Islamic University of Technology (IUT)

Performance Improvement of a Front-end only Web Application

Authors

Shah Eftakher Sazid, 170042008

Saad Bin Johir, 170042047

Tasnim Jarin Afra, 170042073

Supervisor

Shohel Ahmed

Assistant Professor, Department of CSE

A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of B.Sc. in Software Engineering

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary organ of the Organization of Islamic Cooperation (OIC)

Academic Year: 2020-2021

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Shah Eftakher Sazid, Saad Bin Johir and Tasnim Jarin Afra under the supervision of Shohel Ahmed, Assistant Professor, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of reference is given.

[illegible]

Shah Eftakher Sazid
Student ID: 170042008
B.Sc in Software Engineering
Department of Computer Science and Engineering(CSE),
Islamic University of Technology (IUT)

Saad

Saad Bin Johir
Student ID: 170042047
B.Sc in Software Engineering
Department of Computer Science and Engineering(CSE),
Islamic University of Technology (IUT)

[illegible]

Tasnim Jarin Afra
Student ID: 170042073,
B.Sc in Software Engineering
Department of Computer Science and Engineering(CSE),
Islamic University of Technology (IUT)

Performance Improvement of a Front-end only Web Application

Approved by:

Supervisor:

[illegible]

Shohel Ahmed
Assistant Professor
Department of Computer Science and Engineering (CSE),
Islamic University of Technology (IUT), Gazipur.

Acknowledgment

This paper and the research behind it would not have been possible without the exceptional support of our supervisor, Shohel Ahmed, Assistant Professor, Department of Computer Science and Engineering, IUT. His enthusiasm, knowledge, and exacting attention to detail have been an inspiration and kept our work on track from the ideation of our project until the end. It motivated us to test and implement our work on our own built project. His valuable opinion, time, and feedback helped us to come this far and grow to what we bring here today. He helped us shape our thought process and focus on more important aspects of the research work and successfully guided us throughout this whole journey. It would be less to say that without his active participation this research would not reach its optimum level. We are honored to work with such a humble person whose industry expertise and eagerness to teach new things enriched our horizon and enlarged our domain of knowledge to present our work with such a standard.

Table of Contents

List of Figures

List of Tables

Abstract

In this fast-paced era where there are billions of active internet users worldwide, there is a huge demand for video streaming services, social networking sites, and communication using the internet. The Internet is being used for transactions, official email, or entertainment purposes. Users having short attention spans quickly move to different web pages if web pages lag by a few seconds, This results in loss of customers for e-commerce websites or loss of viewers in a video streaming platform. Many businesses are solely dependent on the internet for whom the loss of customers makes a huge difference due to slow websites. Hence, nowadays performance of websites is as important as the website itself. In this paper, we discussed how the performance of a website can be improved using performance enhancement techniques in the web application. There are many front-end only websites due to having Backend as a Service (BaaS) infrastructure and the ease of managing the backend using BaaS. We found there's a performance gap in such websites that results in users leaving the website. Our work is based on how to improve this performance to retain website traffic. We have studied how this performance gap can be reduced and explored some of the topics mentioned widely in papers. Therefore, we discussed various techniques which can be applied in a front-end only application to increase the performance. Here we implemented a few techniques such as caching, pseudo streaming, image optimization, etc. on a completed video streaming web application and performed testing on the modified application, and saw how much the changes affected the performance of the website. We used average response time, Error rate, Maximum response time and throughput as performance metrics to measure the performance and then analyzed the result with the previous performance to figure out which techniques created more impact on the overall improvement of performance. Performance of the application was improved up to 17.13% in response time and 51.76% in throughput on a modules using the selected performance techniques. Also some relation with content size and performance was found in this research.

Chapter 1

Introduction

Web performance shows how fast content is loaded and rendered and made visible to the user when requested. A user expects a page to load within one or two seconds, and three seconds is considered to be a slow website which results in a loss of trust and interest and an increase in bounce rate. The performance of an application has a strong correlation with the user experience. Website visitors retention and user satisfaction can be increased with a good performing website. This will in return increase the conversion and revenue generation from the website. The overall goal of improving performance is to attenuate the perceived delay the user experiences between the instant he clicks on the link and when the page is finally displayed.

Many a time it is seen that software developers give functional requirements of a system more priority and deliver to the customer without caring much about the non-functional requirements such as response time, throughput, scalability, etc. Non-functional requirements such as the performance of the website can make or break the success of the application. Closely observing through user behavior and contextual inquiry it has been noticed users are always more interested in website performance improvement rather than new features. Having a bad user experience in this sector can create a huge downfall in the popularity and acceptance of the platform. The problems related to performance are mainly slow video loading, slow content loading due to slow network issues, unable to access or load the desired file, and creating some lags. This led to our research project which is the performance improvement of a front-end only web application.

1.1 Problem Statement

In recent times, web pages are very complex and take a lot of memory. It is seen that user abandonment of websites increases significantly with an increase in response time [1]. Developers need to come across various ways of implementing performance-enhancing techniques that will result in less data consumption and fast page loading.

These days Backend as a Service(BaaS) is quite popular for the many benefits it provides. Using BaaS allows the developer to focus more on the front-end of the website allowing them to get more time working on the business needs rather than handling data and network-related issues. BaaS provides user authentication configuration and cloud storage which allows developers to not worry about backend infrastructure. Despite its many advantages and ease of focus from the developer's end, the performance of a BaaS is not up to the mark of its counterpart with a backend server-centric website. So we studied multiple papers and found various techniques that are widely popular in enhancing the performance of websites. We implemented a few techniques in a complete front-end only web application [2] and measured the impact it created on website performance.

1.2 Research Challenges

As a front-end only web application is comparatively costly for systems to handle and can seem somewhat inefficient not much direct study regarding this topic is available. So any direct performance improvement methods for front-end only systems were not available. We had to study the traditional improvement techniques for web applications with both a front-end and a dedicated backend. Then we tried to modify it for our system and implement it.

As front-end systems often directly rely on libraries availability of widely used libraries for performance improvement was not there. The libraries used are mostly created for a system with both front-end and backend. So we had to modify parts of our system to accommodate these libraries.

Another challenge we faced was the number of contents on our system. As the system was in a development environment the number of contents was limited. For testing purposes, we also disabled the authentication system, as it was inefficient for our study to test the performance of the authentication system.

1.3 Research outline

In this paper, we discussed our findings and the current state of our application. This paper covers few performance improvement techniques such as Web caching, Content Delivery Network, Reducing HTTP requests, Image optimization as well as pseudo streaming. The most widely used technique is caching where data is stored for reuse so when the user requests a resource, it can be fetched easily. This reduces server overload from traffic. This paper briefly covers the content delivery network(CDN) which helps in reducing load time and increasing responsiveness by using a group of servers and data centers. Later we discussed a few techniques to reduce HTTP requests, for example using removing unnecessary images, reducing the file size of images, etc. Pseudo-streaming is the concept of showing a video as it is being downloaded. This allows the user to wait for less for seeing the video and gives a sense of fast performing website. We studied further on a few techniques and implemented them on the tested website.

We have discussed our system architecture, methodology to carry out the experiments. We have shared the modules used in this research along with the tools used for testing. To determine the performance of a web application the common and most effective way is Performance testing. Performance testing can determine the current score of the performance metrics like response time, throughput, and error rate. Also through load testing, the maximum load handling capacity of web applications can be found. [3,4]

We have analyzed the performance of our system based on these performance metrics to find out the state of the system. Then through iterative process we have implemented the performance improvement techniques, re-tested and re-analyzed the results. This paper also includes the implementation details of the techniques used on our systems.

Finally we concluded our study after finding improvement in the system. After implementing the changes and testing for performance analysis revealed that some technique like caching are effective enough to improve the performance by 17.13% in response time and 51.76% in throughput. Then further correlation between content size and performance was discovered. Also consequence of a bad architecture has been discussed in this paper.

Chapter 2

Background Study and Literature Review

Throughout the evolution of web applications researchers have been trying to ensure better performance for a better user experience. Among the researched methods only some efficient techniques are popular. Performance improvement techniques like Web caching, Content Delivery Network, Reducing HTTP requests, and Image optimization are widely used. Also to ensure a smooth streaming experience pseudo streaming is used.

[5–7] discussed Content Delivery Network(CDN) as a possible way to improve web performance. Using CDN load on an actual web server can be reduced to perform better. [7, 8] discussed reducing HTTP requests as an effective way of improving the performance of a web application.

Techniques like Compressing textual content, reducing the size of graphical objects, reducing the number of DNS requests, and correctly formatting images in the right place have been suggested to the developers to reduce the load time and waiting time.

According to the result of [8] a combination of multiple techniques can reduce the response time and throughput by twice. To understand which techniques are improving the performance, testing plays a vital role. [3] suggests various testing techniques like stress testing, load testing, and virtual user load testing as good performance testing techniques. Systematic analysis of the result is important to find out the performance status of the application.

Progressive rendering to load the relevant part to the users only, compressing the multimedia contents like images on the pages can help improve the load time of the application [6, 9]. The concept of pseudo streaming, pre-fetching based on user interaction has been suggested to improve the user experience while interacting with the application [5, 8].

2.1 Web caching

Web caching has been identified as one of the most effective methods for relieving service bottlenecks and reducing network traffic, consequently reducing user access latency. As the web continues to expand at an exponential rate, pages climb by around 15% every month. In this paper, the research frontier in web caching has been discussed after an analysis of the properties and techniques used in web caching. Even though Internet backbone capacity grows at a rate of 60% per year, demand for bandwidth is projected to surpass supply for the foreseeable future as more and more information services migrate to the Web. If a solution to the difficulties posed by its fast expansion is not implemented, the WWW will become too crowded, and its whole attractiveness will finally be lost. To work on this caching popular items near clients have been identified as one of the efficient strategies to ease Web service bottlenecks, minimize Internet traffic, and improve the scalability of the WWW system [10]. When proxy servers [11] were initially used to facilitate access to the Internet from users behind a firewall, the notion of utilizing them to cache first appeared. Web caching at proxy servers can not only reduce network bandwidth but also minimize access latency for clients.

An application-level proxy allows users in an organization to securely pass through a firewall without creating a possible security hole through which infiltrators might enter the business's network. A proxy allows client writers to ignore the tens of thousands of lines of networking code required to support each protocol and focus on more important client issues. It is possible to have lighter clients that only understand HTTP. Other protocols are transparently handled by the proxy. No protocol capability is lost by utilizing HTTP between the client and proxy since other protocols translate nicely into HTTP methods. [11]

So far, the majority of research works on cache replacement algorithm design have concentrated on a single cache, [12–14]. When caches are placed in a hierarchical structure, however, using a cache replacement technique intended for an isolated cache may not result in overall high performance.

So basically the technique of keeping copies of files in a cache, or temporary storage area, so that they can be retrieved more rapidly. To speed up website loading, web browsers cache HTML files, JavaScript, and pictures, while DNS servers cache DNS records for faster lookups and CDN servers cache material to reduce latency. [15, 16] Web caching can improve the latency by 26% of any web application [17]

2.2 Prefetching

Web page prefetching is a popular research topic that has sparked a lot of attention in recent years. This a predictive approach. It is an effective solution for reducing user latency [17]. Prefetching obtains some web items in advance of user requests. Thus reducing user-perceived latency. In a lot of scenarios prefetching is used combined with web caching to get better results.

Users often navigate the web by following hyperlinks from one web page to the next. Hyperlinks on a website frequently refer to sites on the same server. After each page is loaded, there is usually a gap while the user reads the displayed material. The client might utilize this time to prefetch files that are anticipated to be retrieved soon, avoiding retrieval delay if and when such files are requested. The retrieval latency has not been lowered; it has just been overlapped with the time the user spends reading, resulting in a shorter access time. The server calculates the chance of a specific web page being viewed next and communicates this information to the client. The client program then determines whether or not to execute the command and gets the page in advance. This division of labor between the server and the client is very normal. The ability to notice the pattern of accesses from several clients can utilize this knowledge to make decisions. The customer, on the other hand, is in the greatest position to determine whether or not to prefetch files based on whether they are previously cached or are cost effective (in terms of CPU time, memory, and network bandwidth). A balance must be achieved between the increased traffic and the improved access time. Predictive prefetching of Web data can reduce perceived latency significantly, at the cost of increased network traffic. [16]

2.3 Image Optimization

Multimedia contents of a web page like Images is a large part of the design elements. It is often found that these multimedia content can cause performance issues to the system. To resolve this optimized multimedia content is used. So optimized formats like png and jpeg in terms of image can significantly reduce the load time of the website.

Using image optimization the web sites load can be improved. The multimedia content load time is a major factor in the load time of a web application. So optimized formats like png and jpeg in terms of an image can reduce the load time of the website [6, 7].

Some suggestions have been given related to image optimization like- Use CSS to generate an image, formally format and compress images, use of CSS sprites i.e. putting multiple background images into one composite image and displaying using

CSS background-position [1]. Moreover, Image optimization suggests that JGEP format for normal pictures, PNG for the logo. Avoiding fromats like BMP, TIFF have been suggested. Keeping the number of images low in the application is a good practice as it can improve load time [6]. These are some of the implications to optimize images to gain web performance that has been suggested in these mentioned papers and we have implemented our research on basis of those.

2.4 Load Balancing

Load balancing is the process of dispersing incoming network traffic among a set of servers, commonly referred to as a server pool. Load balancers disperse the load without the client knowing so there is no effect on load time or response time. Rather the application can serve the need of more users with load balancers active in the backend.

Tasks are performed by load balancers are client requests and network traffic are efficiently dispersed over multiple servers, ensuring excellent availability and reliability by sending requests exclusively to online servers, allowing developers to add or delete servers based on your need [18].

Cloud computing has grown in popularity in recent years. It offers a versatile and simple service as part of its offerings. A method of storing and retrieving data and files, particularly for creating huge data sets and files is available for a growing number of users. For users throughout the globe handling such huge data sets necessitates several approaches for optimizing and streamlining processes and offering users adequate levels of performance. One critical challenge in this industry is dynamic load balancing or job scheduling. Load balancing techniques have been extensively researched in a variety of situations; nevertheless, certain extra issues exist in Cloud systems and must be addressed. The major problems in Cloud Computing include properly distributing jobs to Cloud nodes so that effort and request processing is as efficient as possible [19] while being able to bear different impacting restrictions such as heterogeneity and excessive communication latency. A comprehensive replication algorithm does not consider efficient storage consumption. This is because the identical data will be saved in all replication nodes. Full replication techniques are more expensive since more storage is required. Partial replication techniques, on the other hand, might save sections of the data sets in each node with some overlap dependent on each node's characteristics such as processing power and capacity [20]. This may result in improved usage, but also increases the complexity of the load balancing algorithms as they seek to account for the availability of the data set's portions across the many Cloud nodes.

Controlling load balancing and collecting data about the different nodes must be built in such a manner that the algorithm does not have a single point of failure. Some algorithms like centralized algorithms can provide efficient and effective methods for tackling load balancing problems in a specific pattern. They however have the problem of just having one controller for the entire system. In such instances, if the controller fails, the entire system fails. To address this difficulty, any load balancing algorithm must be constructed [21]. Although distributed load balancing algorithms appear to be a superior option, they are far more complicated and require more coordination and management to perform properly.

Load balancing helps networks and resources by delivering maximum throughput with the shortest possible response time [22]. Data may be transferred and received without significant delay by dividing traffic amongst servers. There are several techniques available to assist load traffic between accessible servers. Websites are a simple illustration of load balancing in our daily lives. Users may face delays, timeouts, and even delayed system replies if load balancing is not used. Load balancing systems often employ redundant servers, which aid in the better allocation of communication traffic, ensuring that website availability is established [21] [23].

Load balancing [24] aims to achieve the following- improve performance, maintain system stability, provide a fault tolerance mechanism and accommodate future modifications. [25]

2.5 Pseudo Streaming

Pseudo streaming is a streaming technique where the content will be streamed as it is being downloaded. Unlike traditional streaming where the whole content is to be downloaded before it can be streamed. So pseudo steaming can reduce the waiting time of the users.

The Content Delivery Network(CDN) can be used to avail the benefit of pseudo streaming. CDN static and static push services can be used for pseudo streaming. Serving multimedia material over the Internet with minimal latency is still a difficulty. With the introduction of Web 2.0, a variety of video-sharing services utilizing various storage and content delivery mechanisms have gained popularity. However, nothing is known about these models on a worldwide scale. Such comprehension is necessary for creating systems that can efficiently provide video material to people all around the world. The new services make use of a video distribution technology called pseudo-streaming [26]. This is distinct from standard web design. The video material can be played back as it is streamed and is being downloaded in phases. Unlike traditional materials, HTTP/TCP is used to deliver streaming through generic web servers.

Although there have been many studies on conventional and live online streaming [27, 28], as well as their respective workloads [29, 30], as well as content distribution via content delivery. There is a need for networks [31, 32] or peer-assisted techniques [33,34]. There is minimal information available about the most current pseudo-streaming services. Only a few recent studies have looked at how user-generated material affects search results is seen and shared on the Internet [26, 35–37]. These studies, on the other hand, either passively monitor pseudo streaming multimedia traffic to describe media access patterns or evaluate the spread of user-generated content, available on these websites by crawling them. [5]

Chapter 3

Methodology

3.1 System Architecture

Our current system architecture consists of 4 modules, 1. Video module, 2. Forum module, 3. User profile module and 4. Authentication module. Each module can operate independently. Figure 3.1 shows the system architecture of our Gameroom [2] web application.

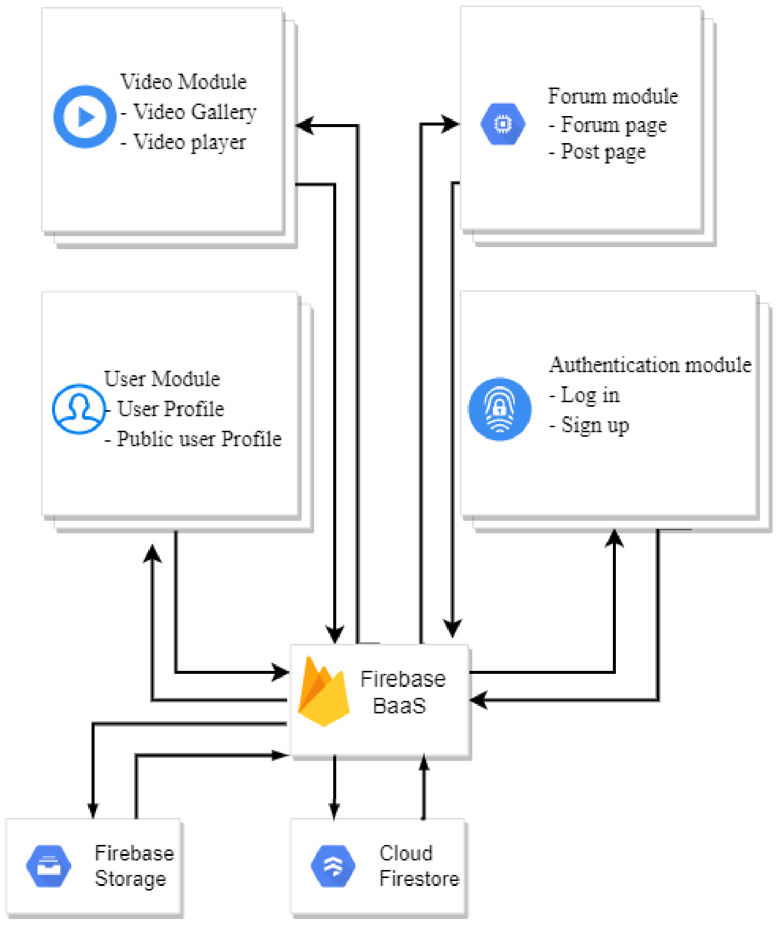


Figure 3.1: System architecture of Gameroom

The video module contains the video gallery where all the video available in the app is displayed, the video player to play a video, and user interaction with the video i.e. likes and comments. The forum module holds the forum posts and each of their user interactions, for example: comments. The user profile module holds all the user information and their uploaded contents.

Whenever any of these modules are visited these modules send one or more HTTP requests to the backend for the data from Firestore a NoSQL database. Upon receiving the data the module processes the data and displays them for the user. The multimedia contents are stored in the Firebase storage. The modules can access the storage through Firebase.

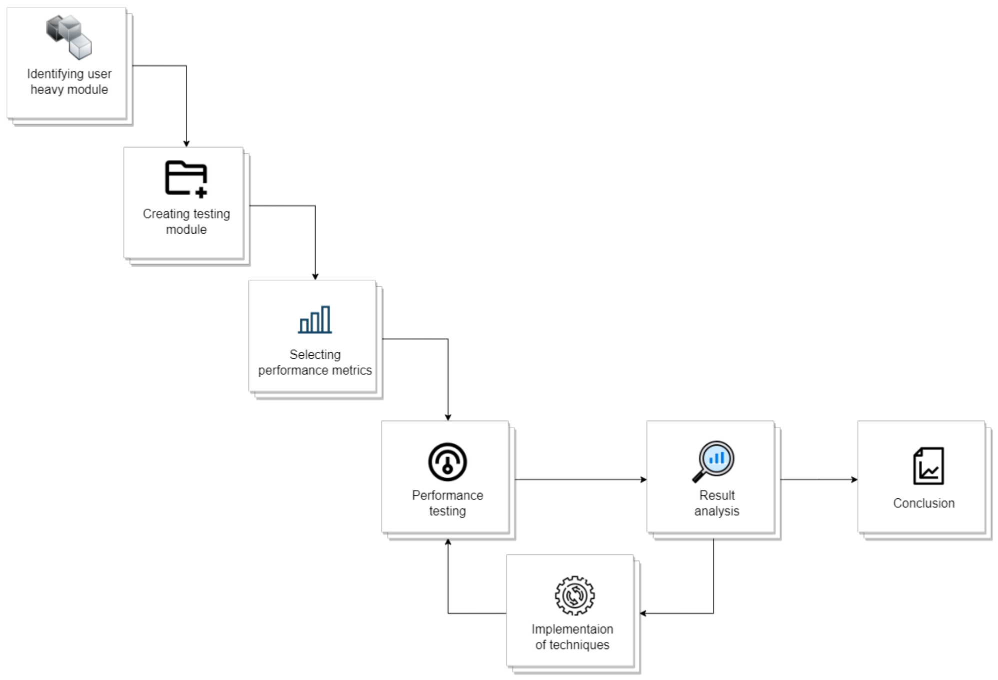

Identifying user heavy module
Creating testing module
Selecting performance metrics
Performance testing
Result analysis
Conclusion
Implementaion of techniques

Figure 3.2: Research methodology

Figure 3.2 describes our methodology. At first, we identified the modules that have maximum user interaction and need to fetch data more often. We found out that the Video module and the forum module have the most user interaction and most data fetching, followed by the user profile module. Based on this information we created two testing modules. Then we identified some testing metrics and selected suitable metrics for our analysis. Then we performed performance testing and analyzed the results. After that we implemented our selected performance improvement techniques and again tested the updated module and analyzed the results. Based on the results if there is any scope of improvement we went back to implementation phase and implement it. Then we re-tested and re-analyzed the modules. When there were no scope of improvement we concluded the experiment.

3.2 Testing modules

We designed two testing modules containing most of the user-heavy routes from all the modules. Our study of the Gameroom system led us to believe that the Homepage, Video Gallery page, Video player page, Forum page, and Forum post page will have the highest amount of traffic. So we created a Video testing module consisting of the video routes Figure 3.3 and a Forum testing module containing the forum routes Figure 3.4.

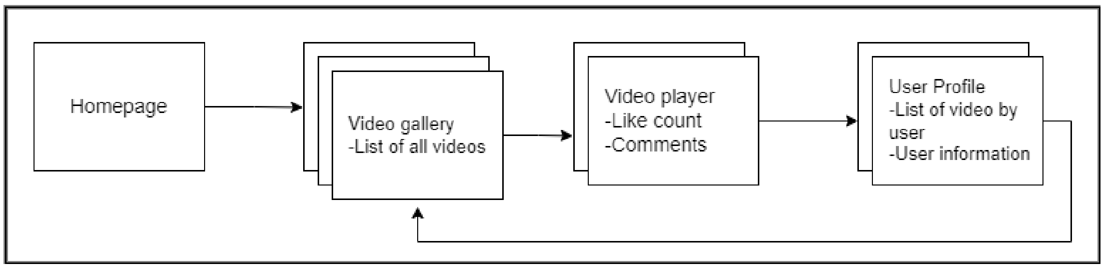


Figure 3.3: Video Testing module

The video module consists of routes: Homepage, Video Gallery page, Video player page along with User Profile Page. The forum module contains a homepage, a Forum page, a Forum post page, and a user profile page. The flow of interaction is according to the Figure:

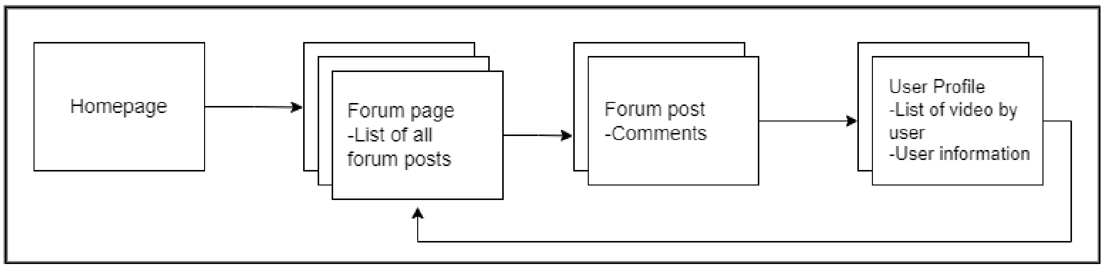


Figure 3.4: Forum testing module

We have monitored the performance of these modules by analyzing the performance metrics using Load testing.

3.3 Performance Metrics

To analyze the performance of the web application we needed some performance metrics. Performance metrics are the quantitative measure of the quality of the system. While selecting performance metrics the attributes we are trying to analyze must be considered. We selected Average response time, Throughput [6] and Error rate as we want to improve the overall performance of the application.

Average response time is the average time the user has to wait to get the content to the browser. This is primarily due to the time required to fetch data from the server. This response time is also affected by the load on the server. The higher the load on the server the higher the response time gets. Throughput is measured in units delivered per second. This can be request handled per second, the page loaded per second. In our case we considered throughput to be requested/second. [38] The error rate is the performance of requests that failed. This error usually occurs when the request handling capacity of the server exceeds.

3.4 Testing method

We selected Load testing as our performance testing approach. Load testing is done by gradually increasing the load on the system to monitor the system performance. With the change of load, various performance indicators also change. We used virtual users to perform load testing. The virtual user load simulates the real user interaction on the system. In our case, a series of routes will be visited by a virtual user and the system performance will be monitored. Figure 3.5 shows our testing approach for each iteration.

With each implementation of the performance improvement technique, we tested the new state with the whole testing suite to get the current performance state and analyze improvements. In this stage the performance metric scores are collected for analysis.

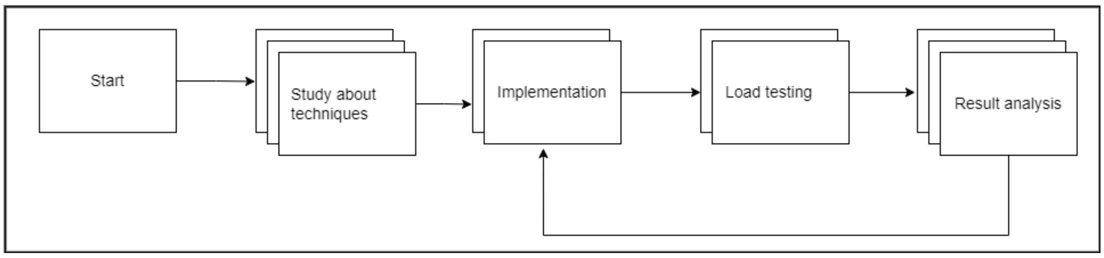


Figure 3.5: Testing methodology

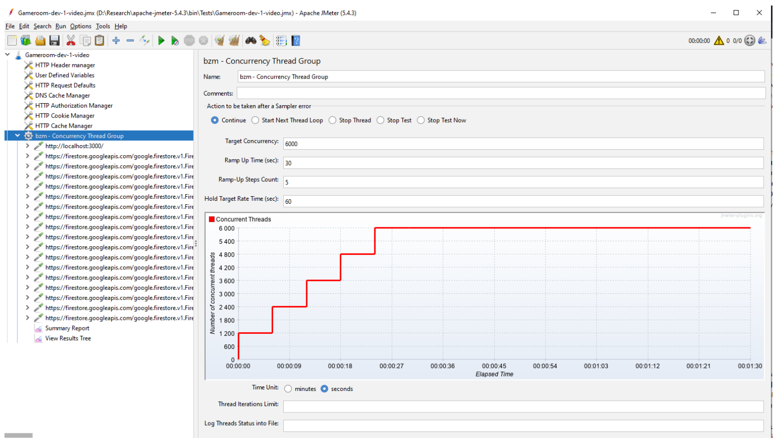
Chapter 4

Implementation and Experimental Setup

We have implemented the performance enhancements and tested our video streaming application Gameroom [2] in a development environment. This application is created using ReactJS [39] web development framework and for the backend, we have used Firebase BaaS and Firestore as a NoSQL Database. To store our multimedia contents we have used Firebase storage. The test scripts have been created using an automated recording tool and performance testing has been done in JMeter [40].

4.1 Test setup

The test bed used for testing was an intel i7-4700, 24GB ram. The test script was created using Blazemeter [41]. This is an automated tool for test case creation for JMeter. The script contained the test routes and the configurations. To test the modules JMeter [40] was used as a performance testing tool. Load testing was performed on the scripts to find out the performance of the module. Concurrent loads of 100, 500, 1000, 2000, 3000, 4000, 5000, 6000 virtual users were used with concurrency thread group, Figure 4.1.



Gameroom-dev-1-video.jmx (D:\Research\apache-jmeter-5.4.3\bin\Tests\Gameroom-dev-1-video.jmx) - Apache JMeter (5.4.3)

- Gameroom-dev-1-video
  - HTTP Header manager
  - User Defined Variables
  - HTTP Request Defaults
  - DNS Cache Manager
  - HTTP Authorization Manager
  - HTTP Cookie Manager
  - HTTP Cache Manager
  - bzm - Concurrency Thread Group
    - http://localhost:3000/
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - https://firestore.googleapis.com/google.firestore.v1.Fire
    - Summary Report
    - View Results Tree

bzm - Concurrency Thread Group

Name: bzm - Concurrency Thread Group

Comments:

Action to be taken after a Sampler error: [✓ Continue] Start Next Thread Loop / Stop Thread / Stop Test / Stop Test Now

Target Concurrency: 6000

Ramp Up Time (sec): 30

Ramp-Up Steps Count: 5

Hold Target Rate Time (sec): 60

Time Unit: minutes [✓ seconds]

Thread Iterations Limit:

Log Threads Status into File:

Figure 4.1: JMeter test suite for 6000 virtual users

4.2 Caching

Caching has been implemented using a library called swr-firestore-v9 [42]. This library is specifically created to avail the benefit of another caching library SWR [43]. SWR is a React [39] hook library for data fetching. SWR allows the application to automatically cache the data from the server. When the user requests the same data it checks the cache to check if the data is available. If the data is available then it does not request the data from the server instead it takes the data from the cache and displays it. SWR-firestore-v9 is built upon SWR functionality to integrate firestore query with caching. When a database query is sent the response is deduped, cached, and shared automatically. As the request uses the same SWR key there is no need for multiple requests. Also, this library automatically re-evaluates the contents whenever a page comes into focus. Using this library we cached the video data on our Homepage, Gallery page, Userprofile page, and the Forum page, Figure 4.2. Then the modules were tested for performance.

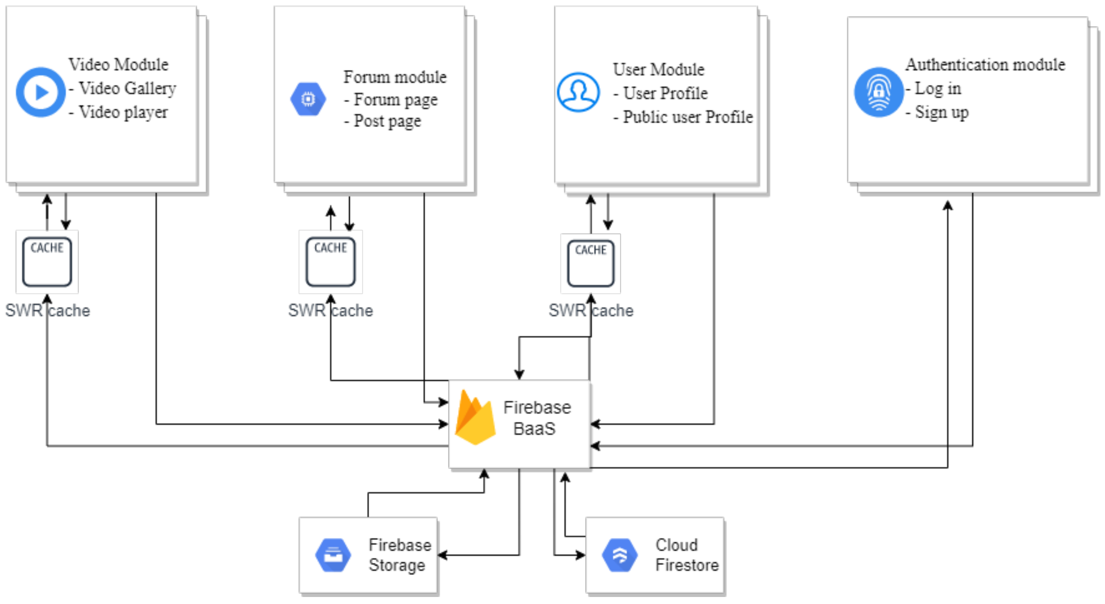


Figure 4.2: Gameroom with caching

4.3 Image optimization

To implement image optimization we tested with different image formats of different sizes. We stored 3 images of different sizes and formats in Firebase storage. Then we updated our image contents in the database, Firestore. As the application receives all the multimedia contents from firestore, the application will receive the updated images, Figure 4.3. The image selected was a PNG image of size: 1.3MB, a PNG image with 500KB, and a jpeg image with 47KB. Then we tested for performance on this version of the application.

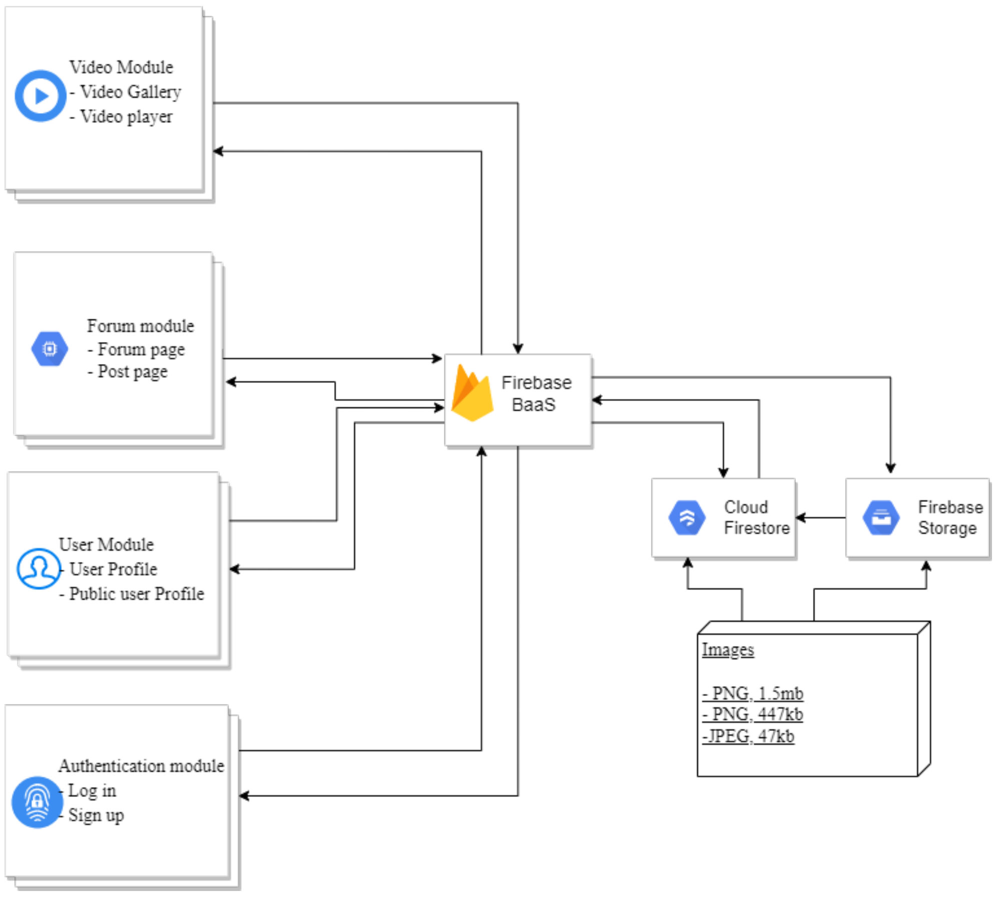


Figure 4.3: Gameroom with image optimization

Chapter 5

Result analysis

The performance measures have been directly derived from the JMeter Summary report. This report contains our desired performance metrics Average Response time, Error rate, Throughput, and also some other measures like Maximum, Minimum response time, etc. First, we extracted our desired metrics from these reports. Then we analyzed these metrics by comparing with the baseline metrics that we found during the initial study of the system. Figure 5.1 shows the initial performance of the modules of this system.

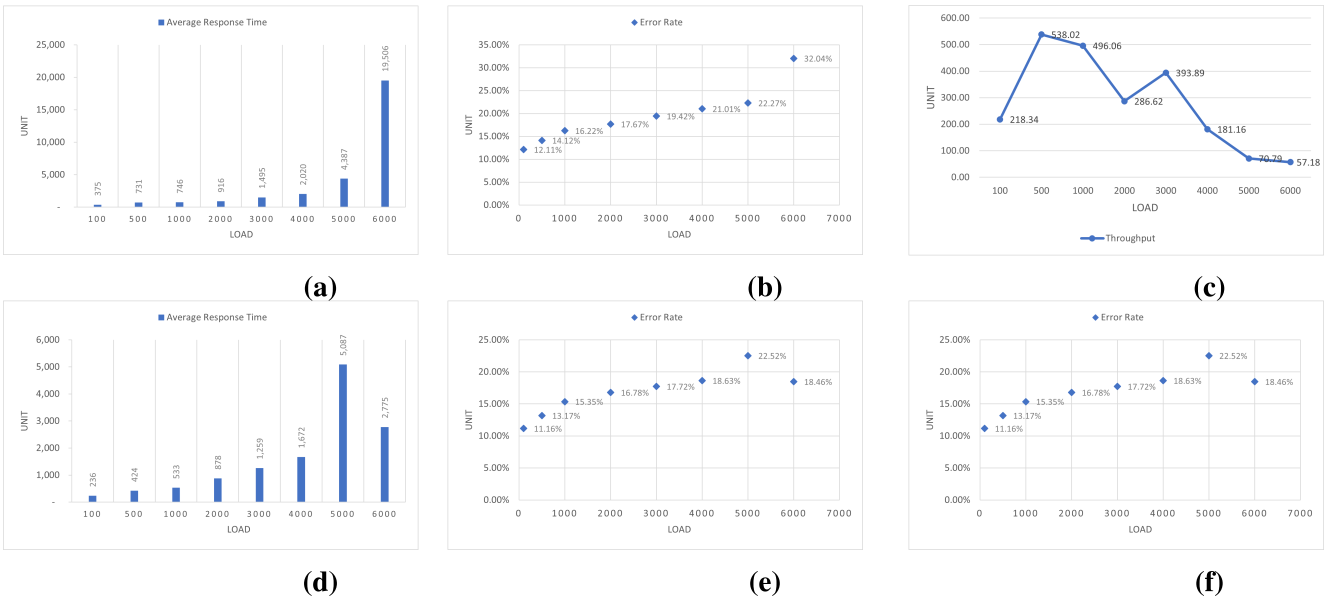


Figure 5.1: Initial performance measure of Gameroom 1.Forum module- (a) ART, (b) Error rate, (c) Throughput, 2. Video module -(d) ART, (e) Error rate, (f) Throughput.

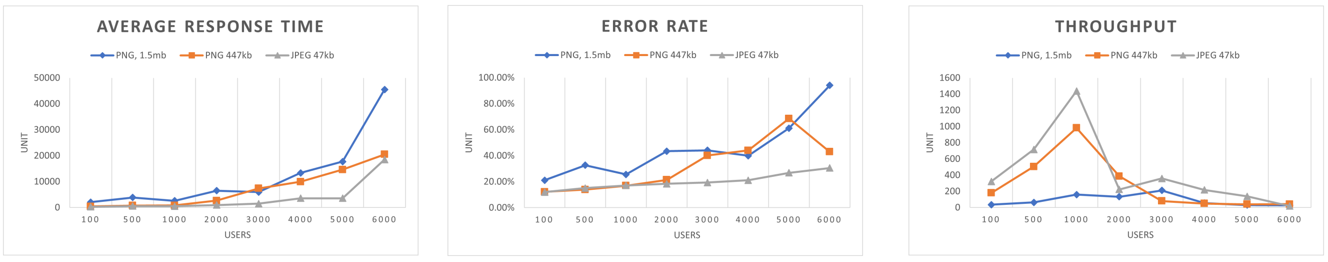
5.1 Image Optimization

Implementation of Image optimization was done by uploading 3 different Images to the storage and linking them to the database. Then the module was tested for performance. The test results showed some interesting insights. The table 5.1 shows the average response time, average error rate and average throughput for 3 types of image file. We can see from the test results that the size of the image file has a direct impact on the performance of the application. For both testing modules the findings were similar. With the increase of size the performance decreases.

Table 5.1: Performance metrics with different format and size of images; ATR: Average response time, AT: Average Throughput, AER: Average Error rate

		Forum Testing module			Video Testing module		
Format	Size	ART	AT	AER	ART	AT	AER
PNG	1.5 mb	5512.5	2258.079	45.28%	5512.5	2258.079	18.11%
PNG	447kb	7132	283.26775	32.51%	5184.875	479.73	19.26%
JPEG	47kb	3683.625	428.8593325	20.05%	3089.875	314.82	17.75%

In terms of modules, the improvement in the Forum module is more significant. The graphs in Figure 5.2 show the performance metrics with three different image files for the forum module. From this we have concluded that JPEG format with size 47kb is the most efficient image format.



(a) (b) (c)

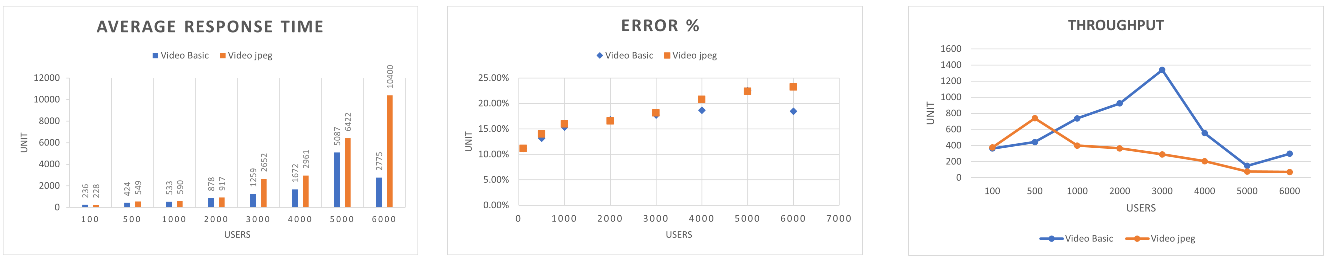
Figure 5.2: Image Test data for forum module-(a) ART, (b) Error rate, (c) Throughput.

Then comparing the result with baseline we found out that the average response time was improved by 2%, error rate remained the same and throughput increased by 53% for the forum module, Table 5.2. Another noteworthy improvement here is that with less load the performance improvement was significant.

Table 5.2: Improvement in % for Image Optimization in Forum Module

Improvement for Image Optimization		
Virtual User Load	Average Response Time	Throughput
100	32%	47%
500	25%	33%
1000	28%	190%
2000	N/A	N/A
3000	N/A	N/A
4000	N/A	20%
5000	19%	95%
6000	5%	N/A
Average	2%	53%

For video testing module there were no significant improvement. For this module the approach was similar. We found JPEG as the most efficient image format and compares with baseline. But the comparison revealed that performance measure was similar to forum module for smaller load, but as the load increased the system failed to perform, Figure: 5.3.



(a) (b) (c)

Figure 5.3: Image Test data for video module(a) ART, (b) Error rate, (c) Throughput.

The possible reason behind video module not improving with higher load is that the video module additionally has to handle video data. As the video contents of this system are not optimized enough so it takes more time to load. Thus increases throughput. So overall performance improvement using only Image optimization was not enough here.

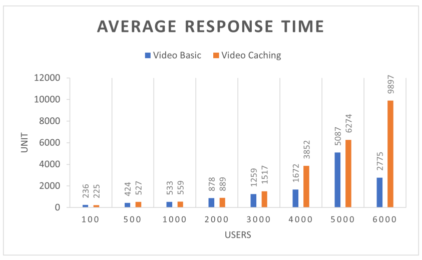
5.2 Caching

Caching was implemented using swr. The test results showed some improvement in both the test modules. The forum module had a significant improvement in throughput and average response time. The system performs better with both smaller and higher load. According to test results of forum testing module Table-5.3 shows that the throughput was improved by 51.76% and the average response time was improved by 17.13%.

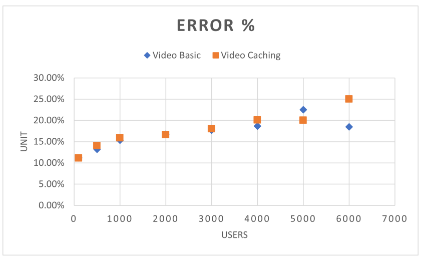
Table 5.3: Improvement in % for Caching in Forum Module

Improvement for Caching		
Virtual User Load	Average Response Time	Throughput
100	41.07%	69.60%
500	19.70%	24.69%
1000	27.35%	186.07%
2000	N/A	N/A
3000	2.61%	N/A
4000	N/A	67.34%
5000	N/A	N/A
6000	39.74%	119.80%
Average	17.13%	51.76%

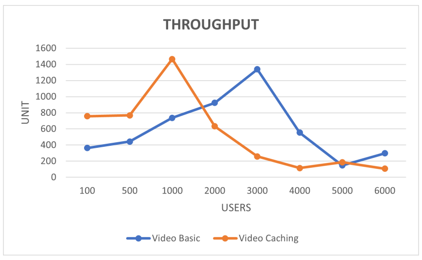
In case of video testing module, even with caching implemented, improvement is low. The improvement with less load is good enough but as the load increased the system started lagging in terms of performance. This is due to the additional video loading done by the video module. As the video data is not optimized in this application whenever the module visits a particular video in the video player the extra load decreases the performance. The performance data comparing to the initial system is shown in Figure 5.4



(a)



(b)



(c)

Figure 5.4: Caching Test data for video module(a) ART, (b) Error rate, (c) Throughput.

Chapter 6

Conclusion and Future Work

The performance of a web application is a non-functional requirement that is often ignored while designing the system. Creating an architecture without considering the performance can make it hard to improve performance later on. Our system faced a similar issue here. We implemented the improvements even after that the performance improvement is a maximum of 17.13% for average response time for caching in the forum module and 53% in throughput which is also in the forum module but for the optimized image. The video testing module did not have any significant improvement at all. This is due to the fact the system has a lot of dependencies that could not be addressed during our implementation. The firebase implementation before implementing the swr-firebase-v9 library made our implemented caching lag in terms of performance. Also, the video contents were not optimized, so it caused additional performance issues in the case of the Video testing module. Another shortcoming of our research is that we did not take into account the processing power and the memory space taken by the app during the execution in the client system.

In the future, the overall architecture needs to be changed to enable performance enhancement without side effects. Also, improvement techniques like Prefetching with machine learning models can improve the system performance. Load balancing is a well-known method that can be enabled to improve performance as well. Also while improving the application the resource requirement should be considered. Overall for a front-end only web application there are not many direct improvement techniques but the techniques available for traditional web apps can be modified to be used in front-end only web applications.

References

[1] N. SyamimiSaid, H. Shaker, M. Abdelhaq, O. Alsaqour, and M. Uddin, “Review on web performance,” *Journal of Engineering and Applied Sciences*, vol. 9, pp. 18–23, 01 2014.

[2] “Gameroom,” Jan 2022. [Online]. Available: https://gameroom-93fa3.web.app/

[3] K. Zhu, J. Fu, and Y. Li, “Research the performance testing and performance improvement strategy in web application,” in *2010 2nd International Conference on Education Technology and Computer*, vol. 2, 2010, pp. V2–328–V2–332.

[4] “Software performance testing metrics: What are metrics and how to use them?” Oct 2020. [Online]. Available: https://u-tor.com/topic/performance-testing-metrics

[5] M. Saxena, U. Sharan, and S. Fahmy, “Analyzing video services in web 2.0: A global perspective,” in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 39–44. [Online]. Available: https://doi.org/10.1145/1496046.1496056

[6] A. Garg, “Analysis of various techniques for improving web performance,” *IJARCSMS*, vol. 3, pp. 271–279, 04 2015.

[7] B. Subrayen, G. Elangovan, V. Muthusamy, and A. Anantharajan, “A case study for improving the performance of web application,” *International Journal of Web Technology*, vol. 2, no. 01, pp. 17–20, 2013.

[8] I. Jugo, D. Kermek, and A. Meštrović, “Analysis and evaluation of web application performance enhancement techniques,” in *Web Engineering*, S. Casteleyn, G. Rossi, and M. Winckler, Eds. Cham: Springer International Publishing, 2014, pp. 40–56.

[9] D. Manhas, “A study of factors affecting websites page loading speed for efficient web performance,” *International Journal of Computer Sciences and Engineering*, 12 2013.

[10] J. Wang, “A survey of web caching schemes for the internet,” *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, p. 36–46, oct 1999. [Online]. Available: https://doi.org/10.1145/505696.505701

[11] A. Luotonen and K. Altis, “World-wide web proxies,” *Computer Networks and ISDN Systems*, vol. 27, no. 2, pp. 147–154, 1994, selected Papers of the First World-Wide Web Conference. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0169755294901287

[12] J. Zhang, R. Izmailov, D. Reininger, and M. Ott, “Web caching framework: analytical models and beyond,” in *Proceedings 1999 IEEE Workshop on Internet Applications (Cat. No.PR00197)*, 1999, pp. 132–141.

[13] A. Belloum and B. Hertzberger, “Dealing with one-timer-documents in web caching,” 09 1998, pp. 544–550 vol.2.

[14] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, “Caching proxies: Limitations and potentials,” 1995.

[15] W. Ali, S. M. Shamsuddin, A. S. Ismail *et al.*, “A survey of web caching and prefetching,” *Int. J. Advance. Soft Comput. Appl*, vol. 3, no. 1, pp. 18–44, 2011.

[16] “what is caching?” Jan 2022. [Online]. Available: https://www.cloudflare.com/learning/cdn/what-is-caching/

[17] W. Ali, S. M. Shamsuddin, and A. S. Ismail, “A survey of web caching and prefetching a survey of web caching and prefetching,” *International Journal of Advances in Soft Computing and its Applications*, vol. 3, 03 2011.

[18] “What is load balancing? how load balancers work,” Jan 2022. [Online]. Available: https://www.nginx.com/resources/glossary/load-balancing/

[19] M. Randles, D. Lamb, and A. Taleb-Bendiab, “A comparative study into distributed load balancing algorithms for cloud computing,” in *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, 2010, pp. 551–556.

[20] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” in *2008 grid computing environments workshop*. Ieee, 2008, pp. 1–10.

[21] R. Ranjan, L. Zhao, X. Wu, A. Liu, A. Quiroz, and M. Parashar, *Peer-to-Peer Cloud Provisioning: Service Discovery and Load-Balancing*, 05 2010, pp. 195–217.

[22] R. J. Shimonski, *Windows Server 2003 Clustering & Load Balancing*. McGraw-Hill, Inc., 2003.

[23] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, “A survey of load balancing in cloud computing: Challenges and algorithms,” in *2012 Second Symposium on Network Cloud Computing and Applications*, 2012, pp. 137–142.

[24] D. Escalnte and A. J. Korty, “Cloud services: policy and assessment,” *Educause Review*, vol. 46, no. 4, pp. 60–61, 2011.

[25] T. Desai and J. Prajapati, “A survey of various load balancing techniques and challenges in cloud computing,” *International Journal of Scientific & Technology Research*, vol. 2, no. 11, pp. 158–161, 2013.

[26] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “Youtube traffic characterization: a view from the edge,” in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 15–28.

[27] Y. Wang, M. Claypool, and Z. Zuo, “An empirical study of realvideo performance across the internet,” in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, pp. 295–309.

[28] K. Sripanidkulchai, B. Maggs, and H. Zhang, “An analysis of live streaming workloads on the internet,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 41–54.

[29] M. Arlitt and T. Jin, “A workload characterization study of the 1998 world cup web site,” *IEEE Network*, vol. 14, no. 3, pp. 30–37, 2000.

[30] M. Arlitt and C. Williamson, “Internet web servers: workload characterization and performance implications,” *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 631–645, 1997.

[31] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, “An analysis of internet content delivery systems,” *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 315–327, 2002.

[32] B. Krishnamurthy, C. Wills, and Y. Zhang, “On the use and performance of content distribution networks,” in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, pp. 169–182.

[33] B. Cheng, X. Liu, Z. Zhang, and H. Jin, “A measurement study of a peer-to-peer video-on-demand system.” in *IPTPS*. Citeseer, 2007.

[34] Y.-F. Chen, Y. Huang, R. Jana, H. Jiang, M. Rabinovich, B. Wei, and Z. Xiao, “When is p2p technology beneficial for iptv services,” in *Proceedings of the 17th International Workshop on Network and Operating System Support for Digital Audio and Video*, 2007, pp. 04–05.

[35] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, “I tube, you tube, everybody tubes: analyzing the world’s largest user generated content video system,” in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 1–14.

[36] X. Cheng, J. Liu, and C. Dale, “Understanding the characteristics of internet short video sharing: A youtube-based measurement study,” *IEEE transactions on multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.

[37] M. Zink, K. Suh, Y. Gu, and J. Kurose, “Watch global, cache local: Youtube network traffic at a campus network: measurements and implications,” in *Multimedia Computing and Networking 2008*, vol. 6818. SPIE, 2008, pp. 35–47.

[38] “Glossary,” Oct 2020. [Online]. Available: https://jmeter.apache.org/usermanual/glossary.html

[39] Meta Platforms, Inc, “Reactjs.” [Online]. Available: https://reactjs.org/

[40] Apache Software Foundation, “Jmeter.” [Online]. Available: https://jmeter.apache.org/

[41] Perforce Software, Inc, “Blazemeter.” [Online]. Available: https://www.blazemeter.com/

[42] lemasc, “swr-firestore-v9.” [Online]. Available: https://github.com/lemasc/swr-firestore

[43] Vercel, “swr.” [Online]. Available: https://swr.vercel.app/