Screenshot 3 125
List of Tables

5.1 Average User Response Time on HDD vs SSD(in s) .................................................. 28
5.2 Average User Response Time for 1 iteration .............................................................. 30
5.3 Average User Response Time for 10 iterations ............................................................ 30
5.4 Average User Response Time for 100 iterations ......................................................... 31
5.5 Average User Response Time for 1 user different iterations ....................................... 33
5.6 Average User Response Time for 10 iterations with different number of users ............... 35
5.7 Average User Response Time with and without DNS Cache for 1 user ......................... 36
5.8 Average User Response Time for 1 user for the scenarios mentioned in Section 5.3.3.3 .......................................................... 37
5.9 Average User Response Time for 1 user with and without compression ....................... 38
7.1 Execution Time on HDD vs SSD(in s) ................................................................. 51
7.2 Results(in sec):SSD v/s HDD for Quiz Module ......................................................... 55
7.3 Results(in sec):SSD v/s HDD Hardware Scaling ......................................................... 81
7.4 Results(in sec):SSD v/s HDD Test for 100 Users giving a quiz simultaneously ................ 83
8.1 Activity vs Response Time(in ms) .............................................................................. 94
11.1 Explanation of various tags defined in Policy (xml) file[23] ........................................ 119
# Contents

1 Introduction to Moodle  
   1.1 Introduction ................................................... 1  
   1.2 Moodle .......................................................... 1  
   1.3 Features .......................................................... 1  
   1.4 Installation ..................................................... 2  
      1.4.1 Prerequisites ................................................. 3  
         1.4.1.1 Hardware Requirements ............................... 3  
         1.4.1.2 Installing in a LAMP Environment ................. 3  
         1.4.1.3 Downloading Moodle ................................. 3  
         1.4.1.4 Creating the moodle database and the data directory. 4  
         1.4.1.5 Running Installer Script ............................ 4  
         1.4.1.6 Setting up the cron process ......................... 4  
         1.4.1.7 Updating Moodle ...................................... 5  
         1.4.1.8 Manual Update ...................................... 5  
2 Moodle System .................................................... 7  
   2.1 Moodle architecture ............................................ 7  
      2.1.1 Moodle Layer .............................................. 7  
      2.1.2 FileManagement ........................................... 7  
3 Moodle Database .................................................. 9  
   3.1 Database Structure ............................................. 9  
   3.2 Moodle Database Tables ....................................... 9  
      3.2.1 Configuration ............................................. 10  
      3.2.2 Users and their profiles ................................ 11  
      3.2.3 Groups .................................................... 15  
      3.2.4 The logging system ..................................... 16  
      3.2.5 Blocks .................................................... 16  
      3.2.6 Events .................................................... 17  
      3.2.7 Backup and restore ..................................... 17  
      3.2.8 Statistics ................................................ 17  
      3.2.9 GradeBook ................................................ 17  
      3.2.10 Question Bank .......................................... 18  
      3.2.11 Messaging system ..................................... 18  
      3.2.12 Moodle Network ....................................... 19  
      3.2.13 Caching .................................................. 19  

ix
4 Optimization to Database
4.1 Table Size Problem ........................................ 21
4.2 Table Indexes ............................................ 21
4.3 Useful SQL Queries ...................................... 21
4.4 Removal of irrelevant php functions from moodle ........................................ 24
  4.4.1 Possible removal of unused functions in shortanswer/questiontype.php ........... 24
  4.4.2 Unused functions from questionarre in locallib.php ............................... 25
  4.4.3 Remove the grade item for an activity if the grade for that activity is always zero ........................................................................................................... 25
  4.4.4 Deprecated functions still allowed ................................................... 25
5 Moodle Optimization Techniques ........................................ 27
5.1 Objective .................................................................. 27
5.2 Hardware Optimization ........................................... 27
  5.2.1 Load Testing on SSD vs HDD ........................................... 27
    5.2.1.1 Experiment Setup .............................................. 27
    5.2.1.2 Results ....................................................... 28
    5.2.1.3 Graph ........................................................ 29
    5.2.1.4 Conclusion .................................................... 29
5.3 Front-End Optimization Techniques .................................. 30
  5.3.1 Reduce Number of HTTP Requests by using Image Maps ....................... 30
    5.3.1.1 Experiment Setup .............................................. 30
    5.3.1.2 Results ....................................................... 30
    5.3.1.3 Graph ........................................................ 31
  5.3.2 Caching By Using Far Future Expires Header ...................................... 33
    5.3.2.1 Experiment 1 Setup .............................................. 33
    5.3.2.2 Experiment 1 Results ......................................... 33
    5.3.2.3 Experiment 1 Graphs .......................................... 34
    5.3.2.4 Experiment 2 Setup .............................................. 34
    5.3.2.5 Experiment 2 Results ......................................... 35
    5.3.2.6 Experiment 2 Graphs .......................................... 35
    5.3.2.7 Conclusion .................................................... 35
  5.3.3 Reducing DNS Lookups ........................................ 36
    5.3.3.1 Experiment 1 .................................................. 36
    5.3.3.2 Experiment 1 Results ......................................... 36
    5.3.3.3 Experiment 2 .................................................. 36
    5.3.3.4 Experiment 2 Results ......................................... 37
    5.3.3.5 Conclusion .................................................... 37
  5.3.4 Gzip Components .............................................. 37
    5.3.4.1 Experiment .................................................. 38
    5.3.4.2 Results ....................................................... 38
  5.3.5 Deactivating ETags ........................................... 38
    5.3.5.1 Experiment .................................................. 39
    5.3.5.2 Results ....................................................... 39
    5.3.5.3 Conclusion .................................................... 39
5.4 Conclusion ................................................................ 39
CONTENTS

6 Quiz Module in Moodle 41
   6.1 Quiz Database ........................................... 41
   6.2 Question Types in Moodle ................................. 42
   6.3 SQL Query Log for Quiz .................................. 43
      6.3.1 Log Set Up ........................................ 43

7 SSD v/s HDD for the 'Moodle' DATABASE 45
   7.1 Introduction .............................................. 45
      7.1.1 Read and Write ..................................... 45
      7.1.2 HDD .................................................. 45
      7.1.3 PCIe SSDs ........................................... 46
      7.1.4 Latency .............................................. 47
      7.1.5 Price ................................................ 47
      7.1.6 Speed ............................................... 48
      7.1.7 Reliability ......................................... 48
   7.2 Experiments Performed .................................... 48
      7.2.1 Experiment Setup ....................................... 48
      7.2.2 Experiment 1 ......................................... 48
      7.2.3 Experiment 2 ......................................... 49
         7.2.3.1 Objective ...................................... 49
         7.2.3.2 Setup and Testing ................................. 49
      7.2.4 Results (in seconds) of Experiment 2 .................. 49
         7.2.4.1 JOIN ............................................. 50
         7.2.4.2 LEFT JOIN ...................................... 50
         7.2.4.3 RIGHT JOIN ..................................... 50
         7.2.4.4 UNION ........................................... 51
      7.2.5 Experiment 3 : SSD v/s HDD for Quiz module ....... 51
         7.2.5.1 Objective ...................................... 51
         7.2.5.2 Quiz ............................................. 51
         7.2.5.3 Moodle Used ..................................... 52
         7.2.5.4 Quiz- query log file .............................. 52
         7.2.5.5 Cleaning Up log file ............................. 52
         7.2.5.6 Commands Used ................................... 52
         7.2.5.7 Reading The Queries Script ....................... 53
         7.2.5.8 Query Log - Final v/s Initial file ............... 53
         7.2.5.9 Code Snippet .................................... 54
         7.2.5.10 Results ......................................... 54
         7.2.5.11 Latency and Specification ......................... 55
         7.2.5.12 Effect of cache minimised ......................... 55
         7.2.5.13 Tests in two systems SSD vs HDD .................. 58
         7.2.5.14 THE DATABASE TABLES (11) INVOLVED IN GIV-.... 60
      7.2.6 Experiment-4 SSD v/s HDD for viewing a forum ....... 60
         7.2.6.1 Setup ............................................. 60
         7.2.6.2 Question Raised ................................... 60
         7.2.6.3 Steps ............................................. 61
         7.2.6.4 Result ............................................ 61
         7.2.6.5 Memory Configurations ............................ 62
7.2.7 Experiment 5 - SSD v/s HDD - A TEST ON DIFFERENT HARDWARE

7.2.7.1 Objective .............................................. 63
7.2.7.2 Systems Used ........................................ 63
7.2.7.3 Quiz Details ........................................ 63
7.2.7.4 Steps .................................................. 63
7.2.7.5 System 1 Configuration ............................... 67
7.2.7.6 System 2 Configuration ............................... 70
7.2.7.7 System 3 Configuration ............................... 73
7.2.7.8 System 4 Configuration - Best Among All four .... 76
7.2.7.9 A Slight Change On The Current System ........ 78

7.2.8 Experiment 6 .............................................. 81
7.2.8.1 Objective .............................................. 81
7.2.8.2 Results ................................................ 82
7.2.8.3 Factor .................................................. 83

7.3 Conclusion .................................................. 83
7.3.1 High Performance ...................................... 83

8 Testing Moodle with Jmeter ................................ 85
8.1 Introduction ................................................ 85
8.2 Testing by integrating different severs with Moodle ...
8.2.1 Common Experimental Setup .......................... 85
8.2.1.1 Installation ........................................ 85
8.2.1.2 Testing ............................................. 85
8.2.2 To integrate with another server ................. 86
8.2.2.1 Setting up Moodle on Lighttpd .................... 86
8.2.2.2 Testing ............................................. 86

8.3 Experiments ................................................. 86
8.3.1 Experiment 1 ............................................ 86
8.3.1.1 Objective ........................................... 86
8.3.1.2 Procedure ........................................... 86
8.3.1.3 TEST CONDITIONS ................................. 87
8.3.1.4 Results .............................................. 87
8.3.1.5 Conclusion .......................................... 87
8.3.2 Experiment 2 ............................................ 87
8.3.2.1 OBJECTIVE .......................................... 87
8.3.2.2 PROCEDURE ......................................... 87
8.3.2.3 TEST CONDITIONS ................................. 88
8.3.2.4 Results .............................................. 88
8.3.2.5 Conclusion .......................................... 89
8.3.3 Experiment 3 ............................................ 90
8.3.3.1 Objective ........................................... 90
8.3.3.2 Test Conditions .................................... 90
8.3.3.3 Results ............................................. 90
8.3.3.4 Conclusion .......................................... 90
8.3.4 Experiment 4 ............................................ 91
8.3.4.1 Objective ........................................... 91
8.3.4.2 Procedure ........................................... 91
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.4.3 Test Conditions</td>
<td>91</td>
</tr>
<tr>
<td>8.3.4.4 Results</td>
<td>91</td>
</tr>
<tr>
<td>8.3.4.5 Conclusion</td>
<td>92</td>
</tr>
<tr>
<td>8.4 Jmeter Testing on Moodle 2.5</td>
<td>92</td>
</tr>
<tr>
<td>8.4.1 Experimental Setup</td>
<td>92</td>
</tr>
<tr>
<td>8.4.2 Comparing Different Activities</td>
<td>94</td>
</tr>
<tr>
<td>8.5 Quiz Activity</td>
<td>94</td>
</tr>
<tr>
<td>8.5.1 Increasing loop count</td>
<td>94</td>
</tr>
<tr>
<td>8.5.2 Increasing the number of users</td>
<td>94</td>
</tr>
<tr>
<td>8.5.3 Login</td>
<td>94</td>
</tr>
<tr>
<td>8.5.3.1 Results</td>
<td>94</td>
</tr>
<tr>
<td>8.5.4 SQL QUERIES GENERATED FOR JMETER</td>
<td>95</td>
</tr>
<tr>
<td>8.5.4.1 Experimental Setup</td>
<td>95</td>
</tr>
<tr>
<td>8.5.4.2 Result</td>
<td>95</td>
</tr>
<tr>
<td>8.5.5 Jmeter Testing For Real Time Quiz Plugin</td>
<td>96</td>
</tr>
<tr>
<td>8.5.5.1 Steps</td>
<td>96</td>
</tr>
<tr>
<td>8.5.5.2 Result</td>
<td>99</td>
</tr>
<tr>
<td>9 Implementation of Long Polling to realtime quiz plugin</td>
<td>101</td>
</tr>
<tr>
<td>9.1 Introduction</td>
<td>101</td>
</tr>
<tr>
<td>9.1.1 Concept of Long Polling</td>
<td>101</td>
</tr>
<tr>
<td>9.1.2 Implementation over a simple php based quiz</td>
<td>101</td>
</tr>
<tr>
<td>9.1.3 Real Time Quiz Plugin</td>
<td>102</td>
</tr>
<tr>
<td>9.2 Functionality</td>
<td>102</td>
</tr>
<tr>
<td>9.2.1 Plugin Details</td>
<td>102</td>
</tr>
<tr>
<td>9.2.1.1 Working</td>
<td>103</td>
</tr>
<tr>
<td>9.3 Description of code for Initial Implementation:(Default Plugin)</td>
<td>104</td>
</tr>
<tr>
<td>9.4 LONG POLLING IMPLEMENTATION</td>
<td>105</td>
</tr>
<tr>
<td>9.5 Snapshots</td>
<td>106</td>
</tr>
<tr>
<td>9.6 Experimental Results</td>
<td>111</td>
</tr>
<tr>
<td>9.6.1 No. Of requests</td>
<td>111</td>
</tr>
<tr>
<td>9.6.2 Throughput</td>
<td>111</td>
</tr>
<tr>
<td>9.7 Conclusion</td>
<td>112</td>
</tr>
<tr>
<td>10 Implementation of Random batch-mode connectivity algorithm</td>
<td>113</td>
</tr>
<tr>
<td>10.1 Objective</td>
<td>113</td>
</tr>
<tr>
<td>10.2 Introduction to random number generation</td>
<td>113</td>
</tr>
<tr>
<td>10.3 Implementation in moodle quiz</td>
<td>114</td>
</tr>
<tr>
<td>10.3.1 Benefit</td>
<td>114</td>
</tr>
<tr>
<td>10.3.2 Settings to use</td>
<td>114</td>
</tr>
<tr>
<td>10.4 Results after Jmeter Testing</td>
<td>115</td>
</tr>
<tr>
<td>10.4.1 System specification</td>
<td>115</td>
</tr>
<tr>
<td>10.4.2 Experimentations over Jmeter</td>
<td>115</td>
</tr>
<tr>
<td>11 POLICY ENFORCEMENT FRAMEWORK FOR MOODLE</td>
<td>117</td>
</tr>
<tr>
<td>11.1 Introduction</td>
<td>117</td>
</tr>
<tr>
<td>11.2 Policy Enforcement Framework</td>
<td>117</td>
</tr>
<tr>
<td>11.2.1 Context Attributes</td>
<td>118</td>
</tr>
<tr>
<td>11.2.2 Database Schema</td>
<td>118</td>
</tr>
</tbody>
</table>
CONTENTS

11.3 Policy Enforcement Interface ........................................ 120

11.3.1 Create Policy ...................................................... 120

11.3.2 Add context ....................................................... 121

11.3.3 Edit Policy ........................................................ 121

11.3.4 Resequence ......................................................... 122

11.3.5 Generate XML Files ................................................ 123

11.4 Database Schema Screenshots ........................................ 123

11.5 Conclusion ............................................................ 125

12 Future Scope ........................................................... 127
Chapter 1

Introduction to Moodle

1.1 Introduction

Moodle is a Virtual Learning Environment (VLE). As per statistics of June 2013 it had of 83,008 registered and verified sites, serving 70,696,570 users in 7.5+ million courses with 1.2+ million teachers. [1].

1.2 Moodle

Moodle means Modular Object Oriented Dynamic Learning Environment which is used to help educators in creating online courses. It is a free source e-learning. It is also known 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.

1.3 Features

Moodle has several features. It is a typical of an e-learning platform, plus some innovations. Moodle is same to a learning management system. Moodle can be used in environments such as in education, training and development, business settings. Some typical features of Moodle are:[2]

- 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)

• Enrolment methods

• Content filters

Many freely available third-party Moodle plugins make use of this infrastructure. Moodle users can use PHP to write and contribute new modules. Moodle’s development has been assisted by the work of open source programmers. 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.

1.4 Installation

Users can install Moodle from any source like, 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. Fig. 1.1 shows all the versions of Moodle which have been released.
1.4. INSTALLATION

1.4.1 Prerequisites

There are a number of hardware and software requirements for installing Moodle.\[3\].

1.4.1.1 Hardware Requirements

These requirements apply if you host Moodle yourself or if it is hosted on an external server (shared, virtual, dedicated, or clustered). On cheaper hosting packages, the hardware configuration is often insufficient to run Moodle efficiently.

- Disk space: Moodle takes up between 150 and 200 MB of disk space. However, this only provides you with an empty system and does not take into account the space you require for any learning resources. The faster the disks, the better. RAIDed disks are recommended, but are not essential on smaller installations.

- Memory: The (absolute) minimum requirement is 256 MB for a single-user instance, but more is necessary in a multiuser setup. A good rule of thumb is to have 1 GB of RAM for every 30-50 concurrent users. You have to double this calculation on Windows-based systems due to the higher overhead of the operating system.

1.4.1.2 Installing in a LAMP Environment

Moodle is developed in Linux using Apache, MySQL, and PHP (known as the LAMP platform). This is the preferred environment. There is ongoing debate whether PostgreSQL is the more suitable database option, but MySQL is the system most administrators are most familiar with.

1.4.1.3 Downloading Moodle

We can download moodle from www.moodle.org to download moodle as we can find many distributions.\[4\].

![Figure 1.1: Moodle Versions](image-url)
1.4.1.4 Creating the moodle database and the data directory

Moodle requires a database where it can store its information. While it is possible to share an existing database, it is highly recommended to create a separate database for Moodle. This can either be done via a web interface, as provided by hosted servers, or via the Unix command line.

1.4.1.5 Running Installer Script

The installer script performs two main actions: populating the database and creating the configuration file. The configuration file is config.php. The Moodle installer is initiated by entering the URL of wwwroot (the location where you copied Moodle) into your web browser; Moodle will recognize that it hasn’t been installed yet and start the process automatically.

The Moodle installer has to set a session cookie. If your browser has been configured to trigger a warning, make sure you accept that cookie. The first screen lets you choose the language to be used during installation. This is not the locale used for Moodle, only the language for the installation.

1.4.1.6 Setting up the cron process

Moodle has to perform a number of background tasks on a regular basis. The script that performs these tasks is known as a cron script and is executed by the so-called cron process. An entire page has been dedicated to this in the Moodle documentation; you can find it at docs.moodle.org/en/Cron. It is important that you set up the cron process. Otherwise, any timed Moodle features, such as scheduled backups, sending forum notifications, statistics processing, and so on, will not work.

The script cron.php is located in the admin directory and can be triggered manually through a web browser (unless your security settings have been changed).
1.4.1.7 Updating Moodle

Moodle is updated constantly, which is common practice in open source development environments. A new version containing resolved bug fixes is created every night and, as mentioned earlier, a fully-tested version is released on a weekly basis. There is usually no need to install updates every week. However, Moodle should be upgraded when:

- Security patches have been issued
- New features have been added
- Bugs have been fixed that affect your setup
- A major new update is released (every 6 months)

There are two ways Moodle systems can be updated either run updates manually (using the web interface or the CLI) or stay up-to-date using the CVS or GIT commands.

1.4.1.8 Manual Update

The high-level process for updating a Moodle system is as follows:

1. Creating a backup.
2. Creating a new Moodle system
3. Installing the update.

Moodle updates can also be run using the CLI. After back up data and updated to the latest version run the following script:

```
sudo u www-root /usr/bin/php admin/cli/upgrade.php –non-interactive
```
CHAPTER 1. INTRODUCTION TO MOODLE
Chapter 2

Moodle System

Here we are going to have a look at the building blocks of the learning platform. These are the foundation on which Moodle is built.

2.1 Moodle architecture

2.1.1 Moodle Layer

Moodle distinguishes between code (PHP, HTML, and CSS) and data. Moodle libraries, modules (such as resources and activities), blocks, plugins, and other entities are represented in code. It is stored in a filesystem in a Moodle directory dirroot. The code includes all elements that deal with the backend and frontend operations.

Moodle courses, users, roles, groups, grades, and other data such as learning resources added by teachers, forum posts added by students, and system settings added by the administrator are mostly stored in the Moodle database. However, files such as user pictures or uploaded assignments, are stored in another Moodle directory, known as moodledata.

2.1.2 FileManagement

Dealing with files in web-based applications is always tough. While Moodle provides a user interface to perform this task, it is sometimes necessary that, as the administrator, you will have to bypass this mechanism and use other means. First though, let us look at the built-in file handling that is also the one used by students and teachers.

- Moodle’s file management
- Web host file management
- Bugs have been fixed that affect your setup
- File management via the File system repository
Chapter 3

Moodle Database

Moodle is not a single complex application. It is an aggregation of different plugins. The scripts in moodle are written in PHP. So only some plugins 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. But neither PHP nor moodle architecture doesn’t have clear separation of UI layer. Every interaction has logic in it. Transaction script organizes all the script as a single procedure. Later common sub-tasks can be broken into sub-procedures.

3.1 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 by 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 RDBMS.

3.2 Moodle Database 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

3.2.1 Configuration

config
"id" BIGINT NOT NULL,
"name" NVARCHAR(255) NOT NULL,
"value" NVARCHAR(-1) NOT NULL

config log
"id" BIGINT NOT NULL,
"userid" BIGINT NOT NULL,
"timemodified" BIGINT NOT NULL,
"plugin" NVARCHAR(100) NULL,
"name" NVARCHAR(100) NOT NULL,
"value" NVARCHAR(-1) NULL,
"oldvalue" NVARCHAR(-1) NULL

config plugin
"id" BIGINT NOT NULL,
"plugin" NVARCHAR(100) NOT NULL,
"name" NVARCHAR(100) NOT NULL,
"value" NVARCHAR(-1) NOT NULL
Users and their profiles

user "id" BIGINT NOT NULL,
"auth" NVARCHAR(20) NOT NULL,
"confirmed" SMALLINT NOT NULL,
"policyagreed" SMALLINT NOT NULL,
"deleted" SMALLINT NOT NULL,
"suspended" SMALLINT NOT NULL,
"mnethostid" BIGINT NOT NULL,
"username" NVARCHAR(100) NOT NULL,
"password" NVARCHAR(32) NOT NULL,
"idnumber" NVARCHAR(255) NOT NULL,
"firstname" NVARCHAR(100) NOT NULL,
"lastname" NVARCHAR(100) NOT NULL,
"email" NVARCHAR(100) NOT NULL,
"emailstop" SMALLINT NOT NULL,
"icq" NVARCHAR(15) NOT NULL,
"skype" NVARCHAR(50) NOT NULL,
"yahoo" NVARCHAR(50) NOT NULL,
"aim" NVARCHAR(50) NOT NULL,
"msn" NVARCHAR(50) NOT NULL,
"phone1" NVARCHAR(20) NOT NULL,
"phone2" NVARCHAR(20) NOT NULL,
"institution" NVARCHAR(40) NOT NULL,
"department" NVARCHAR(30) NOT NULL,
"address" NVARCHAR(70) NOT NULL,
"city" NVARCHAR(120) NOT NULL,
"country" NVARCHAR(2) NOT NULL,
"lang" NVARCHAR(30) NOT NULL,
"theme" NVARCHAR(50) NOT NULL,
"timezone" NVARCHAR(100) NOT NULL,
"firstaccess" BIGINT NOT NULL,
"lastaccess" BIGINT NOT NULL,
"lastlogin" BIGINT NOT NULL,
"currentlogin" BIGINT NOT NULL,
"lastip" NVARCHAR(45) NOT NULL,
"secret" NVARCHAR(15) NOT NULL,
"picture" BIGINT NOT NULL,
"url" NVARCHAR(255) NOT NULL,
"description" NVARCHAR(-1) NULL,
"descriptionformat" SMALLINT NOT NULL,
"mailformat" SMALLINT NOT NULL,
"maildigest" SMALLINT NOT NULL,
"maildisplay" SMALLINT NOT NULL,
"htmleditor" SMALLINT NOT NULL,
"autosubscribe" SMALLINT NOT NULL,
"trackforums" SMALLINT NOT NULL,
"timecreated" BIGINT NOT NULL,
"timemodified" BIGINT NOT NULL,
"trustbitmask" BIGINT NOT NULL,
"imagealt" NVARCHAR(255) NULL,
"screenreader" SMALLINT NOT NULL

user enrolments
"id" BIGINT NOT NULL,
"status" BIGINT NOT NULL,
"enrolid" BIGINT NOT NULL,
"userid" BIGINT NOT NULL,
"timestart" BIGINT NOT NULL,
"timeend" BIGINT NOT NULL,
"modifierid" BIGINT NOT NULL,
"timecreated" BIGINT NOT NULL,
"timemodified" BIGINT NOT NULL

user info category
"id" BIGINT NOT NULL,
"name" NVARCHAR(255) NOT NULL,
"sortorder" BIGINT NOT NULL

user info data
"id" BIGINT NOT NULL,
"userid" BIGINT NOT NULL,
"fieldid" BIGINT NOT NULL,
"data" NVARCHAR(-1) NOT NULL,
"dataformat" SMALLINT NOT NULL

user info field
"id" BIGINT NOT NULL,
"shortname" NVARCHAR(255) NOT NULL,
"name" NVARCHAR(-1) NOT NULL,
"datatype" NVARCHAR(255) NOT NULL,
"description" NVARCHAR(-1) NULL,
"descriptionformat" SMALLINT NOT NULL,
"categoryid" BIGINT NOT NULL,
"sortorder" BIGINT NOT NULL,
"required" SMALLINT NOT NULL,
"locked" SMALLINT NOT NULL,
"visible" SMALLINT NOT NULL,
"forceunique" SMALLINT NOT NULL,
"signup" SMALLINT NOT NULL,
"defaultdata" NVARCHAR(-1) NULL,
"defaultdataformat" SMALLINT NOT NULL,
"param1" NVARCHAR(-1) NULL,
"param2" NVARCHAR(-1) NULL,
"param3" NVARCHAR(-1) NULL,
"param4" NVARCHAR(-1) NULL,
"param5" NVARCHAR(-1) NULL
3.2. MOODLE DATABASE TABLES

user last access
This is separated from the user table for performance reasons
"id" BIGINT NOT NULL,
"userid" BIGINT NOT NULL,
"courseid" BIGINT NOT NULL,
"timeaccess" BIGINT NOT NULL

user preferences
"id" BIGINT NOT NULL,
"userid" BIGINT NOT NULL,
"name" NVARCHAR(255) NOT NULL,
"value" NVARCHAR(1333) NOT NULL

user private key
"id" BIGINT NOT NULL,
"script" NVARCHAR(128) NOT NULL,
"value" NVARCHAR(128) NOT NULL,
"userid" BIGINT NOT NULL,
"instance" BIGINT NULL,
"iprestriction" NVARCHAR(255) NULL,
"validuntil" BIGINT NULL,
"timecreated" BIGINT NULL

Fig. 3.1 shows all users and profile tables ER Diagram.

![Figure 3.1: Users and Profile](image-url)
Fig. 3.2 shows all users tables ER Diagram.

Figure 3.2: Users
3.2. Moodle Database Tables

3.2.3 Groups

Fig. 3.3 shows all groups tables ER Diagram.

```
groups
  "id" BIGINT NOT NULL,
  "courseid" BIGINT NOT NULL,
  "name" NVARCHAR(254) NOT NULL,
  "description" NVARCHAR(-1) NULL,
  "descriptionformat" SMALLINT NOT NULL,
  "enrolmentkey" NVARCHAR(50) NULL,
  "picture" BIGINT NOT NULL,
  "hidepicture" SMALLINT NOT NULL,
  "timecreated" BIGINT NOT NULL,
  "timemodified" BIGINT NOT NULL,
  "idnumber" NVARCHAR(100) NOT NULL

members
  "id" BIGINT NOT NULL,
  "groupid" BIGINT NOT NULL,
  "userid" BIGINT NOT NULL,
  "timeadded" BIGINT NOT NULL

groupings
  "id" BIGINT NOT NULL,
  "courseid" BIGINT NOT NULL,
  "name" NVARCHAR(255) NOT NULL,
```
"description" NVARCHAR(-1) NULL,
"descriptionformat" SMALLINT NOT NULL,
"configdata" NVARCHAR(-1) NULL,
"timecreated" BIGINT NOT NULL,
"timemodified" BIGINT NOT NULL,
"idnumber" NVARCHAR(100) NOT NULL

groupings groups
"id" BIGINT NOT NULL,
"groupingid" BIGINT NOT NULL,
"groupid" BIGINT NOT NULL,
"timeadded" BIGINT NOT NULL

3.2.4 The logging system

tab
date display

3.2.5 Blocks

block
block instance
block rss client
block pinned
block search documents

Fig. 3.4 shows all blocks tables ER Diagram.
3.2. MOODLE DATABASE TABLES

3.2.6 Events

- event
- events handlers
- events queue
- events queue handlers

3.2.7 Backup and restore

- backup config
- backup courses
- backup files
- backup ids
- backup log

3.2.8 Statistics

- stats daily
- stats monthly
- stats user daily
- stats user monthly
- stats user weekly
- stats weekly

3.2.9 GradeBook

Fig. 3.5 shows all gradebook tables ER Diagram.

Figure 3.5: GradeBook
CHAPTER 3. MOODLE DATABASE

3.2.10 Question Bank

question answers
question attempts
question categories
question sessions
question states

3.2.11 Messaging system

message "id" BIGINT NOT NULL,
"useridfrom" BIGINT NOT NULL,
"useridto" BIGINT NOT NULL,
"subject" NVARCHAR(-1) NULL,
"fullmessage" NVARCHAR(-1) NULL,
"fullmessageformat" SMALLINT NULL,
"fullmessagehtml" NVARCHAR(-1) NULL,
"smallmessage" NVARCHAR(-1) NULL,
"notification" SMALLINT NULL,
"contexturl" NVARCHAR(-1) NULL,
"contexturlname" NVARCHAR(-1) NULL,
"timecreated" BIGINT NOT NULL

message contacts
"id" BIGINT NOT NULL,
"userid" BIGINT NOT NULL,
"contactid" BIGINT NOT NULL,
"blocked" SMALLINT NOT NULL

message processors
"id" BIGINT NOT NULL,
"name" NVARCHAR(166) NOT NULL,
"enabled" SMALLINT NOT NULL

message providers
"id" BIGINT NOT NULL,
"name" NVARCHAR(100) NOT NULL,
"component" NVARCHAR(200) NOT NULL,
"capability" NVARCHAR(255) NULL

message read
"id" BIGINT NOT NULL,
"useridfrom" BIGINT NOT NULL,
"useridto" BIGINT NOT NULL,
"subject" NVARCHAR(-1) NULL,
"fullmessage" NVARCHAR(-1) NULL,
"fullmessageformat" SMALLINT NULL,
"fullmessagehtml" NVARCHAR(-1) NULL,
"smallmessage" NVARCHAR(-1) NULL,
"notification" SMALLINT NULL,
"contexturl" NVARCHAR(-1) NULL,
3.2. MOODLE DATABASE TABLES

"contexturlname" NVARCHAR(-1) NULL,
"timecreated" BIGINT NOT NULL,
"timeread" BIGINT NOT NULL

message working
"id" BIGINT NOT NULL,
"unreadmessageid" BIGINT NOT NULL,
"processorid" BIGINT NOT NULL

3.2.12 Moodle Network

mnet application
mnet enrol assignments
mnet enrol course
mnet host
mnet host2service
mnet log
mnet rpc
mnet service
mnet service2rpc
mnet session
mnet sso access control

3.2.13 Caching

cache filters
cache flags
cache text
Chapter 4

Optimization to Database

4.1 Table Size Problem

mdl_backup_controllers table 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.

```bash
!/bin/bash mysql −u YOURDATABASEUSERHERE −pYOURPASSWORDHERE moodle < < EOF
DELETE FROM mdl_backup_controllers WHERE timecreated < unix_timestamp(now()) – interval 7 day) EOF
```

Put in your database username after the −u and your database password after −p (no space here).

If your database is not called moodle, then change this to the appropriate name. This will delete out all controllers over a week old.

4.2 Table Indexes

Indexes are required to speed up the log table.

Example

<table>
<thead>
<tr>
<th>Keyname</th>
<th>Type</th>
<th>Cardinality</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY PRIMARY</td>
<td>PRIMARY</td>
<td>28892057</td>
<td>id</td>
</tr>
<tr>
<td>mdl_log_coumodact(ix</td>
<td>INDEX</td>
<td>179453</td>
<td>course moduleaction</td>
</tr>
<tr>
<td>mdl_log_tim(ix</td>
<td>INDEX</td>
<td>14446028</td>
<td>time</td>
</tr>
<tr>
<td>mdl_log_act(ix</td>
<td>INDEX</td>
<td>494</td>
<td>action</td>
</tr>
<tr>
<td>mdl_log_usecou(ix</td>
<td>INDEX</td>
<td>291838</td>
<td>userid</td>
</tr>
<tr>
<td>mdl_log_cmi(ix</td>
<td>INDEX</td>
<td>202042</td>
<td>cmid</td>
</tr>
</tbody>
</table>

When searching for a column that have no index like ip or info it becomes very slow. Default indexes are also present which improve the speed of data retrieval operations on a database table at the cost of slower writes and the use of more storage space.

4.3 Useful SQL Queries

1. Counting up the modules added to a given course

   ```sql
   SELECT modules.name, COUNT(c_modules.id) AS "Module Count"
   FROM mdl_course_modules c_modules
   ```
INNER JOIN mdl_modules modules ON c_modules.module = modules.id
WHERE
c_modules.course = 560
AND c_modules.visible = 1
GROUP BY
modules.name

2. Select users who have not logged in for over 180 days (but not those who have never logged in)
SELECT *
FROM mdl_user
WHERE lastlogin < UNIX_TIMESTAMP(DATE_SUB(NOW(), INTERVAL 180 DAY))
AND lastlogin != 0
AND lastaccess < UNIX_TIMESTAMP(DATE_SUB(NOW(), INTERVAL 180 DAY))
AND deleted = 0

3. Delete users who have not logged in for over 180 days (but not those who have never logged in)
UPDATE mdl_user
SET deleted=1
WHERE lastlogin < UNIX_TIMESTAMP(DATE_SUB(NOW(), INTERVAL 180 DAY))
AND lastlogin != 0
AND lastaccess < UNIX_TIMESTAMP(DATE_SUB(NOW(), INTERVAL 180 DAY))
AND deleted = 0

4. Select users who have NEVER logged in
SELECT id, concat( firstname, " ", lastname ) AS name, lastaccess, lastlogin,
currentlogin
FROM mdl_user
WHERE currentlogin = 0
AND lastlogin = 0
AND lastaccess = 0
AND deleted = 0

5. Delete users who have NEVER logged in
UPDATE mdl_user
SET deleted=1
WHERE currentlogin = 0
AND lastlogin = 0
AND lastaccess = 0
AND deleted = 0

6. Number of views (hits) per student in a course from 2010.
SELECT u.username AS Username, u.firstname AS "First Name", u.lastname AS
"Last Name", u.email AS Email, count(l.userid) AS Views
FROM 'prefix_log' l, 'prefix_user' u, 'prefix_role_assignments' r
WHERE l.course=1965
AND l.userid = u.id
AND l.time > UNIX_TIMESTAMP('20100101 00:00:00')
AND r.contextid= 0
SELECT id
FROM prefix_context
WHERE contextlevel=50 AND instanceid=l.course
)AND r.roleid=5
AND r.userid = u.id
GROUP BY l.userid
ORDER BY Views

7. Gives a count for resources and activities in a given course
SELECT x. type, COUNT(x.type) AS mod_count
FROM (SELECT resources.type
FROM mdl_resource resources
WHERE resources.course = 560
UNION ALL
SELECT modules.name AS type
FROM mdl_course_modules c_modules
INNER JOIN mdl_modules modules ON c_modules.module = modules.id
WHERE modules.name != 'resource'
AND c_modules.course = 560
) x
GROUP BY x.type

8. Lists all the resources and modules available and makes a count for a given course for those resources or modules that course contains.
SELECT score. type, IFNULL(y.mod_count,0) mod_count
FROM (SELECT x.type, COUNT(x.type) AS mod_count
FROM (SELECT resources.type
FROM mdl_resource resources
GROUP BY TYPE
UNION ALL
SELECT modules.name AS type
FROM mdl_modules modules
WHERE modules.name != 'resource'
GROUP BY TYPE
) x
GROUP BY x.type
) AS score
LEFT JOIN (
SELECT x.type, COUNT(x.type) AS mod_count
FROM (  
SELECT resources.type  
FROM mdl_resource resources  
WHERE resources.course = 560  
UNION ALL  
SELECT modules.name AS type  
FROM mdl_course_modules c_modules  
INNER JOIN mdl_modules modules ON c_modules.module = modules.id  
WHERE modules.name != 'resource'  
AND c_modules.course = 560  
) x  
GROUP BY x.type  
)y ON score.type = y.type

9. Number of students who have completed and passed each module at a particular time
SELECT c.name AS 'COURSE NAME', Count(c.name) AS 'NO OF COURSE COMPLETERS '  
FROM user u, certificate_issues ci, certificate c  
WHERE u.id = ci.userid AND  
ci.certificateid = c.id  
GROUP BY c.name

10. List all the rows in your role assignments table that no longer match to a user:
SELECT *  
FROM 'mdl_role_assignments'  
WHERE 'userid' NOT  
IN (  
SELECT id  
FROM mdl_user  
)

4.4 Removal of irrelevant php functions from moodle

4.4.1 Possible removal of unused functions in shortanswer/questiontype.php
In file question/type/shortanswer/questiontype.php, the following 2 functions are in fact not used at all:
function check_response(& $question,& $state)
function compare_responses($question,$state, $teststate)
4.4. REMOVAL OF IRRELEVANT PHP FUNCTIONS FROM MOODLE

4.4.2 Unused functions from questionarre in locallib.php

questionnaire_response_key_cmp,
questionnaire_preview,
questionnaire_get_active_surveys_menu,
questionnaire_get_surveys_menu,
questionnaire_survey_has_questions,
questionnaire_survey_exists

In locallib.php, unused constants are
QUESTIONNAIRE_BGALT_COLOR1,
QUESTIONNAIRE_BGALT_COLOR2,
QUESTIONNAIRE_EDITING,
QUESTIONNAIRE_ACTIVE1,
QUESTIONNAIRE_ENDED,
QUESTIONNAIRE_ARCHIVED,
QUESTIONNAIRE_TESTING,
QUESTIONNAIRE_ACTIVE2

4.4.3 Remove the grade item for an activity if the grade for that activity is always zero

When these modules create an instance of the activity, they all call a function called something like "modulename_grade_item_update()". This function sets up some parameters and then calls the standard "grade_update()" function.

If the activity always has a grade of zero, e.g. $glossary>scale==0 or $quiz>grade==0 or $hotpot>grade==0, then these modules all use the following parameter setting:

1. $params['gradetype'] = GRADE_TYPE_NONE;

As a result, no grade item is created for the activity. Fair enough, so far. Later on the activity may be edited and a non-zero grade becomes possible. The modules then use the following parameter setting:

params['gradetype'] = GRADE_TYPE_VALUE,

At this point grade item activity gets created. Great, That's what we would want. But what happens if the.

It is indeed the case that this function does not initially create grade items for which gradetype = GRADE_TYPE_NONE. That's OK. It will create grade items if gradetype later changes to GRADE_TYPE_VALUE.

4.4.4 Deprecated functions still allowed

Isteacher(); Get_course_teachers() Isstudent() These functions are still allowed and should be removed.
Chapter 5

Moodle Optimization Techniques

5.1 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.

5.2 Hardware Optimization

5.2.1 Load Testing on SSD vs HDD

5.2.1.1 Experiment Setup

Testing was done on two servers with 1GB RAM on Intel Core i5-2310 CPU @ 2.90GHz 4 processor, one with HDD(HardDisk) and one with SSD(Solid State Disk). The Experiment was performed for a chat activity. The sequence of pages visited were as follows:

Login -> View Course -> View Chat Page -> Initialize Chat -> Initialize Initial Update
The following pages were visited 5 times

Post Chat -> Initialize Chat Update For 1 user and 1 iteration, a total of 15 samples were tested using Apache Jmeter
5.2.1.2 Results

<table>
<thead>
<tr>
<th>Number of concurrent users</th>
<th>Response Time on HDD(s)</th>
<th>Response Time on SSD (s)</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3.671</td>
<td>0.349</td>
<td>10.51</td>
</tr>
<tr>
<td>20</td>
<td>8.874</td>
<td>1.048</td>
<td>8.467</td>
</tr>
<tr>
<td>30</td>
<td>15.303</td>
<td>1.938</td>
<td>7.89</td>
</tr>
<tr>
<td>40</td>
<td>129.786</td>
<td>3.438</td>
<td>37.75</td>
</tr>
<tr>
<td>50</td>
<td>364.480</td>
<td>5.274</td>
<td>69.11</td>
</tr>
<tr>
<td>60</td>
<td>Database Overload</td>
<td>5.97</td>
<td>-</td>
</tr>
<tr>
<td>70</td>
<td>Database Overload</td>
<td>6.492</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>Database Overload</td>
<td>8.009</td>
<td>-</td>
</tr>
<tr>
<td>90</td>
<td>Database Overload</td>
<td>8.085</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>Database Overload</td>
<td>9.797</td>
<td>-</td>
</tr>
<tr>
<td>110</td>
<td>Database Overload</td>
<td>13.759</td>
<td>-</td>
</tr>
<tr>
<td>120</td>
<td>Database Overload</td>
<td>16.828</td>
<td>-</td>
</tr>
<tr>
<td>130</td>
<td>Database Overload</td>
<td>22.991</td>
<td>-</td>
</tr>
<tr>
<td>140</td>
<td>Database Overload</td>
<td>30.187</td>
<td>-</td>
</tr>
<tr>
<td>150</td>
<td>Database Overload</td>
<td>36.119</td>
<td>-</td>
</tr>
<tr>
<td>151</td>
<td>Database Overload</td>
<td>39.141</td>
<td>-</td>
</tr>
<tr>
<td>152</td>
<td>Database Overload</td>
<td>Database Overload</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.1: Average User Response Time on HDD vs SSD (in s)
5.2. HARDWARE OPTIMIZATION

5.2.1.3 Graph

The graph is shown in Fig. 5.1.

![Graph showing average user response time for SSD vs HDD](image)

Figure 5.1: Average User Response Time (for SSD vs HDD)

5.2.1.4 Conclusion

The number of users supported by Moodle installed on SSD for 1 GB RAM is 151 as compared to 50 for Moodle on HDD for 1 GB RAM.
5.3 Front-End Optimization Techniques

5.3.1 Reduce Number of HTTP Requests by using Image Maps

Internet finds servers through IP addresses. URLs typically contain hostnames instead of IP address, but the IP address is necessary for the browser to make its request. Domain Name Servers map hostnames to IP addresses. However, it takes 20120 milliseconds for the browser to translate the IP address for a specific hostname. The browser is idle until the DNS lookup is completed. [11]

5.3.1.1 Experiment Setup

Experiment was performed on a server with 4GB RAM on SSD and Intel Core i5-2310 CPU @ 2.90GHz 4 processor.

5.3.1.2 Results

<table>
<thead>
<tr>
<th>users</th>
<th>without image map</th>
<th>with image map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXP 1</td>
<td>EXP 2</td>
</tr>
<tr>
<td>100</td>
<td>68.5</td>
<td>51.2</td>
</tr>
<tr>
<td>200</td>
<td>390.3</td>
<td>370.9</td>
</tr>
<tr>
<td>300</td>
<td>1094.8</td>
<td>991.3</td>
</tr>
<tr>
<td>400</td>
<td>1614.2</td>
<td>1526.6</td>
</tr>
</tbody>
</table>

Table 5.2: Average User Response Time for 1 iteration

<table>
<thead>
<tr>
<th>Users</th>
<th>without image map</th>
<th>with image map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXP 1</td>
<td>EXP 2</td>
</tr>
<tr>
<td>100</td>
<td>73.4</td>
<td>82.7</td>
</tr>
<tr>
<td>200</td>
<td>406.7</td>
<td>407</td>
</tr>
<tr>
<td>300</td>
<td>1061.1</td>
<td>1146.1</td>
</tr>
<tr>
<td>400</td>
<td>1801.9</td>
<td>1785</td>
</tr>
</tbody>
</table>

Table 5.3: Average User Response Time for 10 iterations
5.3. FRONT-END OPTIMIZATION TECHNIQUES

Table 5.4: Average User Response Time for 100 iterations

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

5.3.1.3 Graphs

Figure 5.2: Average User Response Time for 1 iteration
Figure 5.3: Average User Response Time for 10 iterations

Figure 5.4: Average User Response Time for 100 iterations
5.3.2 Caching By Using Far Future Expires Header

To reduce the number of HTTP requests and decrease the size of HTTP responses, the browser uses a cache. This makes the web pages load faster. A server uses the Expires header to inform the 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). It was changed to 20th Aug 2015 20:00 GMT.

5.3.2.1 Experiment 1 Setup

Experiment 1 was performed on a system with 1GB RAM, 128GB SSD and Intel Core i5-2310 CPU @ 2.90GHz 4 Processor.
To sequence of pages followed starting from Login Screen:

Login -> Home -> View Course -> Logout

5.3.2.2 Experiment 1 Results

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Response Time with Caching(s)</th>
<th>Response Time without Caching(s)</th>
<th>Percent Reduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>84.075</td>
<td>136.372</td>
<td>38.34</td>
</tr>
<tr>
<td>20</td>
<td>168.97</td>
<td>302.044</td>
<td>44.06</td>
</tr>
<tr>
<td>30</td>
<td>233.708</td>
<td>470.641</td>
<td>50.34</td>
</tr>
<tr>
<td>40</td>
<td>344.575</td>
<td>641.054</td>
<td>46.25</td>
</tr>
<tr>
<td>50</td>
<td>454.205</td>
<td>892.575</td>
<td>49.11</td>
</tr>
<tr>
<td>60</td>
<td>454.205</td>
<td>892.575</td>
<td>43.95</td>
</tr>
<tr>
<td>70</td>
<td>670.668</td>
<td>1188.28</td>
<td>43.56</td>
</tr>
</tbody>
</table>

Table 5.5: Average User Response Time for 1 user different iterations
5.3.2.3 Experiment 1 Graphs

Figure 5.5: Average User Response Time for 1 user different iterations

5.3.2.4 Experiment 2 Setup

Experiment was performed on a system with 1GB RAM, 128GB SSD and Intel Core i5-2310 CPU @ 2.90GHz 4 Processor.
The Experiment was performed for a chat activity. The sequence of pages visited were as follows:

Login -> View Course -> View Chat Page -> Initialize Chat -> Initialize Initial Update
The following pages were visited 5 times

Post Chat -> Initialize Chat Update
For 1 user and 1 iteration, a total of 15 samples were tested using Apache Jmeter.
5.3. FRONT-END OPTIMIZATION TECHNIQUES

5.3.2.5 Experiment 2 Results

<table>
<thead>
<tr>
<th>No. of Concurrent Users</th>
<th>Avg user response Time Without Expires Header (in s) (no caching)</th>
<th>Avg user response Time With Expires Header (in s) (caching)</th>
<th>Reduction in time(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.625</td>
<td>0.144</td>
<td>76.96</td>
</tr>
<tr>
<td>20</td>
<td>1.839</td>
<td>0.408</td>
<td>77.81</td>
</tr>
<tr>
<td>40</td>
<td>5.061</td>
<td>1.21</td>
<td>76.09</td>
</tr>
<tr>
<td>60</td>
<td>7.086</td>
<td>1.778</td>
<td>74.91</td>
</tr>
<tr>
<td>80</td>
<td>8.124</td>
<td>2.426</td>
<td>70.14</td>
</tr>
<tr>
<td>100</td>
<td>9.882</td>
<td>3.071</td>
<td>68.92</td>
</tr>
</tbody>
</table>

Table 5.6: Average User Response Time for 10 iterations with different number of users

5.3.2.6 Experiment 2 Graphs

![Average Response Time (s)](image)

Figure 5.6: Average User Response Time for 10 iterations with different number of users

5.3.2.7 Conclusion

There is a reduction of around 70-80 % in average user response time after implementing Far Future Expires Header Optimization Technique.
5.3.3 Reducing DNS Lookups

Internet finds servers through IP addresses. URLs typically contain hostnames instead of IP address, but the IP address is necessary for the browser to make its request. Domain Name Servers map hostnames to IP addresses. However, it takes 20120 milliseconds for the browser to translate the IP address for a specific hostname. The browser is idle until the DNS lookup is completed. [11]

5.3.3.1 Experiment 1

Experiment was performed to observe and calculate the difference in average user response time for a client on a network with average download speed of 2 Mbps.

5.3.3.2 Experiment 1 Results

<table>
<thead>
<tr>
<th>Continent</th>
<th>Country</th>
<th>University</th>
<th>Without DNS CACHE</th>
<th>WITH DNS CACHE</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia</td>
<td>India</td>
<td>IITB</td>
<td>2.357</td>
<td>1.426</td>
<td>0.931</td>
</tr>
<tr>
<td>Asia</td>
<td>India</td>
<td>IITM</td>
<td>2.516</td>
<td>1.612</td>
<td>0.904</td>
</tr>
<tr>
<td>Asia</td>
<td>Singapore</td>
<td>Singapore Institute of Management</td>
<td>1.381</td>
<td>1.055</td>
<td>0.326</td>
</tr>
<tr>
<td>Asia</td>
<td>Japan</td>
<td>Sojo University</td>
<td>6.223</td>
<td>3.116</td>
<td>3.107</td>
</tr>
<tr>
<td>Asia &amp; Europe</td>
<td>Russia</td>
<td>Tomask University</td>
<td>1.989</td>
<td>1.562</td>
<td>0.427</td>
</tr>
<tr>
<td>Europe</td>
<td>Spain</td>
<td>Graduate School of Management, Barcelona</td>
<td>3.138</td>
<td>1.813</td>
<td>1.325</td>
</tr>
<tr>
<td>Europe</td>
<td>UK</td>
<td>University of Nottingham</td>
<td>4.174</td>
<td>2.041</td>
<td>2.133</td>
</tr>
<tr>
<td>North America</td>
<td>US</td>
<td>UCLA</td>
<td>4.6</td>
<td>3.657</td>
<td>0.943</td>
</tr>
<tr>
<td>North America</td>
<td>US</td>
<td>University of Florida</td>
<td>19.88</td>
<td>9.3</td>
<td>10.58</td>
</tr>
<tr>
<td>South America</td>
<td>Argentina</td>
<td>LirWeb(University of Católica Argentina)</td>
<td>2.534</td>
<td>1.71</td>
<td>0.824</td>
</tr>
<tr>
<td>South America</td>
<td>Colombia</td>
<td>University of Grand Colombia</td>
<td>2.341</td>
<td>1.438</td>
<td>0.903</td>
</tr>
<tr>
<td>Africa</td>
<td>Egypt</td>
<td>Orirflame University</td>
<td>5.497</td>
<td>4.288</td>
<td>1.209</td>
</tr>
<tr>
<td>Africa</td>
<td>South Africa</td>
<td>Virtual Academy of South Africa</td>
<td>4.936</td>
<td>2.588</td>
<td>2.348</td>
</tr>
<tr>
<td>Australia</td>
<td>Australia</td>
<td>Australian National University(AUN)</td>
<td>4.525</td>
<td>3.559</td>
<td>0.966</td>
</tr>
<tr>
<td>Australia</td>
<td>Australia</td>
<td>Monash University</td>
<td>4.947</td>
<td>4.241</td>
<td>0.706</td>
</tr>
</tbody>
</table>

Table 5.7: Average User Response Time with and without DNS Cache for 1 user

5.3.3.3 Experiment 2

Experiment 1 was carried out for 100 iterations and three scenarios:

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

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

Scenario 3:
DNS Cache Entries = 512
DNS Cache Expiration Period = 3600 seconds
HTTP Keep Alive Timeout = 0

5.3.3.4 Experiment 2 Results

<table>
<thead>
<tr>
<th>Continent</th>
<th>University</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Diff between S1 &amp; S2</th>
<th>Scenario 3</th>
<th>Diff between S1 &amp; S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>North America</td>
<td>UCLA</td>
<td>173.984</td>
<td>169.284</td>
<td>4.7</td>
<td>178.69</td>
<td>9.406</td>
</tr>
<tr>
<td>Asia</td>
<td>IITM</td>
<td>108.93</td>
<td>105.677</td>
<td>3.253</td>
<td>110.548</td>
<td>4.871</td>
</tr>
<tr>
<td>Australia</td>
<td>AUN</td>
<td>347.361</td>
<td>344.961</td>
<td>2.4</td>
<td>354.336</td>
<td>9.375</td>
</tr>
<tr>
<td>Africa</td>
<td>Virtual Academy of South Africa</td>
<td>244.035</td>
<td>240.246</td>
<td>3.789</td>
<td>256.08</td>
<td>15.834</td>
</tr>
<tr>
<td>Europe</td>
<td>University of Nottingham</td>
<td>153.71</td>
<td>150.213</td>
<td>3.497</td>
<td>156.76</td>
<td>6.547</td>
</tr>
<tr>
<td>South America</td>
<td>University of Grand Colombia</td>
<td>142.241</td>
<td>135.908</td>
<td>6.333</td>
<td>146.763</td>
<td>10.855</td>
</tr>
</tbody>
</table>

Table 5.8: Average User Response Time for 1 user for the scenarios mentioned in Section 5.3.3.3

5.3.3.5 Conclusion

Scenario 3 provides the best results. Hence, increasing the number of DNS cache entries and DNS expiration period for a network which supports HTTP keep-alive increased the average user response time.

5.3.4 Gzip Components

We use the same compression technique which is used to reduce file sizes in email messages and on FTP sites. Starting with HTTP/1.1, web clients indicate compression with the Accept-Encoding header in the HTTP request.

Accept-Encoding: gzip, deflate

When the web server sees this header in the request, it compresses the response using the methods listed by the client. The web server informs the web client of this using the Content-Encoding header in the response.

Content-Encoding: gzip
5.3.4.1 Experiment

The experiment was performed for 10 Moodle pages.

5.3.4.2 Results

<table>
<thead>
<tr>
<th>No. of Files</th>
<th>Compressed (KB)</th>
<th>Not Compressed (KB)</th>
<th>Response Time with Compression (in s)</th>
<th>Response Time without Compression (in s)</th>
<th>Difference (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>42</td>
<td>215</td>
<td>926</td>
<td>711</td>
<td>2.586</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.477</td>
</tr>
<tr>
<td>Login</td>
<td>13</td>
<td>138</td>
<td>597</td>
<td>459</td>
<td>0.611</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.819</td>
</tr>
<tr>
<td>View Course</td>
<td>42</td>
<td>187</td>
<td>804</td>
<td>617</td>
<td>2.247</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.26</td>
</tr>
<tr>
<td>View Forum(1 post)</td>
<td>41</td>
<td>187</td>
<td>802</td>
<td>615</td>
<td>2.319</td>
</tr>
<tr>
<td>View Notes</td>
<td>35</td>
<td>182</td>
<td>768</td>
<td>586</td>
<td>1.959</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.584</td>
</tr>
<tr>
<td>View Blog</td>
<td>35</td>
<td>218</td>
<td>889</td>
<td>671</td>
<td>2.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.113</td>
</tr>
<tr>
<td>View Calendar</td>
<td>42</td>
<td>207</td>
<td>861</td>
<td>654</td>
<td>2.267</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.932</td>
</tr>
<tr>
<td>View Participants</td>
<td>40</td>
<td>188</td>
<td>806</td>
<td>618</td>
<td>2.21</td>
</tr>
<tr>
<td>View Tags</td>
<td>27</td>
<td>182</td>
<td>781</td>
<td>599</td>
<td>1.644</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.713</td>
</tr>
<tr>
<td>1 page of quiz with 5 questions</td>
<td>49</td>
<td>198</td>
<td>847</td>
<td>649</td>
<td>2.679</td>
</tr>
<tr>
<td>View Assignments</td>
<td>43</td>
<td>187</td>
<td>804</td>
<td>617</td>
<td>2.311</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.076</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.394</td>
</tr>
</tbody>
</table>

Table 5.9: Average User Response Time for 1 user with and without compression of components

5.3.5 Deactivating ETags

Another way to determine whether the component which is stored in the browsers cache is similar to the one on the origin is ETags which were introduced in HTTP/1.1. An ETag is a string that uniquely identifies a specific version of a component. The origin server has the components ETag in the ETag response header.

However, ETags are constructed using attributes that make them unique to a specific server hosting a site. ETags won’t match when a browser gets the original component from one server and later makes a conditional GET request that goes to a different server.
5.3.5.1 Experiment

The experiment was performed when response contained ETag (default) header and when it did not (header ETag was deactivated).

The ETag Header was deactivated using the following directive for Apache2 Server.

Header unset ETag

5.3.5.2 Results

Response Header Size with ETag Header = 500 bytes Response Header Size without ETag Header = 412 bytes Savings = 48 bytes

5.3.5.3 Conclusion

48 bytes are saved after deactivating ETags

5.4 Conclusion

Of the 6 techniques mentioned in this chapter, the Hardware Optimization Technique (Section 5.2.1) and the Far Future Expires Header Optimization Technique (Section 5.3.2) are the most efficient.
Chapter 6

Quiz Module in Moodle

6.1 Quiz Database

There are 11 quiz tables in the database of moodle. In addition to these tables this module references other tables like questions, grades etc. The detailed description of these tables can be seen in Fig. 6.1.

Figure 6.1: Moodle Quiz Database
6.2 Question Types in Moodle

Moodle supports many types of questions. It supports 12 types of questions by default. [3]

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.
6.3 SQL Query Log for Quiz

As moodle uses its own abstraction layer to convert the PHP queries to SQL queries 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.

6.3.1 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, 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 \( \backslash \text{flush}\) /var/log/mysql/mysql.log.

To have a clear idea of the sql queries generated it will be better to separate the queries for each action. So obtain separate log for each action. After obtaining the log for attempting the quiz it can be formatted and tested on SSD to know the performance issues of the quiz. This is explained in the next chapter.
Chapter 7

SSD v/s HDD for the ’Moodle’ DATABASE

7.1 Introduction

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.

7.1.1 Read and Write

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.

7.1.2 HDD

HDDs have a spiral recording surface, HDD surfaces are more efficient with larger files that are placed in continuous blocks. That way, the drive head can start and end its read in one continuous motion. When hard drives start to fill up, large files can become scattered around the disk platter, which is otherwise known as fragmentation[12]
7.1.3 PCIe SSDs

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

### 7.1.4 Latency

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.

### 7.1.5 Price

As of 2012, SSDs are more expensive per gigabyte than hard drives but prices for SSDs have fallen substantially in recent months. On the other hand hard drives
cost around 0.10\textpermgigabyte for 3.5", or 0.20 for 2.5", a typical flash SSD is about 0.80\textpergigabyte. The rate has come down by about 2 per GB in early 2012.\footnote{14}

7.1.6 Speed

An HDD disk uses rotating platters of magnetic drives and read/write heads for its operation. So the initial latency is more for HDDs than SSDs because a spin-up start for the disk is needed. Intel claims their SSD is about eight times faster than an HDD, thereby offering faster boot up times.\footnote{15}

7.1.7 Reliability

Unlike HDD drives, SSD disks do not have moving parts. So the SSDs are more reliable. Moving parts in an HDD increase the risk of mechanical failure. The fast motion of the platters and heads of the hard disk drive make it vulnerable to head crash. The reasons for head crashes are electronic failure, power failure, physical shock, wear and tear, corrosion, or poorly manufactured platters and heads. The next factor which affects reliability is the presence of magnets. HDDs use magnetic storage so are susceptible to damage or data corruption if the are brought close to powerful magnets. SSDs doesn't face such a risk.

7.2 Experiments Performed

7.2.1 Experiment Setup

Since it was just a preliminary experiment the execution times of the queries were obtained from the MySql server directly. After the execution of each query the time taken to either fetch/write the content will be displayed. This is the time taken by the Sql server to fire a query and receive a result. It is actually recommended to use a programming language to fire the queries and obtain the time, so as to minimise the effect of the cache. So when executing in the MySQL server the first execution time is only actually considered where the data is not present in the cache.

7.2.2 Experiment 1

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';
7.2. EXPERIMENTS PERFORMED

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

7.2.3 Experiment 2

7.2.3.1 Objective
To obtain the execution times of various constructed queries which use JOIN and UNION operations. To construct such queries a knowledge about the primary key-foreign key relation is required. This is obtained from the figure [9] given below.

7.2.3.2 Setup and Testing
1. This experiment was done on an HDD and SSD for analysing the execution times of the queries of JOIN and UNION.
2. These come under the category of complex queries (expensive), which get executed in the backend (MySQL) while moodle is used.
3. Here too SSD outperforms the HDD, like in the case read and write functions.
4. Instead of flushing the cache, totally different queries were tried, and also the systems were restarted at some instances to totally clear the cache.
5. MySQL caching was avoided by modifying the my.cnf file in the MySQL home directory.
6. In this experiment the main table is mdl_user, where the number of entries (rows) is 56,753, in which the primary key is id.
7. Here the join and union operations are performed by using primary key-foreign key relationship between tables.
8. The other two tables are mdl_user_lastaccess and mdl_user_preferences where the foreign keys are having the same name ie, userid.
9. The table mdl_user_preferences and mdl_user_lastaccess have 4,71,703 and 58,830 entries respectively.
10. The schema representing the relationships between the different tables is shown below.

7.2.4 Results (in seconds) of Experiment 2
There are 3 times given in the result against both HDD and SSD, in that the first time is the first access time. The other 2 times were obtained by executing the same operation consecutively. In the below commands ’a’ is an object of the table mdl_user and ’b’ is an object of the other table considered along with it.
7.2.4.1 JOIN

The JOIN keyword returns rows when there is a match in one of the tables.

**Syntax**

```sql
SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name;
```

**Query1**

```sql
select a.id, a.email, a.lastname, b.courseid, b.timeaccess from mdl_user a join mdl_user_lastaccess b on a.id=b.userid order by a.id;
```

HDD : 1.094/0.34/0.35
SSD : 0.55/0.32/0.33

**Query2**

```sql
select a.firstname, a.lastname, a.password, b.name, b.value from mdl_user a join mdl_user_preferences b on a.id=b.userid order by a.id;
```

HDD : 9.21/5.33/4.37
SSD : 5.05/3.76/3.74

7.2.4.2 LEFT JOIN

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

**Syntax**

```sql
SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name;
```

**Query3**

```sql
select a.firstaccess, a.lastaccess, a.confirmed, b.id, b.courseid from mdl_user a left join mdl_user_lastaccess b on a.id=b.userid order by b.id
```

HDD : 0.85/0.33/0.34
SSD : 0.33/0.20/0.19

7.2.4.3 RIGHT JOIN

The RIGHT JOIN keyword returns all the rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

**Syntax**

```sql
SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name;
```

**Query4**

```sql
select a.firstaccess, a.lastaccess, a.confirmed, b.id, b.courseid from mdl_user a right join mdl_user_lastaccess b on a.id=b.userid order by b.id
```

HDD : 0.68/0.25/0.25
SSD : 0.29/015/014
7.2. EXPERIMENTS PERFORMED

7.2.4.4 UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

NOTE: Each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

Syntax:
```
SELECT column_name(s) FROM table_name1 UNION SELECT column_name(s) FROM table_name2;
```

Query5
```
select id,username from mdl_user union select userid,name from mdl_user_preferences
```

HDD : 20.38/14.70/12.62  
SSD : 18.32/11.99/12.03

Query6
```
select timemodified,timecreated from mdl_user union all select id,courseid from mdl_user_lastaccessroot
```

HDD : 0.88/0.51/0.24  
SSD : 0.22/0.10/0.10

<table>
<thead>
<tr>
<th>Query</th>
<th>HDD(s)</th>
<th>SSD (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.094</td>
<td>0.55</td>
</tr>
<tr>
<td>2</td>
<td>9.21</td>
<td>5.05</td>
</tr>
<tr>
<td>3</td>
<td>0.85</td>
<td>0.33</td>
</tr>
<tr>
<td>4</td>
<td>0.68</td>
<td>0.29</td>
</tr>
<tr>
<td>5</td>
<td>20.38</td>
<td>18.32</td>
</tr>
<tr>
<td>6</td>
<td>0.88</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 7.1: Execution Time on HDD vs SSD (in s)

7.2.5 Experiment 3: SSD v/s HDD for Quiz module

7.2.5.1 Objective

1. Obtain the log file generated when a quiz is given in moodle.
2. Write a script to fire all the queries consecutively one after the other in systems loaded with SSD and HDD separately, to note the difference in the execution times.

7.2.5.2 Quiz

The quiz which was created under the course C Programming in moodle and comprised of 5 questions. To give this test a user named test1 was created. Around 200 queries are fired for logging in a user, for this experiment we are concerned only about the queries generated after the quiz is started. So just before giving the test, the MySQL
lof file was flushed. The quiz was given in random a manner with the submission including both right and wrong answers.

### 7.2.5.3 Moodle Used

This test was done on newly installed moodle with two users, it was initially done on a laptop with HDD then the database was dumped as .sql file and ported to the system using an SSD, the test was repeated by restarting the system so as to counter the effect of system cache. MySQL storage cache was disabled, by commenting the lines in the my.cnf file. Therefore the moodle database used was a very light one in terms of the data it contained.

### 7.2.5.4 Quiz- query log file

After the quiz was given, ie from the point of starting the quiz and submitting the answers, an SQL log file containing around 1,157 was generated. The data organisation was very poor that we there was no common pattern to read from the file using a script. So the next task was to relate the entire data to a common pattern so that it could be read and executed by using a script. The quiz comprised of questions from different categories so that the queries generated were a mixture of all types that would generated attempting it.

### 7.2.5.5 Cleaning Up log file

The log file contained many unnecessary data other than the queries, like the query number in the format '01 Query' to '63 Query' with each of these QUERY names repeated at least ten times. This was removed by using the bash script command sed. The command that was used was, sed command.[16]

```
    sed [OPTION]... scriptonlyifnootherscript [inputfile]...
```

Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and is consequently more efficient. But it is sed’s ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

```
    sed i s/old_name/new_name/g filename
```

The i is used to indicate the compiler to edit the file given in the command, ie to save the changes in the file itself. The s stands for substitute. The g stands for global, which means that all matching occurrences in the line would be replaced. The regular expression (i.e. pattern) to be searched is placed after the first delimiting symbol (slash here) and the replacement follows the second symbol.

### 7.2.5.6 Commands Used

For removing the query names the command used was,

```
    sed i /[09][09].Query//g filename
```

Here in the file any pattern of the following form,
7.2. EXPERIMENTS PERFORMED

Query is replaced with a blank space as given in the command as '//'.

The next problem encountered was there were a lot of blank lines in the log file and also the lines were not intact, many lines had white spaces on the left and right, this too was removed by using an sed command given below,

```
$s/ / ; s/ $/ ; /$/$/d' file.txt
```

- `s/ / ;` = left trim
- `s/ $/ ;` = right trim
- `/$/$/d` = remove empty line

This was not the last problem resolved, the next one being a single query existing in 3 or more lines especially those involving join operation, for this the file was edited and each query was made to reside in a single line so that reading by using a script is more easy. This was the general pattern followed throughout the file.

7.2.5.7 Reading The Queries Script

The script to read the queries from the file named 'queries' was written in python, when the code was used to read line by line there were white spaces in the lines obtained and this removed by using the python function 'strip()', line.strip() removes the white spaces at the ends of the line. The python libraries that were imported are time and MySQLdb. The time module was used to measure the time required to fire the queries and the MySQLdb was used to establish the database connection to the moodle database. The port number for MySQL was not given since python takes the default port of 3306 for establishing the connections.

7.2.5.8 Query Log - Final v/s Initial file

After the queries file was edited there were a total of 554 queries, i.e., these many queries were fired to the sql server for giving a quiz containing 5 questions. These includes different queries including select, update, join, union, etc. When the already executed queries considered for the second time, some create an issue of duplicate entry, i.e., the primary keys in some tables. This conflict was resolved by adding delete operations at appropriate places. Initially there were many queries that didn’t get executed when the code was run, this was checked by maintaining a counter for the queries. Two empty lists were initialized with the names worked and not worked both for storing the corresponding query numbers. Initially there were around 60 queries, which showed and error.

The main reason for this complication was, when a query file is generated by MySQL server, it stores the the values for the type BIGINT and DECIMAL with spaces in between the number, due to this the python interpreter return an error. Same numbers repeated at many places and this issue was resolved by searching and replacing the corresponding numbers.

This was more of a manual work, and it mainly occurred in the timestamps.
7.2.5.9 Code Snippet

```python
import time
import MySQLdb

db = MySQLdb.connect("localhost","root","silpa","moodle")
cursor = db.cursor()

query=[]
f=open('queries','r')
lines=f.readlines()

for line in lines:
    query.append(line.strip())
start=time.time()

for q in query:
    try:
        cursor.execute(q)
        db.commit()
    except:
        db.rollback()

end=time.time()
sec=end-start;

print "\nTIME : "+str(sec)+ " Seconds"
db.close()
```

Figure 7.3: Code Snippet for executing Queries from Query Log

7.2.5.10 Results

This time too it was found that the SSD outperformed HDD in its performance, the results have been tabulated below. The first time is assumed to be the real access time since the data is not present in the cache.
7.2. EXPERIMENTS PERFORMED

Table 7.2: Results (in sec): SSD v/s HDD for Quiz Module

<table>
<thead>
<tr>
<th>TEST NUMBER</th>
<th>SSD (sec)</th>
<th>HDD (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.78454</td>
<td>6.88408</td>
</tr>
<tr>
<td>2</td>
<td>0.76447</td>
<td>3.78522</td>
</tr>
<tr>
<td>3</td>
<td>0.73978</td>
<td>3.51644</td>
</tr>
<tr>
<td>4</td>
<td>0.74588</td>
<td>3.47997</td>
</tr>
</tbody>
</table>

This experiment was performed many times, by flushing the system cache (restarting the system) and the average time for HDD was around 4 seconds and that for SSD was around 0.75 seconds.

7.2.5.11 Latency and Specification

This was done on a moodle database for containing just 2 users and the quiz was given by only one person, and the latency involved in handling, for a single user is around 4 seconds, so when a real test is given with more than 100 participants then the latency would very high compared to system loaded with an SSD. To discard the effect of the RAM, the experiment was performed in systems having the same RAM (4 GB) and virtual memory (swap).

7.2.5.12 Effect of cache minimised

Since the queries file contain all the queries from the beginning of a quiz, it can be assumed that on each execution of the code some amount of data is fetched from the database, minimising the effect of cache-storage. This is because the operations involve a lot of update operations which requires writing into the database. The same quiz was given for another user almost the same queries were generated with a difference in the primary key values and also in the answers given. Answering the same questions
differently doesn’t affect the response time as the cost spent for similar queries (with different parameters) always remains the same.
7.2.5.13 Tests in two systems SSD vs HDD

![Code Output]

Figure 7.4: Tests in two systems SSD vs HDD - SSD
7.2. EXPERIMENTS PERFORMED

Figure 7.5: Tests in two systems SSD vs HDD - HDD
7.2.5.14 THE DATABASE TABLES (11) INVOLVED IN GIVING A QUIZ

1. quiz_grades
2. quiz_question_regrade
3. quiz_overrides
4. quiz
5. quiz_question_instances
6. quiz_question_statistics
7. quiz_question_response_stats
8. quiz_reportquiz_attempts
9. quiz_feedback
10. quiz_statistics

7.2.6 Experiment-4 SSD v/s HDD for viewing a forum

The SQL queries for viewing a FORUM were logged, and their execution times were noted (on various systems), so that a confirmation / a better statement could be made on installing an SSD for improving the performance.

7.2.6.1 Setup

1. As directed the experiment was carried out in 3 systems with 2 being i CORE-5 systems and the 3rd one being an i CORE-3 system.

2. This was to check the performance of MySQL server on varying hardware. The system loaded with SSD was an i CORE-5 system.

7.2.6.2 Question Raised

Q) Using an SSD and checking the execution times of various queries isn’t it that we are trying to optimize the database other than moodle.

A) The answer is an absolute NO, because the moodle architecture uses an abstract coded in pHp to generate MySQL queries, it’s sure that there is a latency involved in generating the queries. Even after generating the queries they are sent to the MySQL server execution, and we are analysing the performance of such executions. So the point that we are emphasising is that before bringing the major architectural changes in moodle we can take care of small and vital factors like these.
7.2. EXPERIMENTS PERFORMED

7.2.6.3 Steps

- In the front-end of the moodle a forum was viewed, and a comment was made, and the corresponding queries executed were obtained from LOG file.

- Being precise 284 queries were fired.

- These many queries were read from the file using a python script and were fired consecutively one after the other and the cumulative time was noted.

- The query file was totally dis-organised and messed up with unnecessary key words in between. This file was cleaned up using the sed command to remove blank lines and white spaces.

- The unnecessary keywords were replaced by blank, using the same sed command of the shell.

- The queries had white spaces when read from the file, those were cleared by using the strip() function provided by python.

7.2.6.4 Result

This time too it was found that SSD outperformed the HDD. This was checked for three different systems with different hardware and processing power. And the results which have been tabulated in the next page is in favour of using a server loaded with an SSD instead of a hard-disk. The system and memory configurations have also been given for reference.
### 7.2.6.5 Memory Configurations

<table>
<thead>
<tr>
<th>NO.</th>
<th>SYSTEM-1 HDD i5 (2.90 GHz)</th>
<th>SYSTEM-2 HDD i3 (2.10GHz)</th>
<th>SYSTEM-3 SSD i5 (3.20GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.87204</td>
<td>0.82761</td>
<td>0.22748</td>
</tr>
<tr>
<td>2</td>
<td>0.88721</td>
<td>0.81567</td>
<td>0.22757</td>
</tr>
<tr>
<td>3</td>
<td>0.87281</td>
<td>0.87527</td>
<td>0.21152</td>
</tr>
<tr>
<td>4</td>
<td>0.85449</td>
<td>0.86244</td>
<td>0.23625</td>
</tr>
</tbody>
</table>

Figure 7.6: Results (in sec): SSD v/s HDD Test on Different Systems

Figure 7.7: SSD i5 Memory Configuration

Figure 7.8: HDD i5 Memory Configuration
7.2. EXPERIMENTS PERFORMED

7.2.7 Experiment 5 - SSD v/s HDD - A TEST ON DIFFERENT HARDWARE

7.2.7.1 Objective

- Obtain the queries fired to the MySQL server while attempting a quiz in moodle.
- Clean up the query log file, which contains the queries, so that it can be read by using a script.
- Using a script read these queries from a file, and fire them consecutively one after the other using a looping structure so that the total time could be noted.
- Do this for systems with different hardware, so as to check and confirm the efficiency of an SSD, and also to find the dependence of hardware scaling on the performance of the MySQL database-server.

7.2.7.2 Systems Used

1. HP laptop (HDD) – CORETM i3 processor
2. DELL XPS laptop (HDD) – CORETM i5 processor
3. DELL XPS laptop (HDD) – CORETM i7 processor
4. Desktop System (HDD+SSD) – CORETM i5 processor

7.2.7.3 Quiz Details

- The quiz was a mixture of different questions.
- There were 5 questions in the quiz, the number was kept less so as to reduce the size of the query log.
- The moodle used had only one user named 'test'.
- There were only three courses in total with a total of 2 quizzes and one forum.

7.2.7.4 Steps

- The test user was made to login and attempt the quiz.
- The queries were logged only from the start of the quiz.
- The obtained query log file was cleaned up by running a customized shell file name clean.sh.
The clean.sh file contained various sed commands and two python scripts.

This execution of clean.sh ensured the removal of unnecessary white spaces and blank lines.

After the clean-up, the resultant file is formatted in such a way that each line contains a single query.

The next issue to be resolved is the Primary Key and Foreign Key conflicts that show up when executing the queries. This is resolved by inserting delete operations at appropriate places.

Apart from the above steps the file also requires manual editing, as in the case when a query starts with a paranthesis.

This makes it easy for a script to read from the file line by line into an array.

After this is done the queries in the array were fired by using a looping structure.

A timer was placed at the start and end if the loop, so that the exact time could be noted.

The strategy used to make sure that all the queries were fired was to maintain two arrays named worked and not_worked which contains the corresponding query numbers.

By looking up the array we can find the line number of the file were there was an issue with the SQL query.

```
sed -i -e 's/[0-9]*.Query//g' -e 's/[0-9]*.Quit/Quit /g' final
# to remove Query,Quit
sed -i 's/[0-9]*.Connect/Connect /g' final
# to remove Connect
sed -i "s/^([^[:blank:]]*[^0-9]*[^:][^0-9]*[^0-9]*[^0-9]*[^/]/g" final
# to remove timestamp
sed -i 's/^ */ ; s/ */ ; /^$/d' final
python testcodeon6.py
python testcode.py
```

Figure 7.10: clean_up.sh
7.2. EXPERIMENTS PERFORMED

Figure 7.11: testcode1on6.py

```python
import string
import re

r = open('final', 'r')
w = open('final1', 'w')
lines = r.readlines()

for line in lines:
    if re.match(r'^\s*$', line):
        pass
    else:
        w.write(line.strip())
        w.write('
')
```

Figure 7.11: testcode1on6.py
```python
import time
import MySQLdb

db = MySQLdb.connect("localhost", "root", "mmd", "modl")
cursor = db.cursor()

query=[]
#worked=[]
#not_worked=[]
f=open('quiz','r')
lines=f.readlines()
for line in lines:
    query.append(line.strip())

start=time.time()

##i=0
for q in query:
    try:
        i=i+1
        cursor.execute(q)
        worked.append(i);
        db.commit()
    except:
        not_worked.append(i)
        db.Rollback()

end=time.time()

print "The Queries that didn't work are: "+str(not_worked)
sec=end-start;

print "\nTIME : "+str(sec)+ " Seconds"
db.close()"
```

Figure 7.12: code.py
Figure 7.13: testcode.py

7.2.7.5 System 1 Configuration

Figure 7.14: System 1 Processor Configuration
CHAPTER 7. SSD V/S HDD FOR THE 'MOODLE' DATABASE

Figure 7.15: System 1 Memory Configuration

```
shabna@shabna-hp:~$ free -m
                     total  used   free  shared  buffers  cached
Mem:                  3846  2459   1386     0    109   1145
-/+ buffers/cache:     1264  2641
Swap:                 3987     0    3987
```

Figure 7.16: System 1 CPU Information

```
shabna@shabna-hp:~$ cat /proc/cpuinfo
processor             : 0
vendor_id             : GenuineIntel
cpu family            : 6
model                 : 58
model name            : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz
stepping              : 9
microcode             : 0x15
cpu MHz               : 1200.000
cache size            : 3072 KB
physical id           : 0
siblings              : 4
core id               : 0
cpu cores             : 2
apicid                : 0
initial apicid        : 0
fpu                   : yes
fpu_exception         : yes
cpuid level           : 13
wp                    : yes
flags                 : fpu vmx de pse ts glimpse mpx pae mce cx8 apic sep mtrr pg
                        mca cmov pat pse36 clflush dts acpl mmx fxsr sse sse2 ss ht tm pbe
                        syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl
                        xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 mmm
                        or ds cpl vmx est tm2 ssse3 cx16 xptr pdcm pclid sse4_1 sse4_2 x2apic
                        popcnt tsc_deadline_timer xsave avx f16c lahf_lm arat epb xsaveopt pln
                        pts dtrim tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
                        bogomips            : 4788.87
clfush size            : 64
cache_alignment       : 64
address sizes         : 36 bits physical, 48 bits virtual
```
Figure 7.17: System 1 Result Execution Time

```
shabna@shabna-hp:~$ python code.py
TIME : 4.13639998436 Seconds

shabna@shabna-hp:~$ python code.py
TIME : 4.2459628582 Seconds

shabna@shabna-hp:~$ python code.py
TIME : 4.61331105232 Seconds

shabna@shabna-hp:~$ python code.py
TIME : 4.03018283844 Seconds

shabna@shabna-hp:~$ python code.py
TIME : 4.8818590641 Seconds

shabna@shabna-hp:~$ python code.py
TIME : 4.2362780571 Seconds

shabna@shabna-hp:~$ python code.py
TIME : 4.07007217407 Seconds
```
7.2.7.6 System 2 Configuration

Figure 7.18: System 2 Processor Configuration

Figure 7.19: System 2 Memory Configuration
7.2. EXPERIMENTS PERFORMED

Figure 7.20: System 2 CPU Information
Figure 7.21: System 2 Result Execution Time
7.2. EXPERIMENTS PERFORMED

7.2.7.7 System 3 Configuration

Figure 7.22: System 3 Processor Configuration

Figure 7.23: System 3 Memory Configuration
**Figure 7.24: System 3 CPU Information**

```bash
mevin@DarkEye:~$ cat /proc/cpuinfo
processor: 0
vendor_id: GenuineIntel
cpu family: 6
model: 42
model name: Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz
stepping: 7
microcode: 0x23
cpu MHz: 880.000
cache size: 6144 KB
physical id: 0
siblings: 8
core id: 0
cpu cores: 4
apicid: 0
initial apicid: 0
fpu: yes
fpu_exception: yes
cpu_id_level: 13
wp: yes
flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts nppl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vmm致电flexpriority ept vpid
bogomips: 3991.02
cache_alignment: 64
cache_alignment: 64
address sizes: 36 bits physical, 48 bits virtual
```
7.2. EXPERIMENTS PERFORMED

Figure 7.25: System 3 Result Execution Time

```bash
mevin@DarkEye:~$ python code.py
TIME : 3.96209812164 Seconds
```
```
mevin@DarkEye:~$ python code.py
TIME : 3.133934021 Seconds
```
```
mevin@DarkEye:~$ python code.py
TIME : 3.23302006721 Seconds
```
```
mevin@DarkEye:~$ python code.py
TIME : 3.12024092674 Seconds
```
```
mevin@DarkEye:~$ python code.py
TIME : 3.32521986961 Seconds
```
```
mevin@DarkEye:~$ python code.py
TIME : 3.15912890434 Seconds
```
```
mevin@DarkEye:~$ python code.py
TIME : 3.86703014374 Seconds
```
7.2.7.8 System 4 Configuration - Best Among All four

Figure 7.26: System 4 Processor Configuration

Figure 7.27: System 4 Memory Configuration
### 7.2. EXPERIMENTS PERFORMED

**Figure 7.28: System 4 CPU Information**

<table>
<thead>
<tr>
<th>Processor</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor ID</td>
<td>GenuineIntel</td>
</tr>
<tr>
<td>CPU Family</td>
<td>6</td>
</tr>
<tr>
<td>Model</td>
<td>42</td>
</tr>
<tr>
<td>Model Name</td>
<td>Intel(R) Core(TM) i5-2310 CPU @ 2.90GHz</td>
</tr>
<tr>
<td>Stepping</td>
<td>7</td>
</tr>
<tr>
<td>Microcode</td>
<td>0x26</td>
</tr>
<tr>
<td>CPU MHz</td>
<td>1600.000</td>
</tr>
<tr>
<td>Cache Size</td>
<td>6144 KB</td>
</tr>
<tr>
<td>Physical ID</td>
<td>0</td>
</tr>
<tr>
<td>Siblings</td>
<td>4</td>
</tr>
<tr>
<td>Core ID</td>
<td>1</td>
</tr>
<tr>
<td>CPU Cores</td>
<td>4</td>
</tr>
<tr>
<td>APIC ID</td>
<td>2</td>
</tr>
<tr>
<td>Initial APIC ID</td>
<td>2</td>
</tr>
<tr>
<td>FDTV Bug</td>
<td>no</td>
</tr>
<tr>
<td>HLT Bug</td>
<td>no</td>
</tr>
<tr>
<td>FE0F Bug</td>
<td>no</td>
</tr>
<tr>
<td>COMA Bug</td>
<td>no</td>
</tr>
<tr>
<td>FPU</td>
<td>yes</td>
</tr>
<tr>
<td>FPU Exception</td>
<td>yes</td>
</tr>
<tr>
<td>CPU Id Level</td>
<td>13</td>
</tr>
<tr>
<td>WP</td>
<td>yes</td>
</tr>
<tr>
<td>Flags</td>
<td>fpu vme de pse tsx msr pae mce cx8 alic sep mtrr pge mca cmov pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant tsc arch_perfmon pebs bts xtopology nonstop tsc aperfmperf pni pclmulqdq dtes64 monitor ds cpl vmx est tm2 ssse3 cx16 xtr pdm pcd ssse4 1 ssse4 2 popcnt tsc_dte tsc_zerohit adline timer aes xsave avx lahf ln iida arat epb xsaveopt pni pts dtherm tpr_shad ow vmm hv npt fpxpriority opt vpid</td>
</tr>
<tr>
<td>Bogomips</td>
<td>5786.70</td>
</tr>
<tr>
<td>Clflush Size</td>
<td>64</td>
</tr>
<tr>
<td>Cache Alignment</td>
<td>64</td>
</tr>
<tr>
<td>Address Sizes</td>
<td>36 bits physical, 48 bits virtual</td>
</tr>
</tbody>
</table>
7.2.7.9 A Slight Change On The Current System

Steps

1. Open the my.cnf file of MySQL by using the command
   `sudo gedit /etc/mysql/my.cnf`
2. Change the the datadir field from `datadir=/var/lib/mysql` to `datadir=/solidstate/mysql`
   ie, storage from HDD to SSD
7.2. EXPERIMENTS PERFORMED

Figure 7.30: Initial Configuration

```
[mysqld]
#
# * Basic Settings
#
user = mysql
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/run/mysqld/mysqld.sock
port = 3306
basedir = /usr
data dir = /var/lib/mysql
tmpdir = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address = 127.0.0.1
```

Figure 7.31: Final Configuration

```
[mysqld]
#
# * Basic Settings
#
user = mysql
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/run/mysqld/mysqld.sock
port = 3306
basedir = /usr
data dir = /solidstate/mysql
tmpdir = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address = 127.0.0.1
```
CHAPTER 7. SSD V/S HDD FOR THE ‘MOODLE’ DATABASE

Figure 7.32: Results Using the Same System

The test was carried out for different systems with different hardware and also on the HDD and SSD of the same system as mentioned above. The results obtained prove that the SSD outperforms the HHD on any system with any hardware specification. Therefore it’s recommended to use a server loaded with SSD for moodle database.
7.2. EXPERIMENTS PERFORMED

### 7.2.8 Experiment 6

#### 7.2.8.1 Objective

The same test on quiz module (5 questions) was performed, but this time it was iterated 100 times so as to virtually perform an experiment as if 100 people are giving a quiz simultaneously. Then to calculate the factor of time taken when the execution is done on an SSD and an HDD. Also to check whether the factor is almost the same with 1 user (from previous experiment) and 100 users.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>CORE™ i3 (HDD) (3M 2.40Ghz)</th>
<th>CORE™ i5 (HDD) (3M 2.90Ghz)</th>
<th>CORE™ i7 (HDD) (6M 2.90Ghz)</th>
<th>CORE™ i5 (HDD) (6M 3.20Ghz)</th>
<th>CORE™ i5 (SSD) (6M 3.20Ghz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.13</td>
<td>3.09</td>
<td>3.96</td>
<td>3.17</td>
<td>0.75</td>
</tr>
<tr>
<td>2</td>
<td>4.24</td>
<td>3.69</td>
<td>3.13</td>
<td>2.60</td>
<td>0.77</td>
</tr>
<tr>
<td>3</td>
<td>4.61</td>
<td>3.49</td>
<td>3.23</td>
<td>2.62</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>4.03</td>
<td>3.13</td>
<td>3.12</td>
<td>2.62</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Table 7.3: Results(in sec): SSD v/s HDD Hardware Scaling
### Figure 7.33: Modified Code for 100 Users

```python
code.py
import time
import MySQLdb

db = MySQLdb.connect("localhost","root","root","mdl")
cursor = db.cursor()

query=[]
total_time=0
f=open('quiz','r')
lines=f.readlines()
for line in lines:
    query.append(line.strip())

for i in xrange(100): #looping 100 times
    start=time.time()
    for q in query:
        try:
            cursor.execute(q)
            db.commit()
        except:
            db.rollback()
    end=time.time()
    sec=end-start;
    total_time+=sec

print "\nThe total time for 100 users is : "+str(total_time)+ " Seconds"
db.close()
```

7.2.8.2 Results

**Figure 7.34: HDD**

```
aakash@aakash-desktop:~$ python code.py
The total time for 100 users is : 287.187157889 Seconds
```

**Figure 7.35: SSD**

```
aakash@aakash-desktop:~$ cd /solidstate/
aakash@aakash-desktop:/solidstate$ python code.py
The total time for 100 users is : 77.8809432983 Seconds
```
7.3. CONCLUSION

Table 7.4: Results (in sec): SSD v/s HDD Test for 100 Users giving a quiz simultaneously

<table>
<thead>
<tr>
<th></th>
<th>CORE™i5 (HDD) (6M 3.20Ghz)</th>
<th>CORE™i5 (SSD) (6M 3.20Ghz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>287.187</td>
<td>77.881</td>
</tr>
</tbody>
</table>

7.2.8.3 Factor

1) For 100 Users
   \[\frac{287.187}{77.881} = 3.688\]

2) For a single user,
   The times 2.752 seconds and 0.758 seconds were obtained from Experiment-5 results.
   For this experiment only System-4 was considered. Therefore in the case of a single user the average of the execution times was taken (average of 4 times).
   \[\frac{2.752}{0.758} = 3.631\]

It was proved earlier that SSD shows better performance than an HDD. This experiment was done mainly to calculate the factor (ratio) of the 2 execution times, so as to compare the case of single user with that of 100 users giving a quiz simultaneously. The ratio obtained in the two cases were almost the same ie, around 3.5.

7.3 Conclusion

From the above results it can be confirmed that the execution times of moodle-operations especially those which updates/stores values in the database can be reduced drastically if we load both the moodle and its associated database in server using an SSD. For best performance we could use two dedicated servers, one for WEB (loading the moodle site) and the other for database so as speed up the moodle site.

In the current scenario servers loaded with SSD are available in plenty in the market, but their prices are bit more as compared to the normal conventional server. The SSD-server of the Dell company mainly comes in the PowerEdge series of servers.

We will be able to bridge the gap between server and storage performance with Dell PowerEdge Express Flash PCIe solid-state storage device (SSD). It has no mechanical limitations of a hard disk drive, Dells PowerEdge Express Flash PCIe-SSD can help break away I/O (input/output) bottlenecks for latency-sensitive applications. The problems faced by moodle can be solved to a certain extend by avoiding this latency.[18]

7.3.1 High Performance

The Dell PowerEdge Express Flash PCIe-SSD gives outstanding IOPS performance. Dell Express Flash technology can deliver up to 1000x improvement in IOPS over
traditional HDD. It is designed to deliver sequential throughput on reads and writes of up to 1.8/1.2 Gbps. In addition, the Dell PowerEdge Express Flash PCIe-SSD does not burden the servers CPU and memory with NAND flash management overhead. Flash management is provided by the device controller running firmware on a high-performance ASIC that resides within the PCIe-SSD. This architecture offers superb IOPS and throughput performance without "borrowing" server resources. [18]

Dell Express Flash technology is currently available for purchase on performance 2-socket and 4-socket PowerEdge server platforms, including the R820, R720, R620, T620, M620 and M820.
Chapter 8

Testing Moodle with Jmeter

8.1 Introduction

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

8.2 Testing by integrating different severs with Moodle

8.2.1 Common Experimental Setup

8.2.1.1 Installation

1. Install LAMP Server
2. Download latest version of moodle.
3. Extract into /var/www
4. Create /var/moodledata and assign appropriate permissions
5. Disable other applications that make use of ports 80 and 443
6. Create required databases.
7. Begin Moodle Installation from localhost/moodle
8. Provide details to finish installation.

8.2.1.2 Testing

1. Add loadtesting plugin to moodle : https://github.com/kabalin/moodle-jmeter-script-generator
CHAPTER 8. TESTING MOODLE WITH JMETER

2. Select testing criterion.
4. Load Script onto Jmeter
5. To test different timers, add required timer from menu and provide necessary parameters.
6. To test for different number of users, change number of threads accordingly.
7. To alter the time period within which ‘n’ concurrent users go live, alter the Ramp-Up Time.
8. Run test.

8.2.2 To integrate with another server

8.2.2.1 Setting up Moodle on Lighttpd

1. Set up LLMP stack.
2. Extract Moodle packages to /var/www.
3. Create database directory moodledata.
5. Set up database.
6. Install Moodle.

8.2.2.2 Testing

Same as mentioned in Section 8.2.1.2

8.3 Experiments

8.3.1 Experiment 1

8.3.1.1 Objective

To establish the average response time when different timers are used to generate the time interval between two requests.

8.3.1.2 Procedure

1. The standard script is downloaded using the Jmeter plugin for Moodle.
2. The number of users is set to 20 to ensure speedy results and minimal variation in server response between different requests.
3. The tests are looped 10 times to ensure standardized results.
4. The appropriate timer is added and tests are run for each timer.
8.3. EXPERIMENTS

8.3.1.3 TEST CONDITIONS

Default package installation of Moodle on Windows was used on a system with 1 GB RAM and Core2Duo processor.

8.3.1.4 Results

![Average Response Time (in ms) vs. Type of Timer](image)

Figure 8.1: Average Response Time (in ms) vs. Type of Timer

8.3.1.5 Conclusion

1. Constant Throughput Timer seems to yield lower response times compared to other timers.

2. Synchronized Timer seems to give the highest response time and hence must be used for worst case analysis.

8.3.2 Experiment 2

8.3.2.1 OBJECTIVE

To establish the average response time when different timers are used to generate the time interval between two requests

8.3.2.2 PROCEDURE

1. The standard script is downloaded using the JMeter plugin for Moodle.

2. The number of users is gradually increased until errors are produced by server.
8.3.2.3 TEST CONDITIONS

1. Server Details: 2 Intel Xeon 6 core processor, 96 GB RAM, 1 TB HDD

2. Default Moodle setup with LAMP.

3. Ramp-Up Period : 1 s (To simulate real time traffic)

4. No. of loops : Forever

5. This simulates a consistent ’n’ number of threads handled by the server.

6. We test until we obtain a java.net.connectionexception due to connection timeout.

7. This indicates that the server is overloaded with its maximum capacity of requests and is unable to handle more.

8. The number of threads is increased from 100 in steps of 100.

9. Once errors are encountered, system is tested for smaller number of threads until the threshold is encountered.

10. The tests are constructed for a built in QUIZ plug-in on moodle with 1 question which involves

11. Login, View Course, View Quiz, Start Attempt, Submit Quiz, Finish Attempt and Logout.

8.3.2.4 Results

![Summary Report]

Figure 8.2: Average Response Time (in ms) for 412 users
8.3. EXPERIMENTS

Figure 8.3: Average Response Time (in ms) for 413 users

Figure 8.4: Average Response Time (in ms) vs Number of Users

8.3.2.5 Conclusion

1. There seems to be a linear relationship between number of users and average response time.

2. As per server configurations, the server is capable of handling up to 412 threads for quiz functionality of Moodle without any errors.
8.3.3 Experiment 3

8.3.3.1 Objective

To compare Moodle’s response when different web servers are integrated with it (Lighttpd vs Apache).

8.3.3.2 Test Conditions

1. Ramp-up time: 1s
2. Number of Users: Tested from 25 users to 125 Users in steps of 25.
3. Web Servers: Apache vs Lighttpd
4. The tests are constructed for a built in QUIZ plug-in on moodle with 1 question which involves
5. Login, View Course, View Quiz, Start Attempt, Submit Quiz, Finish Attempt and Logout.
6. Tests are conducted on a system with 6GB RAM, 32 GB SSD, i5 processor.

8.3.3.3 Results

![Average Response Time Graph]

Figure 8.5: Average Response Time (in ms) for Lighttpd vs Apache

8.3.3.4 Conclusion

Average Response Time seems to be slightly smaller for Lighttpd than Apache when tested using JMeter.
8.3. EXPERIMENTS

8.3.4 Experiment 4

8.3.4.1 Objective

To compare Moodle’s response when different web servers are integrated with it (Lighttpd vs Apache) using ApacheBench.

8.3.4.2 Procedure

1. Install Apache as well as Lighttpd on the system.
2. Start Apache, test for the required number of users.
3. Stop Apache Server, start Lighttpd & test for the same criterion.
4. Run ab -n total number of requests -c concurrency address to test.

8.3.4.3 Test Conditions

1. Tests are conducted on a system with 6GB RAM, 32 GB SSD, i5 processor.
2. Two set of tests were conducted.
3. 100 concurrent users.
4. 300 concurrent users.
5. Max-connections was set to 300 for both.
6. Timeout was set to 15s.

8.3.4.4 Results

![100.png](image)

Figure 8.6: Average Response Time (in ms) for Lighttpd vs Apache
8.3.4.5 Conclusion

Lighttpd seems to have a much better response time when loaded with concurrent requests and hence handles load better.

8.4 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.

8.4.1 Experimental Setup

All the tests are done in Intel Core i3-3110M CPU 4 GB RAM, 2.4 Ghz, OS: Linux. Web and database server are two different hosts running on the same system. The various steps in doing the experiment are [19]:

1. Generate Jmeter Script
2. Extract the jmeter script files into [moodle_dir]/admin/report/loadtesting
3. Login to moodle.
4. Select Settings > Site Administration > Reports > JMeter loadtesting
5. Start jmeter.
6. To open JMeter, visit the downloaded Jmeter folder JMeter > bin > ApacheJMeter.jar
7. Double-click on the ApacheJMeter.jar file
8. Open the newly generated script
9. Select the correct file
10. Click Open
11. Run the script and view the results
12. Repeat the experiment by varying the number of users
13. Record the response time and throughput

Figure 8.7: Generate JMeter Script
The following tests were carried out:

1. Compare different activities
2. Response time for Quiz Activity
3. Login time
8.4.2 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).

<table>
<thead>
<tr>
<th>Activity</th>
<th>Response time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forum</td>
<td>626</td>
</tr>
<tr>
<td>Quiz</td>
<td>1439</td>
</tr>
<tr>
<td>Chat</td>
<td>509</td>
</tr>
<tr>
<td>Glossary</td>
<td>640</td>
</tr>
</tbody>
</table>

From the Table 8.1 it is clear that quiz activity takes the highest response time.

8.5 Quiz Activity

Different tests were carried out in quiz activity to measure the throughput by increasing the loop count and varying the number of users.

8.5.1 Increasing loop count

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

8.5.2 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.

8.5.3 Login

Tests were conducted to measure the response time of login activity by increasing the number of users.

8.5.3.1 Results

From these experiments, 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.

### 8.5.4 SQL QUERIES GENERATED FOR JMETER

This experiment analyses the queries generated by jmeter while running the script. The purpose of this test is to compare the queries generated manually with that of queries automatically generated by jmeter. This can be done by setting up the log file. The test is conducted for quiz activity. The Jmeter script is generated for quiz activity which performs

1. Login to site
2. View Course
3. View Quiz
4. Start 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.

#### 8.5.4.1 Experimental Setup

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

The queries can viewed in logfile located at /var/log/mysql/mysql.log

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.

#### 8.5.4.2 Result

The jmeter script performs all activities done by manual testing with lesser number of queries.
8.5.5 Jmeter Testing For Real Time Quiz Plugin

A new quiz plugin was installed in moodle called mcq_quiz. Multiple choice questions were added. The instructor can enrol any number of users and can start the quiz. After the quiz is started the students can join the quiz. After the quiz completes users can view the responses. Jmeter can be used to test the performance of this newly installed plugin using manual settings.

8.5.5.1 Steps

1. Open Jmeter
2. Select Test Plan on the tree
3. Right click on the Test Plan and add a new thread group:
4. Add \textit{\# Threads (Users)} \#Thread Group
5. Select the thread group
6. Right click Add -> Config Element -> HTTP Request Defaults
7. Enter server name as localhost
8. Right click Add -> Listener -> Summary Report to add an summary listener.
9. Right click on WorkBench and add the Http proxy:
10. Add -> Non-Test Elements -> HTTPProxy Server
11. Click the Start button at the bottom
12. Configure browser to use the JMeter HTTP Proxy
13. Start Iceweasel/Firefox, but do not close JMeter.
14. From the tool bar, click Edit -> Preferences (or Tools > Preferences)
15. Select the Advanced tab, and Network tab
16. Click Settings button near the bottom.
18. Address enter localhost or the IP address of your system
19. Port enter 8080.
20. Check Use this proxy server for all protocols
21. Record the navigation
22. Login to moodle and attempt the mdl_quiz
23. The recording controller will record all the navigations
24. Start the test

25. Run the test plan

26. The summary listener will show the statistics

Figure 8.10: JMeter Testing For Real Time Quiz Plugin Fig1

Figure 8.11: JMeter Testing For Real Time Quiz Plugin Fig2
CHAPTER 8. TESTING MOODLE WITH JMETER

Figure 8.12: JMeter Testing For Real Time Quiz Plugin Fig3

Figure 8.13: JMeter Testing For Real Time Quiz Plugin Fig4
8.5.5.2 Result

From the summary report, the throughput and response time of the new quiz plugin is obtained. The values are:

Throughput = 2.1/sec
Response time = 467 msec
Chapter 9

Implementation of Long Polling to realtime quiz plugin

9.1 Introduction

9.1.1 Concept of Long Polling

If you're developing an interface which requires continuous, real-time access to fast-changing data on the server-side, you will run into problems soon.

The most obvious implementation would consist of a timer on the client side which repeatedly triggers AJAX requests (once every 1-2 seconds) to the server side to check for new messages. In Computer Science, this is called polling. Polling over HTTP is, however, incredibly inefficient as these connections take time to establish and they also add stress on the server.

With long polling, the client places the request and the server doesn't reply until it has information to return. The Web client keeps a pending connection that's closed only when some valid response can be returned. That's exactly what we want. Long polling places a smaller number of requests to the server compared with AJAX polling. When real-time communication is required, but each request could take much longer.

9.1.2 Implementation over a simple php based quiz

Client-side:
As discussed the longpolling, client sends the request.
The requests made from the client-side will always feature a parameter describing the data set currently held by the client-side. The parameter describing it is the timestamp of the latest received message.
Including this parameter will instruct the server-side to respond only when new data is available.
The server then looks like this.

```php
while (!hasNewData())
    usleep(50);
outputNewData();
```
So, the AJAX request goes to the server, probably including a timestamp of when it was last update so that your hasNewData() knows what data you have already got.
9.1.3 Real Time Quiz Plugin

The teacher starts the quiz (optionally giving the quiz a 'session name'). Students can now connect to this quiz. Once the teacher is satisfied that all students have connected to the quiz, they can click on 'Next' to show the first question. The question will be displayed for a pre-defined amount of time, after which the correct answer will be displayed, along with a count of how many students gave each answer. The teacher can then discuss the question, before clicking on 'Next' to show the next question. Once all the questions have been shown, the final result for the class is displayed. The teacher can, at a later date, go back through the results and, for each question, see exactly what answer each student gave.

9.2 Functionality

start = make the quiz available and wait for students to join. (for instructor)
join = join to give the quiz which is made available. (for student)
next = pass a question to student screen.

9.2.1 Plugin Details

Physically, a Moodle plugin is just a folder of PHP scripts (and CSS, JavaScript, etc. if necessary). Moodle core communicates with the plugin by looking for particular entry points, often defined in the file lib.php within the plugin.

Files description of those Important in changing client server mechanism to long polling Realtimequiz plugin
1. View.php, This page prints a particular instance of realtimequiz. According to whether user is teacher or student. This page also includes the javascript files(view_student.js and view_teacher.js)

2. View_student.js:- contains the Code for a student taking the quiz.

3. view_teacher.js:- contains the Code for a teacher running a quiz.

4. Quizdata.php:- This dynamically sends quiz data to clients. Uses quiz stored in the database.

5. Teacher sends request type : nextquestion and

6. Student sends request type : getquestion.

7. There are different quiz status based on which state quiz is code execution follows they are:

8. define('REALTIMEQUIZ_STATUS_NOTRUNNING', 0);

9. define('REALTIMEQUIZ_STATUS_READYTOSTART', 10);

10. define('REALTIMEQUIZ_STATUS_PREVIEWQUESTION', 15);

11. define('REALTIMEQUIZ_STATUS_SHOWQUESTION', 20);

12. define('REALTIMEQUIZ_STATUS_SHOWRESULTS', 30);

13. define('REALTIMEQUIZ_STATUS_FINALRESULTS', 40);

9.2.1.1 Working

1. Instructor creates and edits quiz.

2. Any quiz held is named to keep record.

3. Instructor starts quiz(makes it available) and a student joins the quiz.

4. student waiting for teacher to send question.

5. Instructor sending quiz after all students connected

6. First question appears on student page

7. Immediate marking response.

8. Instructor forwards next question

9. question can be answered before time and answer is sent.

10. Analysis page of quiz for instructor.
9.3 Description of code for Initial Implementation: (Default Plugin)

Note: that this description is focused mainly on how client server communication is. Simple Polling (Repeated Request Response) method was used is Evident from the following points:

This (polling) is at two places in working of quiz: When teacher starts the quiz and students join and they wait for teacher to send the first question.Polling here enables the realtime coordination of teacher pressing next button and quiz appearing on student page.

Following snapshots showing the polling code to fetch 1st question.

```php
    case REALTIMEQUIZ_STATUS_READYTOSTART: // quiz is ready to start(q not sent)
        if ($requesttype == 'nextquestion') // if teacher
        {
            $userid = required_param('userid', PARAM_INT);
            realtimequiz_goto_question($context, $quizid, 1);
        } else
        {
            realtimequiz_send_wait_question(); // (don't care what they asked for)
        }
        break;
```

Figure 9.2: Snippet of quizdata.php

Send_wait_question(); sends a response to student with status of ”waitforquest”

```javascript
    // scope of removal when long polling
    else if (quizstatus == 'waitforquestion')
    {
        var waittime = quizresponse.getElementsByTagName('waittime').item(0);
        if (waittime) {
            waittime = parseFloat(node_text(waittime)) * 1000;
        } else {
            waittime = 600;
        }
        // resending the request until quiz status is READY_TO_START
        realtimequiz_delayed_request("realtimequiz_get_question()", waittime);
    }
```

Figure 9.3: Snippet of view_student.js

When result (of 1st quest) appears on student screen and again waits for teacher to send next question.
9.4 LONG POLLING IMPLEMENTATION

$realtimequiz.controlquiz variable is used to decide If it is teacher when result is shown, it enables the next button to allow teacher to send the next question. If it is student when result is shown, it starts first getquestion request, further requests are handled by (status=waitforrequest)

1) Since we are using long polling we no longer need to wait for the question and so the first editing we make is to comment out the following lines:

```
/*else if (quizstatus == 'waitforrequest') {
    var waittime = quizresponse.getElementsByTagName('waittime').item(0);
    if (waittime) {
        waittime = parseFloat(node_text(waittime)) * 1000;
    } else {
        waittime = 600;
    }
    realtimequizDelayed_request("realtimequiz_get_question()", waittime);
}
```

When the quiz is started: We use long polling while the student waits for the teacher to start the quiz using a while loop.

```
/*case REALTIMEQUIZ_STATUS_READYTOSTART: // Quiz is ready to start //status only for start
    if (requesttype == 'nextquestion') {// type 'nextquest' means teacher
        userid = required_param('userid', PARAM_INT);
        realtimequiz_goto_question($context, $quizid, 1);
    } else //for student

    { /* enter into while loop until quizstatus changes from READYTOSTART to SHOWQUESTION
        while($status = REALTIMEQUIZ_STATUS_SHOWQUESTION)
        { $status = realtimequiz_update_status($quizid, $status); }
        realtimequiz_send_question($quizid, $context);
    }
break;
```

We do not use the send await question, as we used long polling. When next question is required by the student after display result:
Again, as mentioned before, since we have implemented long polling there is no use of the await function to wait for the next question to be displayed.

9.5 Snapshots

Figure 9.8: Three questions have been set up by the teacher
Figure 9.9: Specify a name for the quiz and start it to enable the students to take it.

Figure 9.10: After the student connects, it waits for the question to be sent.
Figure 9.11: The teacher waits until all the students are connected.

Figure 9.12: When the teacher presses the next button, it becomes visible to the student and he has 30 seconds to answer it.
Figure 9.13: The question is not sent until the teacher presses the next button. Long polling goes on in this period.

Figure 9.14: After pressing next button the question gets displayed to student within next 3 seconds.
Figure 9.15: The first question gets displayed and the student has 30 seconds to answer it, no polling goes on in this period.

Figure 9.16: After answering all questions the student gets to know his performance.
9.6 Experimental Results

9.6.1 No. Of requests.

The plugin was tested over Jmeter and the long polling was evident from it because as compared to earlier when there used to be a stack of requests from single student for single question until sent. Now there is single request until each response one for question(till received) , one for posting answer, one for results. But there is no repetition.

9.6.2 Throughput

Figure 9.18: 90 users: default method.
CHAPTER 9. IMPLEMENTATION OF LONG POLLING TO REALTIME QUIZ PLUGIN

Figure 9.19: 90 users: long-polling.

9.7 Conclusion

For realtimequiz plugin communication technique changed from ajax polling to long polling, complete and working. Both the default and long-polling worked fine for 90 users but the later with an improved overall throughput.
Chapter 10

Implementation of Random batch-mode connectivity algorithm

10.1 Objective

All the techniques long-polling, random number generation, are experimentations over moodle aimed either to improve optimization, throughput, or increase number of users those operating simultaneously to take quiz or other activity. This technique random number generation to moodle is also aimed at testing over moodle and see if number of users or the throughput is improved. [21]

10.2 Introduction to random number generation

In an client server, communication for a certain upper limit of during simultaneous access, to further increase this limit of number of users, this technique can be used as algorithm which intakes number of users simultaneously accessing and assign them with a random number generated in such a way that users get grouped by same random number and this is equal to the round in which he shall download the quiz.

Random number generation code in PHP

```php
mycode--------start--------
$clients = 2;
numberOfConnectionRounds = 0;
$numberOfConnectionRounds = $clients / 25; //download will start when this will become zero (while exited , code followed)

/*if clients are less the 25 then they will finish in one round*/
if ($clients < 25)
$numberOfConnectionRounds = 1;

// increase the number of rounds if the clients remaining are more than 10
$extraClients = $clients % 25;

if ($extraClients > $10)
{
$numberOfConnectionRounds++;
}
CHAPTER 10. IMPLEMENTATION OF RANDOM BATCH-MODE CONNECTIVITY ALGORITHM

// also increase the round by one if the number of clients are greater then 100 as it is increasing alot of traffic
else if ($clients$>$100)
{
$numberOfConnectionRounds++;
}

while($numberOfConnectionRounds$>$0)
{
    //random_number(hence, the connectionProbability) based on no. of rounds(within no. rounds)
    $connectProbability= rand(1,$numberOfConnectionRounds);//using rand object

    if($connectProbability==1)//allow him to download
    {
        break;
    }
    $numberOfConnectionRounds--;//else wait for next round and sleep_t= download_t
    sleep(4);
}

//end of while//go out and download.............auto click attempt
//------------------------end of mycode-----------------

Here users are grouped into sets of 25 users for simultaneous quiz download.

10.3 Implementation in moodle quiz

10.3.1 Benefit

Moodle quiz itself is with whole lot of benefits of flexibility, different question types, different settings available, timed quiz etc, any enhancement to this shall be a lot of help.

10.3.2 Settings to use

Moodle quiz when done settings that all questions to be displayed at once, they are downloaded at one time, giving us scope to implement the Random number generation download method. The quiz plugin folder, consists of php, css, java script files, with their flow as:-

view.php > startattemp.php > other supportive php files .

View.php: contains the attempt quiz button with its action file being startattemp.php is triggered
Startattemp.php: authenticates the user and subsequently fetches the quiz and starts attempt.

The above code can be placed on the startattemp.php just before the quiz is downloaded and timer starts.
After Line 164 of startattemp.php in moodle/mod/quiz folder of moodle version 2.5
10.4 Results after Jmeter Testing

10.4.1 System specification

Processor : i3
RAM : 3GB
Operating system : Ubuntu 10.04

10.4.2 Experimentations over Jmeter

Observation : the default plugin was completed with system hanging problem, but later resumed to complete.
Conclusion: The upper limit for default quiz is 60 for this specifications.
Figure 10.3: 75 users: Random batch-mode connectivity algorithm.

Observation: The quiz was complete, with 1 user failed to attempt
Chapter 11

POLICY ENFORCEMENT FRAMEWORK FOR MOODLE

11.1 Introduction

Access control in computer systems and networks relies on access policies. The access control process can be divided into two phases:

1. policy definition phase where access is authorized
2. policy enforcement phase where access requests are approved or disapproved.

Authorization is thus the function of the policy definition phase which precedes the policy enforcement phase where access requests are approved or disapproved based on the previously defined authorizations. As moodle supports various types of quiz our aim is to implement policy enforcement framework interface so that it can be integrated with moodle on android. By this the instructor can control what all the apps that a student can access while taking a quiz.

11.2 Policy Enforcement Framework

Policy enforcement deals with users security and privacy concerns, by allowing them to define policy rules. When integrated with moodle as plugin, these policy enforcements can help the teacher/admin to block some apps, such as wikipedia during a quiz/test on a tablet. When the policy is created, an xml file is generated which can be sent to control the tablets. If the policy is valid, the other applications in the tablet can be either allowed/blocked. The context attributes defines time and location attributes so that the policy can be applied to specified locations and time.

The main goals of defining policy are:

- To restrict the usage of resources
- To prevent privilege escalation attack
- To provide fine-grained access control

The policy is created with a unique name. The validity of the policy can be defined and can applied to specific packages.
Some of the use-cases explaining why a policy enforcement framework is required are as follows:

- When quizzes or exams in schools, only the quiz or exam related apps should get open. Any request to start any of the remaining apps should be blocked.

- During school-time, students can open a limited set of apps.

- List of allowed apps would be defined by schools (teachers). For example, students should not be able to open social networking apps or gaming apps during school-time.

### 11.2.1 Context Attributes

The policy framework also allow users to define context attributes, that is, time and location. If the system time and location matches with the context attributes, then the policy is valid. Otherwise it is invalid and the next policy is considered.

### 11.2.2 Database Schema

The values of these attributes are stored in a database. These values are retrieved to generate a xml file which can be sent to control the tablets. The policy database consist of 5 tables namely policy, context, location, package and command.\[^{23}\]
### 11.2. POLICY ENFORCEMENT FRAMEWORK

<table>
<thead>
<tr>
<th>TAGS</th>
<th>TYPE</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of the policy.</td>
</tr>
<tr>
<td>Number</td>
<td>Int</td>
<td>A positive integer assigned to the policy. If ‘Priority’ of two policies is same, then ‘Number’ will be used to sort them (to choose highest priority). A smaller ‘Number’ means higher priority.</td>
</tr>
<tr>
<td>Valid From</td>
<td>DateTime</td>
<td>Indicates a DateTime from which a policy should be considered as valid. If the current (system) DateTime is before ‘ValidFrom’ DateTime, then that policy is not applicable.</td>
</tr>
<tr>
<td>Valid Till</td>
<td>DateTime</td>
<td>Indicates a DateTime till which a policy should be considered as valid. If the current (system) DateTime is after ‘ValidTill’ DateTime, then that policy is not applicable.</td>
</tr>
<tr>
<td>Priority</td>
<td>Int</td>
<td>A positive integer assigned to the policy. It is used to sort the policies. The lower the assigned value, higher is the priority.</td>
</tr>
<tr>
<td>Cmd</td>
<td>String</td>
<td>Possible values are: ALLOW, ALLOW_EXCEPT, ALLOW_ALL, BLOCK, BLOCK_EXCEPT, BLOCK_ALL. Whether to allow launching of an application depends upon value assigned to ‘Cmd’.</td>
</tr>
<tr>
<td>Context</td>
<td>String</td>
<td>Contains logical expression (using &amp;&amp;,</td>
</tr>
<tr>
<td>CA-Time</td>
<td>Time</td>
<td>Defined as &lt;ca-name&gt;&lt;from time&gt;&lt;to time&gt; Its value in logical expression is considered as true, if the current (system) time is within this range.</td>
</tr>
<tr>
<td>CA-Location</td>
<td>Int</td>
<td>Defined as &lt;ca-loc&gt;&lt;latitude&gt;&lt;longitude&gt;&lt;range&gt; If the current location (received from LocationManager) is within the range of mentioned location, then its value is considered as true.</td>
</tr>
<tr>
<td>Pkg</td>
<td>String</td>
<td>Specifies a package name.</td>
</tr>
</tbody>
</table>

Table 11.1: Explanation of various tags defined in Policy (xml) file\textsuperscript{23}
The various commands used are:

1. ALLOW : The specified package is allowed
2. ALLOW_ALL : All the packages are allowed.
3. ALLOW_EXCEPT : All the packages under the specified packages are allowed.
4. BLOCK : The specified package is blocked.
5. BLOCK_ALL : All the packages are blocked.
6. BLOCK_EXCEPT : All the packages except the specified packages are blocked.

11.3 Policy Enforcement Interface

The policy enforcement framework provides an interface to

- Create Policy
- Add context
- Edit policy
- Resequence
- Generate xml file

11.3.1 Create Policy

In this module the user is given an interface where he/she can declare and define policies with all the context information. The details like name, priority, description, command, valid from, valid till and package are entered.
11.3.2 Add context

The details related to the context, context name, from time, to time, location name, latitude, longitude and range are entered. A policy can have more than one context. There is an option to add more context for the policy.

![Add Context](image.png)

Figure 11.2: Add Context

11.3.3 Edit Policy

In this module the user can view all the policies defined and modify them. He can add or delete a context or change the validity of the project.
11.3.4 Resequence

There may be a case where two or more policies have same priority. As a result they conflict. So the user should be given an option to resolve the conflict arisen. This can be done by re-sequencing the policies that conflict. In this interface the user will be given a drag and drop kind of interface where all the policies available are grouped based on the priority. User can re-sequence them to solve the conflict so that the new sequence number will be updated in the database.
11.3.5 Generate XML Files

On selecting this, an xml file is generated. This xml file can be transferred to control other tablets.

11.4 Database Schema Screenshots

![Database Schema Screenshot 1](image_url)

Figure 11.5: Database Schema Screenshot 1
### Figure 11.6: Database Schema Screenshot 2

```sql
mysql> desc policy;
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>pname</td>
<td>varchar(20)</td>
<td>NO</td>
<td>UNI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>pnumber</td>
<td>bigint(20)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>int(11)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>command</td>
<td>varchar(15)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>valid_from</td>
<td>datetime</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>valid_till</td>
<td>datetime</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>longtext</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

7 rows in set (0.01 sec)

```sql
mysql> desc package;
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>bigint(20)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>pkg_name</td>
<td>varchar(20)</td>
<td>NO</td>
<td>UNI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>policy</td>
<td>bigint(20)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

3 rows in set (0.00 sec)

```sql
mysql> desc context;
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>cid</td>
<td>bigint(20)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>policy</td>
<td>bigint(20)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>cname</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>from_time</td>
<td>time</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>to_time</td>
<td>time</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

5 rows in set (0.00 sec)
11.5 Conclusion

The policy enforcement framework can be implemented as a plugin to moodle so that it reduces the traffic to moodle while moodle is running on a tablet.
Chapter 12

Future Scope

We have tried almost all the possible ways in which moodle system can be optimized. As per our knowledge still there are some more techniques that can be tested and implemented if the results are positive. These techniques may be studying the database and the PHP functions that generate SQL queries and try to minimize the queries as far as possible. There are many irrelevant queries that retrieves the data from the database, so we can try to remove those queries that are irrelevant. This can be done by identifying the most expensive queries and then try to minimize them.

Moodle can be integrated with different servers and efficiency can be checked. We can try to share the load generated on moodle to different servers so that it may speed up. Experiments can be done on this area and find out if there is such possibility.

The policy enforcement framework can be implemented as a plugin to moodle so that it reduces the traffic to moodle while moodle is running on a tablet. This may show large difference so it can be integrated on moodle and more experiments can be done on this part to see whether the results are positive.
References

https://moodle.org/stats. accessed on 28th june 2013.”


[8] “Groups er diagram. available at

[9] “Blocks er diagram. available at

[10] “Gradebook er diagram. available at
http://docs.moodle.org/dev/images_dev/e/e0/gradebook_erd.png . accessed on 28th june 2013,” May 2013.


[12] “Hard disk drive available at

REFERENCES


