BeaverCMS : 6.033 Design Project

Fiona Zhang, Nyle Sykes, Ashwath Thirumalai

Massachusetts Institute of Technology

March 22, 2019

1 Introduction

The staff of the MIT course 6.033, the required undergraduate course on computer systems design, has decided to upgrade its current fragmented grading infrastructure. Currently, the infrastructure splits submitting assignments and viewing grades across multiple different, unconnected systems. Furthermore, it does not support forming teams, sharing works in progress, and submitting group assignments and necessitates manual verification for team assignments, deadlines, and penalties. The 6.033 staff wants to modernize this infrastructure by creating a centralized system to facilitate student team work, provide a unified place to view grades, and support additional video upload functionality while retaining similarly rigorous permissions from the current implementation.

To meet these goals, we propose the Beaver Course Management System (BeaverCMS). This system integrates various MIT-provided modules for file systems, locking, syncing, and identity verification, Gradescope, and a central server containing multiple relational databases. The system aims to abstract much of the work currently done manually and provide a seamless, central location to submit and grade assignments, provide and view feedback, and collaborate on the Design Project. The sections that follow specify our implementations of the modules and integration between them as well as how they achieve our key design goals of simplicity, security, and reliability.

2 System Design

BeaverCMS extends and integrates five existing systems: the MIT Identity Service (MIDS), MIT File Service (MFS), MIT Sync Service (MSS), MIT Lock Service (MLS), and Gradescope. These modules and their relationships are illustrated in Figure 1 below.

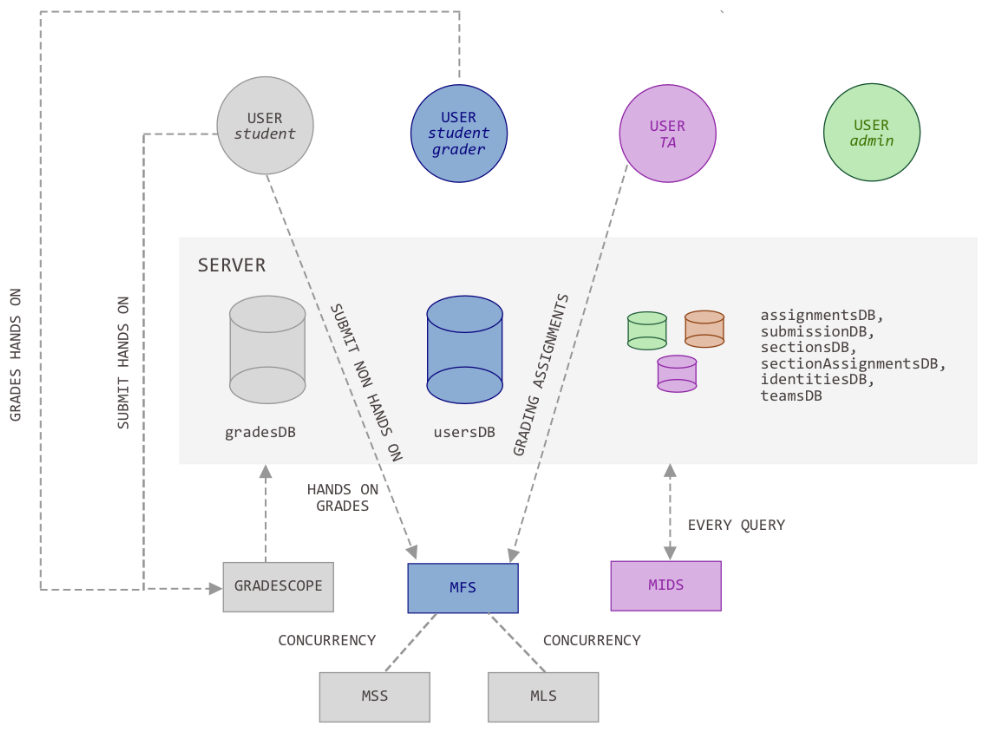

Figure 1: System overview

The core of this system is a single, centralized mirrored-disk server that stores various relational databases. This central server connects users, who can be students, student graders, or members of the course staff, to the other modules in the system. The design of this system prioritizes simplicity, security, and reliability. Simplicity allows the students and staff to focus on the material of the course rather than worry about a complex grading or submission system. Security is crucial in maintaining users' privacy regarding grades and assignment submissions. Lastly, the design focuses on reliability over performance since confidence in the system from both the student and the staff is more important than the time required to upload an assignment or provide feedback.

2.1 Central Server and Data Structures

The centralized server stores all of the information about the class, including, but not limited to, students, grades, assignments, teams, and recitations. To achieve simplicity in our data structures, our server stores and maintains a variety of relational databases linking students, staff, assignments, and grades. These databases facilitate modifying and quickly accessing information and support SQL querying. Table 1 below lists the databases as well as the variables that they store.

Table 1: Database Contents

Database Name	**Contents**
gradesDB	Student Name (Kerberos name), Assignment ID, Number of Points Received for Assignment, Status (On Time/Late), Timestamp of Last Query, Published (True/False)
assignmentsDB	Assignment Name, Assignment ID, Total Possible Points, Assignment Due Date
usersDB	User Kerberos Name, User Status (Active/Inactive), User Role (Student/Student Grader/TA/Administrator)
teamsDB	Student Kerberos Name, Team Kerberos ID
sectionsDB	Section ID, Recitation Team ID, Section TA Kerberos Name, Section Instructor Kerberos Name, Section Location, Section Time, Section Type (Recitation/Tutorial)
sectionAssignmentsDB	Student Kerberos Name, Assigned Section ID
identitiesDB	Kerberos Name, Home Directory in the MIT File System (MFS)
submissionDB	Kerberos Name, File Name, Assigment ID, Timestamp, Final (True/False), On Time (True/False)
recitationTeamsDB	Kerberos Name, Recitation Team ID

We choose to use relational databases because they are highly flexible, well established, and simple to maintain. Although they struggle with more complex data-types, this is likely not to be a concern with this specific use case.

2.2 Identities and Access Control

BeaverCMS's users include students, student graders, and members of the course staff. The system heavily prioritizes security, as this is a strength of the current 6.033 infrastructure. Strong security protocols require managing and authenticating the identities of individuals accessing the system.

Every query from a user to BeaverCMS must first pass through MIDS to verify the identity of the user. While each student and member of staff has a MIDS identity associated with their Kerberos ID, MIDS also supports the creation of additional Kerberos IDs. Our system leverages this functionality to create additional Kerberos IDs for Design Project teams, recitation team staff, recitation teams, and the whole class. Since MIDS does not support querying for specific roles or checking if a student belongs to a certain Design Project team or recitation, our system stores these relationships in the databases in the central server.

Not only is the specific individual's identity authenticated by MIDS, these databases allow for secure access control to specific files in MFS. A simple query can verify that a recitation leader has permissions to view a file of one of their students or that the Course Administrator has permissions to view the grades of the entire class. Since these databases are easily maintainable, a comparable level of security to the current implementation is thus kept in this new system.

2.3 File System

BeaverCMS augments the existing MFS interface to create a customized file system for 6.033 that distinguishes between different submissions of the same assignment and stores staff feedback.

Individual and Design Project team identities are automatically provided corresponding home directories in the file system. An individual student's home directory provides a location for submissions of an individual assignment such as a system critique, and a Design Project team's home directory provides a location for students in the same team to work in a shared file space. Upon creation of a new identity in MIDS, a protocol creates a "FINAL" directory in the home directory and grants the recitation team staff identity access. This directory stores the most recent (or user-chosen) file to be graded, while all other submitted files exist in the main home directory. Figure 2 below illustrates this file structure.

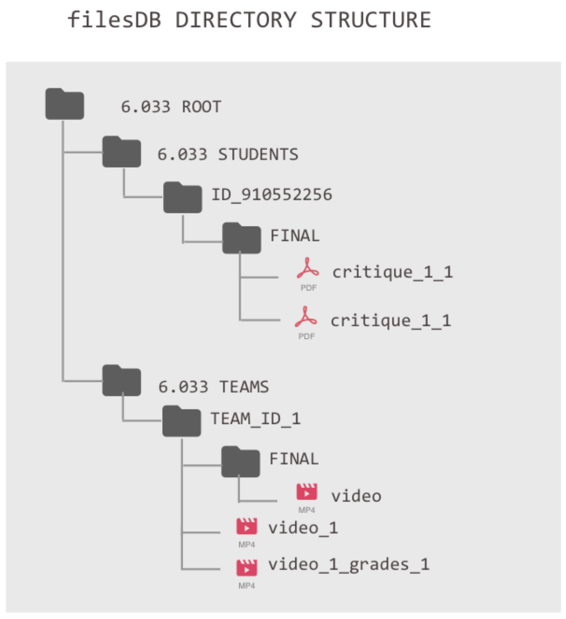

Figure 2: MFS contains a directory for each student and each team.

The simplicity of the system is enforced through a simple nomenclature for submitted assignment files that distinguishes between different assignments, the number of that assignment, and the number of that submission. This design goal is furthered through the handling of "FINAL" submissions. The most recent file is assumed to be the final one; however, the user can designate an alternate submission, which simply swaps this file with the file from the "FINAL" directory.

This file system also handles staff grades and comments. When a staff member submits grades and comments for a given submission, this file is stored in the "FINAL" directory with a similar naming scheme to differentiate between multiple graders submitting multiple sets of comments.

3 Networking

The following section details specific interactions between modules outlined above and corresponding key design decisions.

3.1 MFS Submissions

BeaverCMS's submission protocols are designed to optimize for reliability. This is achieved through failing fast in the case of errors and redundancy in the central server.

BeaverCMS stores a queue of submission requests. The system processes requests at the top of the queue and allows for up to four concurrent submission processes to run on the central server. A submission request consists of the Kerberos ID of the sender and the properly named assignment file. Allowing four concurrent submission processes to run at the same time accelerates the upload process and ultimately improves the reliability of the system. If a file slows down one submission process or if there is high traffic due to many students attempting to upload at once, this system has built-in redundancy to ensure that a student's valid submission will be recorded and stored.

BeaverCMS processes a submission request by attempting to authenticate the Kerberos ID by querying MIDS and receiving the relevant Kerberos name in case of success. BeaverCMS then ensures that the Kerberos name is in *usersDB*. If either of these authentication steps fail, then the system fails fast and returns an error to the client stating that the provided Kerberos ID is invalid. Failing silently for user submissions should be avoided to ensure that the student knows that they must re-attempt the submission.

After authentication, BeaverCMS then connects to MFS and uses the *write_file* function to write the submission file to the directory corresponding to the Kerberos name returned from MIDS. Specifically, the file is written to a subdirectory under the name of the assignment ID. BeaverCMS can find the directory path by looking at the *identitiesDB* relational database table.

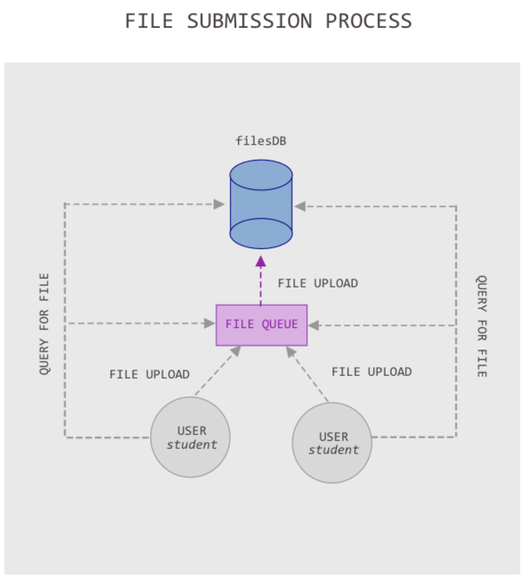

Figure 3: When a file is submitted, it is placed in a queue. The client then periodically queries the queue and database to determine the status of their upload.

To further bolster reliability of the system, BeaverCMS also provides functionality for a client to verify whether an upload was successful. If a client sends a verification request with a Kerberos ID and a file name, BeaverCMS can authenticate the Kerberos ID and check in the corresponding directory if a file with the given file name exists. This protocol returns a true/false result to the client. Figure 3 illustrates the work flow of submitting a file and periodically checking to ensure receipt.

3.2 Video Submissions and Voting

Our system supports the functionality for Design Project teams to submit 5 to 8 minute videos and view and vote on other teams' video submissions. Since these files are significantly larger (approx. 100 MB) than other files, we implement submission differently in order to maximize reliability.

Videos are stored in the directory that is visible to all 6.033 users (i.e the home directory of the whole class identity) and named with their team Kerberos name. Teams must use MSS to connect to MFS in order to upload their video. Specifically, they use the *sync_upload* function to upload their video to the class folder. When a student or team would like to view another team's video, they can run the *sync_download* function through MSS. We choose to use MSS instead of the MFS submission process for video submissions so that reads and writes are synchronized and to allow upload and download processes to be terminated if they take too long. A student can submit their ranking of their top 5 videos like any other file submission to BeaverCMS.

3.3 Peer Review Submissions

Design Project teams must submit their DPR using their recitation team identity in order to be peer-reviewed. This places the files into the home directory of the recitation team, allowing anyone with that recitation team identity to access the files and review their assigned report.

Upon completion, each student can submit their peer review like any other submission, described above in 3.1.

3.4 Staff Grading

BeaverCMS prioritizes simplicity and security when handling grading of submitted assignments.

In order for a staff member to grade assignments on the MFS, they must pull the assignments from the file system, grade them and write comments in a file, and query the server to submit the grade and file with comments.

When a recitation team staff member wants to download in bulk all of the submissions from their recitation team for a given assignment, they can send a request to BeaverCMS with the assignment name. BeaverCMS can query *recitationTeamsDB* to receive a list of student Kerberos names and query MFS using *download_file* to extract and return the desired files from the "FINAL" directories of these students. If this staff member only wants to download a specific student's submission for one assignment, they can query BeaverCMS with the student's Kerberos name and assignment name, and BeaverCMS will again query MFS to extract and return the desired file. Both of these methods of extracting submitted assignments from the file system abstract away almost all of the work from the staff member, achieving greater simplicity than the current system.

If either of these cases fail, the only explanation would be a lack of permission to access the student's directory. This could be due to a miscommunication or typographical error. In either case, BeaverCMS forwards this error back to the client.

Upon grading the received submission, the staff member will send BeaverCMS the grade and/or comments along with the student's Kerberos name and assignment name. BeaverCMS names the comments file accordingly and adds it to the student's or team's "FINAL" directory and adds the grade to the *gradesDB* database.

For bulk grading submissions, BeaverCMS maps each student's Kerberos name to their grade and comment files and processes each request one at a time.

In order for the publication function to work and for all students to see their grades at the same time, the grades are initially designated as unpublished and the client will not display those grades to the student. In order to publish the grades, the staff member runs the publish function along with the name of the assignment. This will update the status of the grades such that the client allows the students to view their grades.

If there are multiple graders grading the same submission by the same student for the same assignment, the system treats this as multiple grading events which corresponds to multiple entries in the *gradesDB* database.

3.5 Gradescope Integration

Pulling grades from Gradescope is the only case where BeaverCMS must interact with a third-party system. As such, this protocol is made as simple as possible to avoid over-reliance on third-party support.

To pull grades from Gradescope, calls to the *pull_gradescope_grades()* function are made periodically up until the regrade period is over. The result is stored as a CSV data structure and then parsed into inputs for *gradesDB*. The timestamp of the last query is also stored in *gradesDB* for convenience.

3.6 Entering Grades

For certain assignments, recitation team staff can manually enter the grades for each student or team. In this case, the recitation team staff can send such a request to BeaverCMS with either a single student or team Kerberos and a grade, or a dictionary mapping Kerberos names to grades. BeaverCMS will authenticate the staff member and ensure the student(s) are in the recitation team, and then add these values to the *gradesDB* database.

3.7 Late Assignments

In the current implementation, deadlines and grading penalties are handled manually. Upon a submission to the MFS, BeaverCMS queries *assignmentsDB* and compares the listed due date with the current date. If this assignment is late, it marks it as such in *submissionDB* and notifies the recitation leader of the late assignment. The recitation leader can then manually change this assignment's grade as discussed above. While it may be simpler to automate grade penalties, we sacrifice simplicity here for flexibility, as we value the accommodating nature of the current manual system.

3.8 Killing Transactions

In the case of a brown-out, the client automatically kills file uploads in process. For video uploads, the queue is automatically cancelled.

We implement a modified TCP such that it can process a kill signal sent from MSS. When a kill signal is issued, it signals a range of packets to ignore. If those packets have already been received, they are discarded. Otherwise, future packets in the desired range are ignored. Once all the packets in the range have been accounted for, the server sends back a success signal to indicate that the file transfer has been successfully killed. This process is illustrated in Figure 4 below.

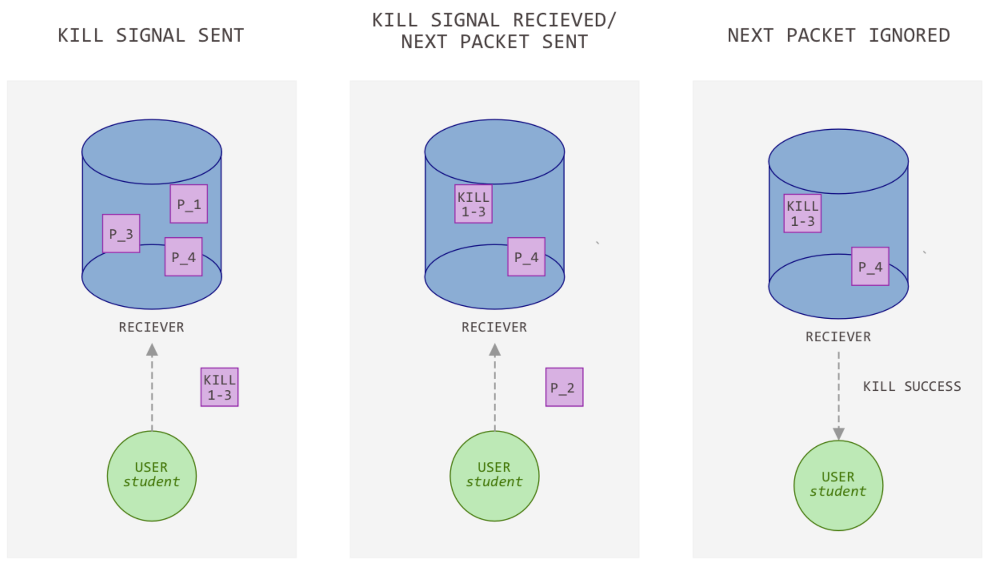

Figure 4: A kill signal is sent for packets 1-3. Packets 1, 3, and 4 have already been received. After receiving the kill signal, packets 1 and 3 are discarded. When packet 2 is received, it is also discarded. Finally, a kill success signal is sent pack to the client.

We choose to implement kill transaction for video submissions this way in order to ensure that files are always correctly uploaded. Partially uploaded files present a potential threat to the reliability and integrity of the system.

4 Conclusion

Our proposed system accomplishes the goals set out by the 6.033 staff: support of forming teams, sharing works in progress, submitting group assignments, providing a uniform place to view grades, and allowing for additional video upload functionality. Built with simplicity in mind, our file system uses an intuitive naming structure, and our grading process automates much of the work on behalf of the course staff. Our system achieves a comparable level of security as the current implementation through integration with MIDS and proper access controls on our databases. To ensure reliability in our system, our central server supports parallel submission processes and is fault tolerant in the case of a brown-out.