Learning Management System: MS5 Final

Rabeet Fatmi, Rakesh Meka, Daniel Teichman (Lead)

Advanced Database Development

Spring 2016

Contents

[Project Description 3](#_Toc448926963)

[Problem Statement 3](#_Toc448926964)

[Expected Outcome 3](#_Toc448926965)

[Source code 4](#_Toc448926966)

[Instructions for Running the System 4](#_Toc448926967)

[Instructions for Using the System 5](#_Toc448926968)

[Screen Snapshots and Test Cases 5](#_Toc448926969)

[Known Issues 5](#_Toc448926970)

[Benchmark Results 6](#_Toc448926971)

# Project Description

We constructed a distributed database architecture and sample application for a scalable Learning Management System (LMS) using a relational and document-store or key-value clusters. The distributed database architecture will employ clustering and replication techniques to demonstrate safeguarding of data located in geographically disparate data centers. We will proceed by gathering LMS requirements, creating a high-level design of an application that meets those requirements, designing a database to support the application, and then creating application code to demonstrate some features of an ideal LMS application.

# Problem Statement

As of now, Blackboard and Canvas are the two major competing LMS applications for universities. Both products are commonly hosted by the vendor and these products are costly and not within the institution's direct control. We seek to reduce the cost of using a high-quality LMS by creating a free open-source (FOSS) LMS application that can operate with organizationally-controlled dedicated server hardware while maintaining safekeeping of documents in a distributed database with geographically disperse cloud servers. We seek to reduce the per-student cost of providing LMS services to universities globally with the hope that universities will benefit from a scalable FOSS LMS as much as other organizations have benefited from other FOSS projects such as Linux, a preeminent server operating system.

# Expected Outcome

We present a distributed database design and architecture to support an LMS. Our design will feature replication to guarantee security of the data in the event of a hardware failure of a dedicated application server. The software application, contains a set of features that allow for course management tasks, and assignment management tasks. The scope of features is a minimum to demonstrate a suitable database architecture in support of a useful LMS application. Scalability testing consists of a benchmark to simulate a large number of students simultaneously downloading course materials and posting assignments to determine that the software is ready to serve the needs of a large university.

# Source code

Source code provided separately from this document, in enclosing ZIP file.

# Instructions for Running the System

Our system requires a PostgreSQL instance and one or more Riak nodes in a (secure) cluster configuration. To implement security for Riak, we recommend connecting the nodes with a VPN so that Riak is not exposed to the Internet directly.

The application environment requires a Linux host running Apache 2 with PHP5 and the PHP5 PostgreSQL library. The contents of our application’s source code folder should be deployed to any folder served by the web server so as to create any URL desired.

The file, **config.php** must be edited by hand to create the desired configuration.

Relevant items in **config.php** are as follows:

* Line 4 defines HTTP\_SCRIPT\_HOME, which should be the absolute URL file path of the directory the web server directory project is stored in. For example, if the project’s index.php is located at http://yourserver.com/one/two/index.php then this variable should be set to /one/two
* Line 6 and 7 contain, by default, abbreviated role names for professors and students. More of these can be added.
* Lines 9 through 14 contain the information needed to connect to and authenticate with the PostgreSQL database.
* Lines 16 and 17 contain configuration information for the Riak cluster. By default, there are two nodes. More nodes can be added to the cluster by adding more node configuration lines with non-conflicting identifiers (ex. “node1” and “node2” are the default identifiers and other nodes must not use those array subscripts).

It is also important to adjust the maximum file size for uploaded files in your PHP configuration because the default is usually very low with respect to today’s large binary file sizes and this default may vary by OS vendor.

# Instructions for Using the System

Rabeet, fill this in.

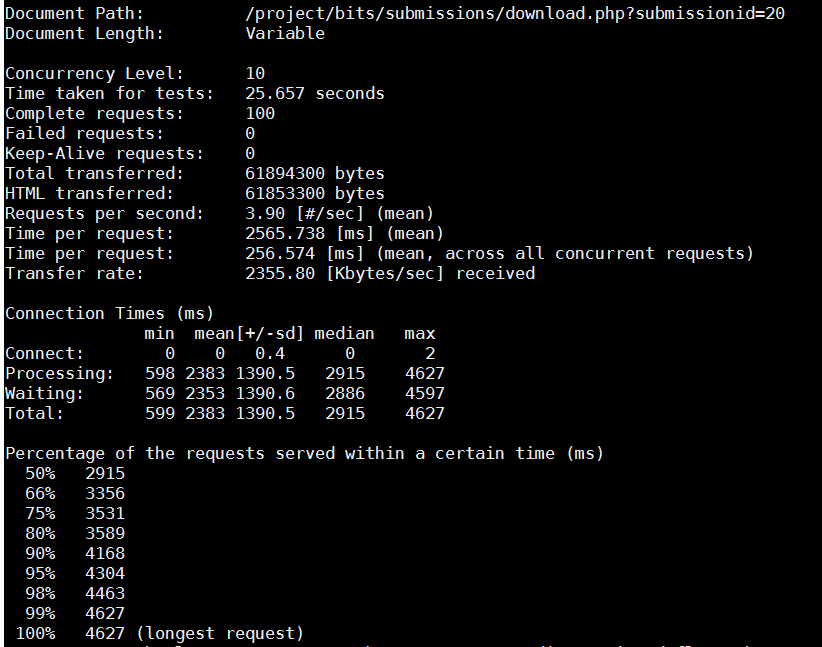
Important feature: The system can detect a submission-upload that does not fully commit. Our file submission code in /bits/submission.php initially creates a submission in Postgres that is marked incomplete (‘done’ = ‘false’). Then, the file is uploaded to Riak with the Postgres insert id. Only if that is successful can the submission be subject to an update so that (‘done’ = ‘true’). This is analogous to a distributed transaction but, if the submission is initially created *undone* and the file upload to Riak fails, then it is detectable that there was an attempted submission and that the system failed instead of the student. Otherwise, it is guaranteed that the submission is fully committed and written to the relational database and no less than two Riak nodes.

# Screen Snapshots and Test Cases

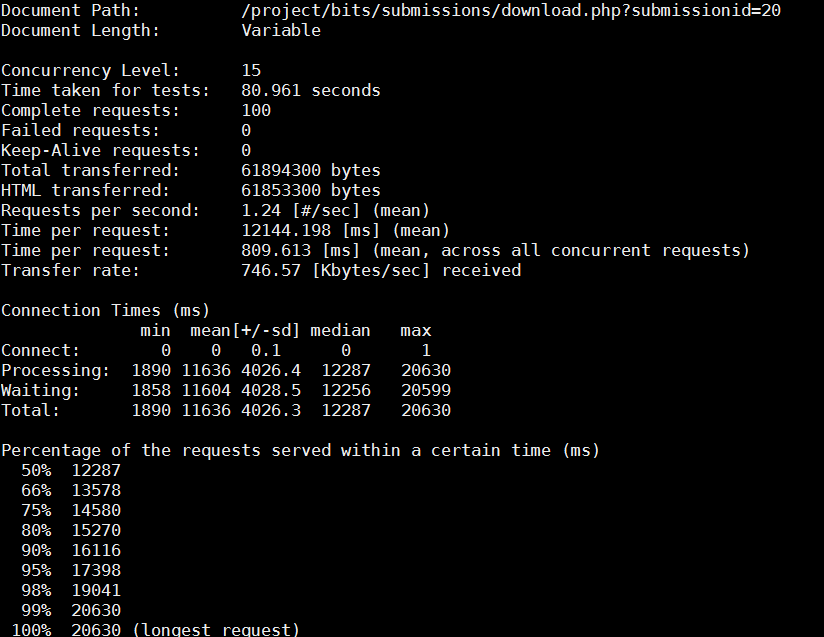
# Known Issues

First, downloading files does not download the file with a “friendly” filename. Right-clicking the download link and choosing “save as” does accelerate this process.

# Benchmark Results



A benchmark downloading of a sample homework submission results in every request being downloaded (with no caching apparatus) in under 5 seconds with an average of 2.9 seconds for 100 requests with a concurrency of 10 simultaneous requests.



The same benchmark repeating with concurrency level of 15 simultaneous requests reveals a limit to performance with requests having a response in 12-21 seconds with a median response time of 12.3 seconds.

With a maximum concurrency of 10 simultaneous assignment downloads, we expect that a single host is not suitable for a university-wide LMS where concurrency may reach levels far in excess of 10 simultaneous downloads but this problem needs to be studied as we do not know the true concurrency of any commercial LMS.