Image Upload and Download speed Comparison Among MySql(local), Oracle(LAN) and Firebase(Cloud)

Balu Tikkisetti
Advisor: Dr. Gang Qian

COLLEGE OF Mathematics and Science
UNIVERSITY OF CENTRAL OKLAHOMA

UNIVERSITY OF CENTRAL OKLAHOMA
Center for Research and Education in Interdisciplinary Computation

COMPUTER SCIENCE

1 ABSTRACT

Our project "College Media" underwent rigorous performance testing to gauge image uploading and downloading efficiency across MySQL, Oracle, and Firebase database models. Employing Spring Boot 3.2.3 for backend services, we ensured seamless integration and smooth operation. For frontend development, React (v18.2.0) coupled with the Vite framework provided a responsive user interface. During testing, local MySQL (v8.3.0) and Cloud Firebase (v10.11.1) instances were utilized, while Oracle was accessed via PHP hosted on our LAN (cs2.uco.edu). While Spring Boot primarily interacts with MySQL for storing user details, comments, and hashtags, image storage was segregated into a dedicated MySQL table using the BLOB datatype. Firebase, on the other hand, leverages its storage capabilities for efficient image handling. For Oracle, a distinctive approach was adopted, storing images as CLOB through base64 encoding conversion. Despite divergent storage methods, our project seamlessly integrates these databases for effective image management.

Beyond basic image handling, our system boasts advanced functionalities such as hashtag-based and keyword-based search algorithms, enhancing user experience. Users can effortlessly navigate through posts by clicking on relevant hashtags, while system-generated posts labeled "#generatedBySystem" are also accessible. Additionally, the system supports text-to-image conversion for posts, augmenting content creation versatility. These sophisticated features are orchestrated through meticulously designed Spring Boot controllers, ensuring robust performance and seamless user interaction throughout the "College Media" platform.

2 PROJECT OVERVIEW

In our college media platform project, users can sign up as students, staff, or alumni, gaining access to a dynamic space for sharing experiences and connecting with others. Upon registration, users can post images along with hashtags and descriptions, facilitating diverse forms of expression and storytelling. The platform's subscription feature enables users to follow others, receiving their posts in reverse chronological order and fostering a sense of community and engagement.

Moreover, our platform features an explore section where users can discover random posts from fellow members, promoting serendipitous connections and exploration. To streamline content discovery, users can search for posts using hashtags, with the option to click on hashtags to access all relevant content. Additionally, fostering interaction within the community, users have the ability to comment on posts, encouraging dialogue and collaboration.

Overall, our college media platform offers a comprehensive and intuitive user experience, empowering individuals to share, explore, and engage in a vibrant digital community centered around image-centric storytelling.

3 DESIGN

In the design of our college media platform, we've integrated essential functions to enhance user experience and functionality. Following user signup, users can seamlessly log in to their accounts, accessing personalized features and content. The login process is secure and straightforward, ensuring user data protection and authentication.Once logged in, users have the ability to create, edit, and delete posts from their respective profile sections. Leveraging Spring Boot as the backend framework, HTTP requests from the frontend trigger corresponding actions in the backend, such as storing post details in MySQL. Images associated with posts are stored in Firebase Cloud Storage, with the file paths stored in MySQL for efficient retrieval and rendering.Users also have the flexibility to edit their personal details, ensuring their profiles remain up-to-date and reflective of their identities. Additionally, users can edit and delete their uploaded posts, empowering them to curate their content and maintain control over their digital presence.

Users can share posts with others through notifications, fostering interaction and community engagement. When a user wishes to share a post, they can select the share option, prompting a notification to be sent to the intended subscribers. This notification includes a preview of the shared post, encouraging recipients to view and engage with the content. Users can securely end their session by logging out from their accounts, safeguarding their personal information and maintaining control over their interactions within the platform. Upon receiving a shared post notification, users can directly access the shared content, facilitating seamless content

The backend communicates with the frontend through HTTP requests, delivering relevant content details stored in MySQL tables. This seamless data exchange facilitates smooth navigation and interaction for users across the platform. Overall, our design integrates MySQL with Spring Boot to manage user-generated content effectively, while leveraging Firebase Cloud Storage for efficient image storage. Through a robust backend architecture and intuitive frontend interface, our college media platform offers users a seamless and engaging experience for sharing, exploring, and interacting within the community.

4 IMAGE STORAGE OPTIONS

There are mutliple storage options for the images like the internal servers local , cloud storing servers, CDNS like LANS etc we considered the internal(local), cdns(LAN) and the firebase(cloud) for our experiment we uploaded the images and downloaded them to test the speeds.

Internel Server(MYSQL):-Images stored within MySQL as BLOB data require careful consideration for performance and scalability. While MySQL offers robust security features, handling large BLOB data can impact retrieval speeds and scalability, necessitating strategies like vertical scaling and database sharding. Despite potential performance challenges, MySQL's reliability and familiarity make it a viable option for smaller-scale applications where image storage needs are manageable and cost-effective. We used the MySql of version 8.3.0- arm64

CDN-LAN(ORACLE):-Oracle Database within a CDN-LAN setup provides efficient image storage and delivery within a localized network environment. Leveraging local caching and Oracle Database's performance optimization features, CDN-LAN setups offer fast image retrieval speeds and scalability. Advanced security measures and enterprise-grade features make Oracle Database a suitable choice for large-scale deployments requiring stringent security and performance requirements, albeit at potentially higher costs compared to other options. we used the oracle version 19.3.0.0.0 which is already installed in the server cs2.uco.edu.

Cloud storage(FIREBASE):-Firebase Cloud Storage offers scalable and cost-effective image storage solutions with global availability and low latency. With automatic scaling and integration with Google's global CDN infrastructure, Firebase ensures fast and reliable image delivery to users worldwide. Robust security features and transparent pricing make Firebase Cloud Storage an attractive option for social media networks seeking scalable and secure image storage without the complexities of managing on-premises infrastructure. we uaed the Firebase version of 9.2.0

5 EXPERIMENTAL COMPARISION OF PERFORMANCE

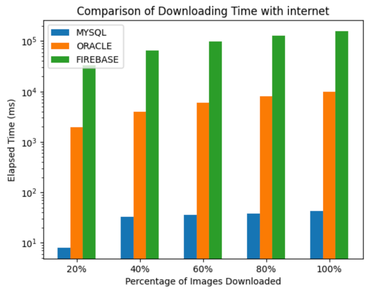
We conducted experiments in two different environments to measure the uploading and downloading times for Oracle, MySQL, and Firebase, using a MAC AIR with CHIP M3 running MAC OS Sonoma 14.4.1. The first environment utilized an Ethernet connection with an upload speed of 36 Mbps and a download speed of 862 Mbps. In this setup, the total upload and download times recorded were as follows: Oracle had an upload time of 3879 ms and a download time of 9993 ms; MySQL had an upload time of 30382 ms and a download time of 57 ms; Firebase had an upload time of 450479 ms and a download time of 170272 ms.

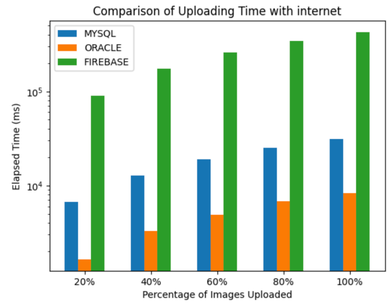
The second environment used an internet connection within the UCO college, where the local server system (cs2.uco.edu) was running. In this setup, the upload and download times were measured as: Oracle had an upload time of 8284 ms and a download time of 10006 ms; MySQL had an upload time of 30382 ms and a download time of 57 ms; Firebase had an upload time of 450470 ms and a download time of 170272 ms.

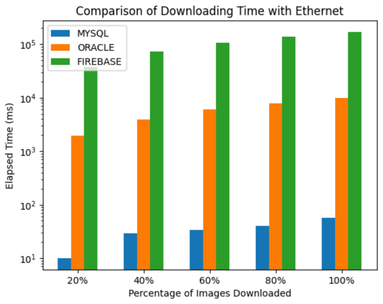
By averaging the upload and download times across both environments, we obtained the following results. For Oracle, the average upload time was 6081.5 ms, and the average download time was 10000 ms. For MySQL, the upload time remained constant at 30382 ms, with an average download time of 57 ms. Firebase showed an average upload time of 450474.5 ms and an average download time of 170272 ms.

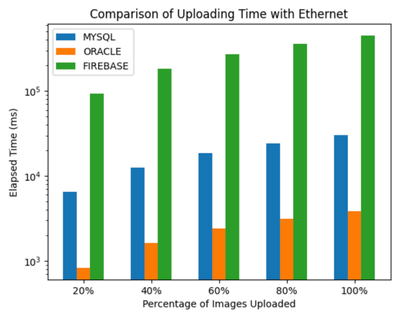
These results indicate that Oracle, MySQL, and Firebase have significantly different performance characteristics across different network environments.

Oracle demonstrated more consistent upload and download times between the two environments compared to MySQL and Firebase. MySQL had a constant upload time but exhibited very low download times. Firebase showed the highest upload and download times, which could be attributed to its larger data handling and storage operations.









6 CONCLUSION

In our extensive testing, Firebase's upload and download speeds are comparable for small datasets but degrade significantly with larger volumes, especially when integrated with backend frameworks like Spring Boot, likely due to network latency and server processing overhead. While Spring Boot's Hibernate repositories are convenient for basic operations, their efficiency drops with large datasets, where manual SQL statements offer better control and optimization. Using Base64 encryption for image uploads improves upload speeds and security but adds computational overhead for Oracle's downloads. In an Ethernet environment (36 Mbps upload, 862 Mbps download), Oracle's upload and download times were 3879 ms and 9993 ms, MySQL's were 30382 ms and 57 ms, and Firebase's were 450479 ms and 170272 ms. In the UCO college internet setup, Oracle's times were 8284 ms and 10006 ms, MySQL's were 30382 ms and 57 ms, and Firebase's were 450470 ms and 170272 ms. Averaging these results, Oracle had 6081.5 ms upload and 10000 ms download times, MySQL had 30382 ms upload and 57 ms download times, and Firebase had 450474.5 ms upload and 170272 ms download times. These findings highlight the importance of tailored strategies for optimizing performance and security in data-intensive applications, considering the trade-offs between different database systems, backend frameworks, and encryption methodologies.

7 REFERENCES

1. Research on Base64 Encoding algorithm and PHP implementation 26th conference on Geoinformatic 28-30 June 2018
2. Real Time Databases for Application IEEE volume 4 issue 06 June 2017.
3. Base64 encoding and decoding almost at almost the speed of a memory copy