A Comparison of Technologies for Developing Web-Based Online Multiplayer Games

Mitchell Petit and Yi Liu
Department of Electrical Engineering and Computer Science
South Dakota State University
Brookings, SD 57007
mitchell.petit@jacks.sdstate.edu, yi.liu@sdstate.edu

Abstract

Real-time interactive experiences between users of web applications are unmatched by online multiplayer games. By themselves, online multiplayer games have the potential to bring in advertising revenue as a website. Additionally, online multiplayer games often contain many features that can be used in other web applications requiring real-time user interaction. However, there currently exists no rigorous comparison between web technologies for the purpose of creating real-time online multiplayer games. With this research, we aim to provide guidelines for selecting an operating system, server framework, and database solution for the use of developing online games. Since an exhaustive comparison of various software would be nearly impossible, selection of software for comparison was limited to only some of the more popular software. For the comparison of the software, various prototypes of a simple online multiplayer game were written in each of the software environments. The game that was implemented had multiple players in one room – also known as a game world – each controlling a spaceship. Players would move their ship and shoot asteroids and other ships. Players could also write and broadcast messages to each other.

For operating systems, both Windows and Linux (specifically, Ubuntu Server) were compared based on compatibility with the other software in the comparison, as well as memory use, size, and cost. For server frameworks, NodeJS, Django, and ASP.NET were compared based on the features of their language, the ability to write prototypes of the simple online game in the framework, and any other features the framework may support. For databases, both PostgreSQL and MySQL were compared based on support for the previously mentioned frameworks and data manipulation and retrieval times. For each comparison criterion, each software was given a score. These scores were weighted by criterion for a weighted score. The weighted scores of each criterion were then totaled for each software. Comparison of server frameworks were done on the operating system with the best score. Comparison of databases were done on the operating system and server framework with the best scores. A final prototype of the game was built on the operating system, server framework, and database solution with the best scores.

1 Introduction

Real-time interactive experiences between users of web applications are unmatched by online multiplayer games. By themselves, online multiplayer games have the potential to bring in advertising revenue as a website. Additionally, online multiplayer games often contain many features that can be used in other web applications requiring real-time user interaction. However, there currently exists no rigorous comparison between web technologies for the purpose of creating real-time online multiplayer games.

With this research, we aim to provide guidelines for selecting an operating system, server framework, and database solution for the use of developing online games. Since an exhaustive comparison of various software would be nearly impossible, selection of software for comparison was limited to only some of the more popular software. For the comparison of the software, various prototypes of a simple online multiplayer game were written in each of the software environments. The game that was implemented had multiple players in one room – also known as a game world – each controlling a spaceship. Players would move their ship and shoot at other ships. Players could also write and broadcast messages to each other. For operating systems, both Windows and Linux (specifically, Ubuntu Server) were compared based on compatibility with the other software in the comparison, as well as memory use, size, and cost. For server frameworks, NodeJS, Django, and ASP.NET were compared based on the features of their language, the ability to write prototypes of the simple online game in the framework, and any other features the framework may support. For databases, both PostgreSQL and MySQL were compared based on support for the previously mentioned frameworks and data manipulation and retrieval times. A link to a repository containing the code and test files will be made available at https://mitbit01.github.io/permenent/web-based-game-tech-comparison.

Section 2 presents the methods used to compare the technologies. Section 3 illustrates the results from the experiments and section 4 discusses the results and concludes the work.

2 Methods

Each technology will be compared against other technologies in its category (operating systems, server frameworks, databases). Each category contains criteria for evaluating the technologies. Each criterion contains a method for providing a weighted score. Technologies are scored and weighted in each criterion, which are summed to provide a total score for the technology.

2.1 Operating System Comparison Methods

The operating systems were compared by the criteria in the table below. All criteria are weighted negatively to keep the greatest score the best. Cost was weighted -1 as a baseline. Usage criteria were given weights of -4.37 and -0.022 given the average cost per GB in

2013 of memory [2] and storage [3], respectively. Compatibility criteria were given weights of -0.12 based on the US federal minimum wage of $7.25/h [1] and the assumption that each piece takes 1 minute.

Criterion	Method of Scoring	Weight
Cost	The purchase price in USD	-1
Memory Use	The minimum required memory amount in GB	-4.37
Storage Use	The minimum required storage amount in GB	-0.022
Compatibility with NodeJS	The number of manually installed, independent pieces of software plus the number of manual changed configuration files to install NodeJS	-0.12
Compatibility with Django	See compatibility with NodeJS	-0.12
Compatibility with PostgreSQL	See compatibility with NodeJS	-0.12
Compatibility with MySQL	See compatibility with NodeJS	-0.12

Table 1: Criteria for comparing operating systems

2.2 Server Framework Comparison Methods

Both NodeJS and Django tests were run on the Linux VM from the previous section. Additionally, they were both the same software version used in the operating system comparison. To test the server's game prototypes, the server received requests from the Windows machine from the previous section. Each server implemented a simple online game. Players could use their mouse to orient their spaceship, accelerate, and shoot at other spaceships. Players could also type into a simple form to chat with other players connected to the server.

Each server was compared by the criteria in the table below. Total Size of Code was weighted -1 so that less code is better. Weights of 10 were given to Language Features and Performance because of the slight advantages they can provide when compared to the Total Size of Code. Real Time Communication was weighted -2 since additional software only requires slightly more effort than writing code.

Criterion	Method of Scoring	Weight
Total Size of Code	The lines of code of the server, including delivered resources, excluding auto-generated code.	-1
Language Features	The number the following features of the server's language: Inheritance, Type Checking, Events, Asynchronous function calls, and Generator functions.	10
Performance	The responsiveness of the game compared against other servers in this category. The best receives a 2 while the worst receives a 1.	10
Real Time Communication	The number of manually installed, independent pieces of software to enable real-time communication support in the server.	-2

Table 2: Criteria for comparing Server Frameworks

2.3 Database Comparison Methods

Both PostgreSQL and MySQL tests were run on the Linux VM from the above sections. Both databases implemented the design shown in the figure below. Since both databases were tested the same, it was sufficient to have the accounts table to contain 27 rows, highscores table to have 21, alliances to have 6, and alliamceMembers to have 10.

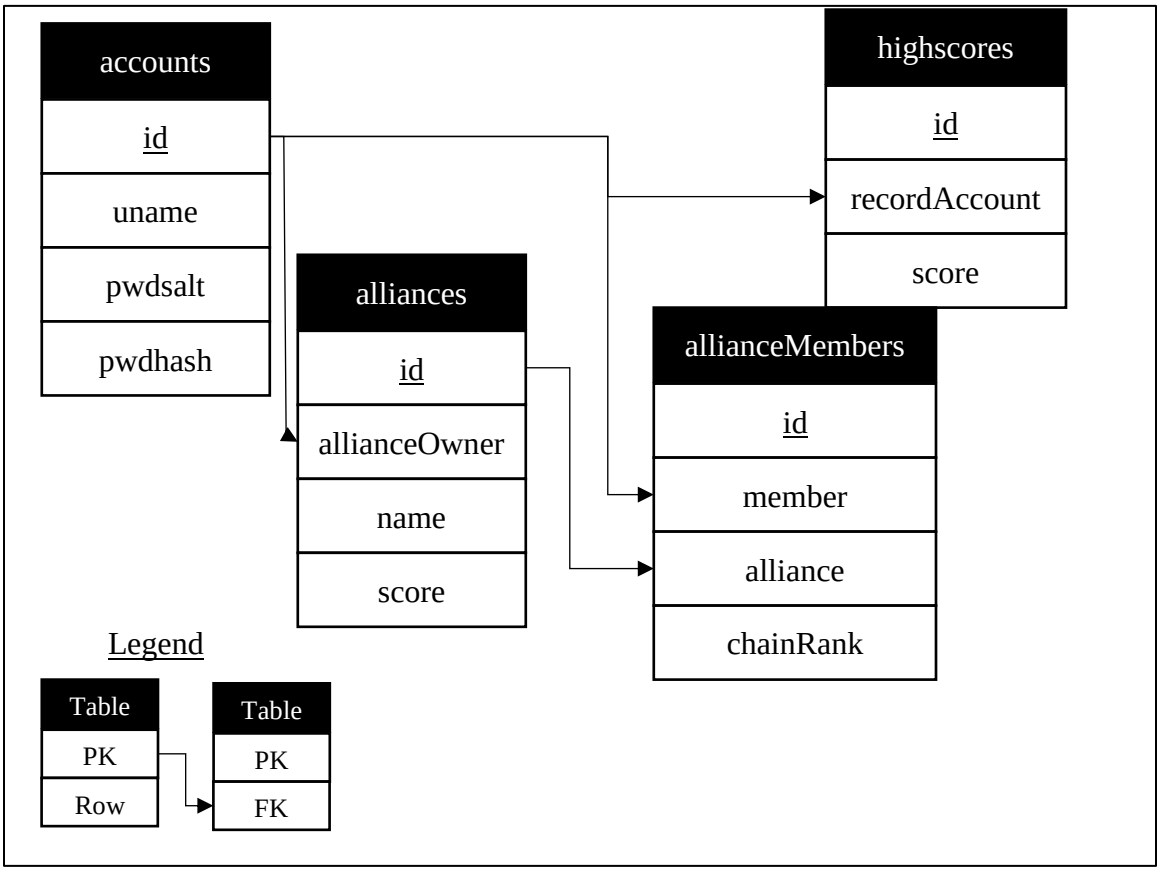

Figure 1: Database Design

Each database was compared by the criteria in the table below. Each criterion is an average of 3 times to run a SQL file containing one or more commands. Each criterion has a weight of -1 so that the least amount of time across all commands is best. Time was calculated using the Linux "time" command's real output for running the scripts.

Criterion	Method of Scoring	Weight
CREATE Script	The average time taken to run the script in seconds. This script first DROPs then CREATEs the tables in the design in Figure 1 above.	-1
Single INSERT Script	The average time taken to run the script in seconds. This script inserts data into the accounts table in a single INSERT statement.	-1
Many INSERT Script	The average time taken to run the script in seconds. This script inserts data into the accounts table in multiple INSERT statements.	-1
UPDATE Script	The average time taken to run the script in seconds. This script manipulates data in the accounts, highscores, and allianceMembers tables.	-1
SELECT Script	The average time taken to run the script in seconds. This script runs multiple SELECT, each growing in the number of clauses like WHERE and INNER JOIN.	-1
DELETE Script	The average time taken to run the script in seconds. This script simple DELETEs a number of rows from accounts, which in turn removes some rows from dependent tables.	-1

Table 3: Criteria for comparing Databases

3 Results

This section presents the results of comparing the operating systems, the server frameworks and the database management systems.

3.1 Operating System Results

The operating systems were chosen based on familiarity and availability, though this comparison could be extended to include other operating systems in the future. Windows 10 Pro version 1709 was chosen for the Windows OS. Ubuntu Server 16.4 was chosen for the Linux OS. For testing, Linux was run in a virtual machine on the Windows machine. Both machines were 64-bit.

For Windows, the cost, memory use, and storage use was $199.999, 2 GB, and 20 GB [4], respectively. For Linux, the cost, memory use, and storage use was $0, 0.5 GB, and 2.5 GB [5], respectively. NodeJS version 8.10.0 was simply installed on each OS. Django version 1.11.7 was installed on each version, but Windows required a separate python installation

while Linux required a separate pip installation. PostgreSQL version 9.6.8 was installed on both, but Linux also required separate signatures and repositories. MySQL version 14.14 was installed on both, but Linux also required a separate repository. The scores were calculated and are tabulated below.

Criterion	Windows Weighted Score	Linux Weighted Score
Cost	-199.99	0
Memory Use	-8.74	-2.185
Storage Use	-0.44	-0.055
Compatibility with NodeJS	-0.12	-0.12
Compatibility with Django	-0.24	-0.24
Compatibility with PostgreSQL	-0.12	-0.36
Compatibility with MySQL	-0.12	-0.24
Total Score	-209.77	-3.2

Table 4: Operating System Results

3.2 Server Framework Results

For the Language Features, NodeJS uses JavaScript and Django uses Python3. Of the evaluated features, JavaScript and NodeJS only fully has 2 of the features in the criteria: Inheritance and Events. Python and Django have 4 of the features: Inheritance, Type Checking, Asynchronous function calls, and Generator functions. Despite JavaScript having types, they aren't as protected as they are in Python due to implicit type coercion. For Real Time Communication, NodeJS only required Socket.io, while Django required Docker, Redis, and Channels. For performance, the Django version was much slower than the NodeJS version of the game, which had no noticeable delay. Scores were calculated and are tabulated below.

Criterion	NodeJS Weighted Score	Django Weighted Score
Total Size of Code	-394	-415
Language Features	20	40
Performance	20	10
Real Time Communication	-2	-6
Total Score	-356	-371

Table 5: Server Framework Results

3.3 Database Results

Below is a table containing the times of each run for the script for each database, including the average.

PostgreSQL Time (s)				
Criterion	Time 1	Time 2	Time 3	Average
CREATE Script	0.175	0.592	0.199	0.322
Single INSERT Script	0.134	0.105	0.038	0.092
Many INSERT Script	0.064	0.075	0.148	0.096
UPDATE Script	0.057	0.043	0.063	0.054
SELECT Script	0.034	0.039	0.035	0.036
DELETE Script	0.038	0.161	0.035	0.078
MySQL Time (s)				
Criterion	Time 1	Time 2	Time 3	Average
CREATE Script	0.104	0.227	0.246	0.192
Single INSERT Script	0.008	0.017	0.007	0.011
Many INSERT Script	0.019	0.021	0.019	0.020
UPDATE Script	0.013	0.010	0.013	0.012
SELECT Script	0.007	0.007	0.007	0.007
DELETE Script	0.005	0.004	0.005	0.005

Table 6: Database Script Averages

Scores were calculated using the table above and are tabulated below.

Criterion	PostgreSQL Weighted Score	MySQL Weighted Score
CREATE Script	-0.322	-0.192
Single INSERT Script	-0.092	-0.011
Many INSERT Script	-0.096	-0.020
UPDATE Script	-0.054	-0.012
SELECT Script	-0.036	-0.007
DELETE Script	-0.078	-0.005
Total Score	-0.678	-0.247

Table 7: Database Results

4 Discussion

By finding the technology with the greatest score in its category, we find that Linux, NodeJS, and MySQL provide the best features, usability, and performance with the least cost and hassle for creating a real-time, online, web-based multiplayer game with mouse controls, player competition, and text-chat. Linux was far superior to Windows for this case, with cost being the greatest factor. Usage requirements were far more favorable on Linux as well. NodeJS outperformed Django in every category except language features. Even with similar code size, which was the largest factor, NodeJS did much better in the

real-time communication and performance categories. Finally, for databases, MySQL performed much faster than PostgreSQL in the average times for all categories.

This research could be easily extended by testing more operating systems, server frameworks, and databases with the same criteria. Additional criteria could also be added, such as the ability to build libraries for the servers and error times for the database. Some criteria could also be expanded upon, such as the performance criterion for servers. NodeJS did much better than Django in performance, but it cannot be readily seen in the results. Finally, an additional category could be added, such as server hosts or client-side frameworks.

References

[1] United States Department of Labor. Minimum Wage. https://www.dol.gov/general/topic/wages/minimumwage. Last accessed: March 16, 2018

[2] Statistic Brain Research Institute. 2017. Average Historic Price of RAM. https://www.statisticbrain.com/average-historic-price-of-ram/ Last accessed: March 16, 2018.

[3] Statistic Brain Research Institute. 2016. Average Cost of Hard Drive Storage. (September 2016). https://www.statisticbrain.com/average-cost-of-hard-drive-storage/ Last accessed: March 16, 2018.

[4] Microsoft. Windows 10 Pro. https://www.microsoft.com/en-us/store/d/windows-10-pro/df77x4d43rkt/48DN. Last accessed: March 16, 2018.

[5] Ubuntu. 2017. Installation/System Requirements. (October 2017). https://help.ubuntu.com/community/Installation/SystemRequirements Last accessed: March 16, 2018.