World Music Web

Suangchun Shen / Shuang Liu / Yizhou Feng / Yiyang Zhao

1) Project Code

Project: music.alanzyy.com

Backup: music1.alanzyy.com

//The entire project code along with the final report should be zipped and submitted on Canvas.

2) Project Report

A modern software infrastructure project isn't done until you understand how it performs, and where the bottlenecks are. Instrument your application to collect timings on various aspects. You should at least be able to determine what the latency in handling each request is, and extra credit will be awarded if you can also see what happens under multiple concurrent requests.

Your final report should include a write-up of:

1. Introduction and project goals
2. Basic architecture (not a dump of the classes)
3. Key features of the project.
4. Technical challenges and how they were overcome
5. Description of your complementary data and how you extracted it
6. Performance evaluation
7. Potential future extensions

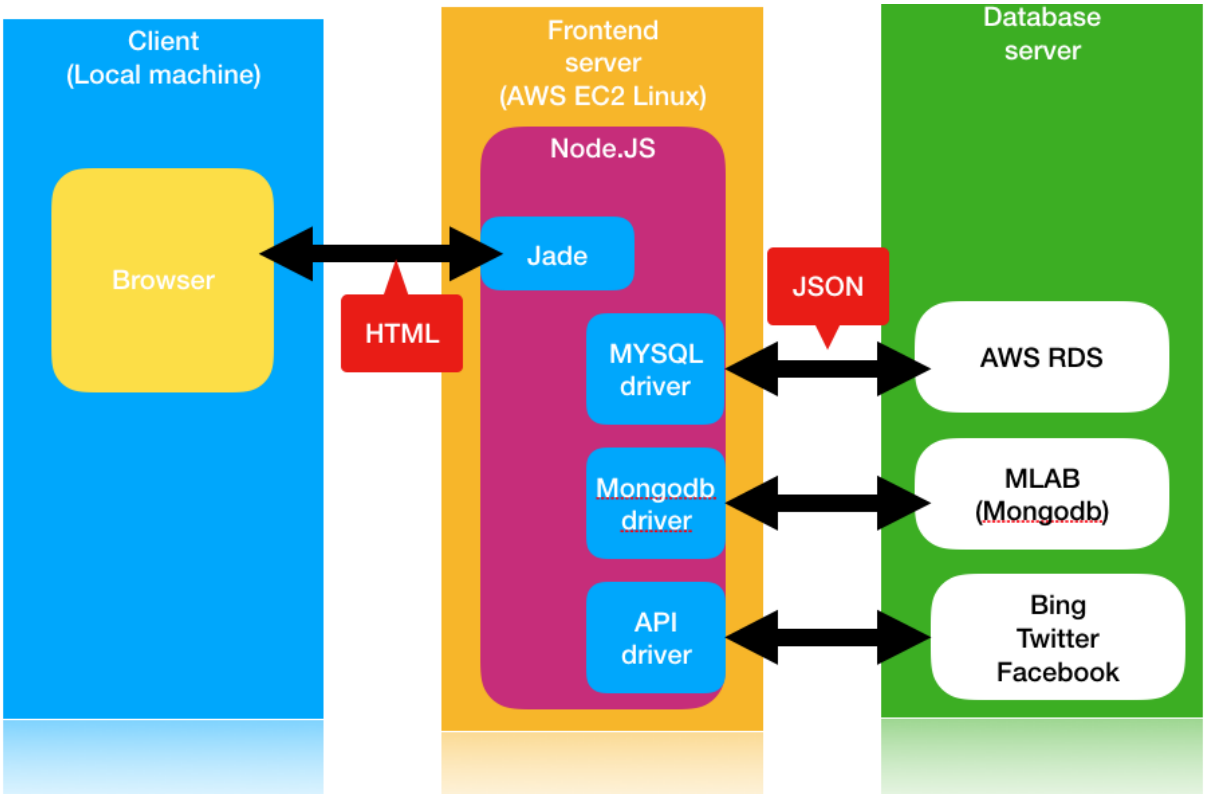
Introduction and Project Goals:

Our web application is the world music web, which enables users to get all kinds of data about music. There are many music apps in reality but most of them just serve as platforms for users to listen to or to download music but few of them help retrieve other information about music. Our goal is to make such unique web app that provides users with various kinds of information about music. For example, if you see a singer that you are not quite familiar with but you want to know more about him/her, then you can

directly search him/her on our bing search page instead of go elsewhere to get the information. Our app is like a gathering of all kinds of music information in which case if the users choose to immerse themselves in the world of music then they can spend hours on our web without using other apps.
In order to achieve all these utilities, we used node js as execution environment, mysql and mongodb drivers to retrieve data from databases, and APIs from facebook, twitter and microsoft.

Basic architecture:

\title{Top Level System Design}



We use NodeJS, deployed on two different AWS EC2 Linux nodes, as web server.
Jade framework is used to generate HTML pages from template. Whenever a GET or POST is issued by client browser, NodeJS triggers appropriate functions and send back results with Jade to client browsers.

A few NodeJS drivers are used to communicate with database server and external services. Our SQL database is hosted on AWS RDS service, and NoSQL database is hosted on MLAB, which is a MongoDB cloud server. MySQL and MongoDB drivers are used to query aforementioned databases. Facebook, Twit (for twitter), and Bing drivers are used to contact external services. Results are sent back in JSON format, and the data is then processed to be displayed to users.

Technical Challenges and How they were overcome:

We actually met numerous problems when developing the application,there are some problems that are time-consuming and tackling.

Problem1: Cannot connect to RDS DB Instance.

After we establish the RDS DB Instance in AWS following the guidance, we tried connect to the DB Instance using SQL Developer in the local end.However, we found the the server is always connecting to the DB, but never successfully.

Solution: At that time what we estabilshed was the Oracle DB, which is not very popular, so few of resources on the website describes it in a detailed way. However, MySQL is very popular, so by analogy, we look through the guidance of installing MySQL in the AWS. And then we found that we did not cover the step of setting the security group in the configuration. By setting the security group configuration as receiving IP address from everywhere, we can connect to the RDS DB Instance successfully.

Problem2: Cannot Connect with OracleDB in AWS through Javascript.

After we solved the problem about how to use SQL developer to connect with OracleDB in AWS, we met a problem of connecting it with node.js. There was always timeout when running the javascript file.

Solution: We have tried many ways, such as downloading the drivers, installing npm packages, or even reinstalling the OracleDB in AWS. None of these methods worked. We assumed perhaps part of the reason was the database problem, so we changed from the OracleDB to the MySQL. It turned out the with javascript we can easily connect with the MySQL.

Problem3: Cannot pass global variable in Javascript SSC sb

As we are new to the javascript, we find that even if you change the value of global variable in one function and use the value in the other one, the value stayed the same as before the change.

Solution: After searching all kinds of tutorial about using Javascript functions, we find the property of callback function.We use callback function to do this.In order to do this, we need to put a function in the other one, and use the value return by the function nested in the other function.

Problem4: MongoDB Version Problem SSC pig

Since the group uses the github to synchronize the progress, there is a very strange phenomenon. The codes can be successfully executed in someone's computer while cannot run in others'.

Solution: We searched the exception the command throws and found that some api in the codes are valid only on a previous specific MongoDB version. For others who installed the latest version of MongoDB, the codes were already out of date. To solve this problem, all of us updated our MongoDB, and changed the api to the latest one.

The problem lies in MongoDB's bad practice: lack of backward compatibility. Newer version of engine does not recognize query in older format.

Description of your complementary data and how you extracted it:

- The nosql database we used was from UCI Machine Learning Repository: A Dataset For Music Analysis Data Set. The dataset was in csv file and we transformed it into json, and then populate it into Mlab. The dataset consists of four tables: genres, raw_albums, raw_tracks, raw_artists.

  Genres: stores information of all types of genres

  Raw_albums: stores all types of information about album, including id, title, url etc.

  Raw_tracks: stores all information about tracks, including id, title, url, etc.; it also includes foreign keys to other tables.

  Raw_artists: stores all types of information about artists, including id, title, url etc.
- The relational database we used was from Millon Song dataset. The original dataset was in db file and we transformed it into SQL file and populate it to AWS

RDS through command line scripts. The dataset consists of four tables: artists, artist_mbtag, mbtag, songs.

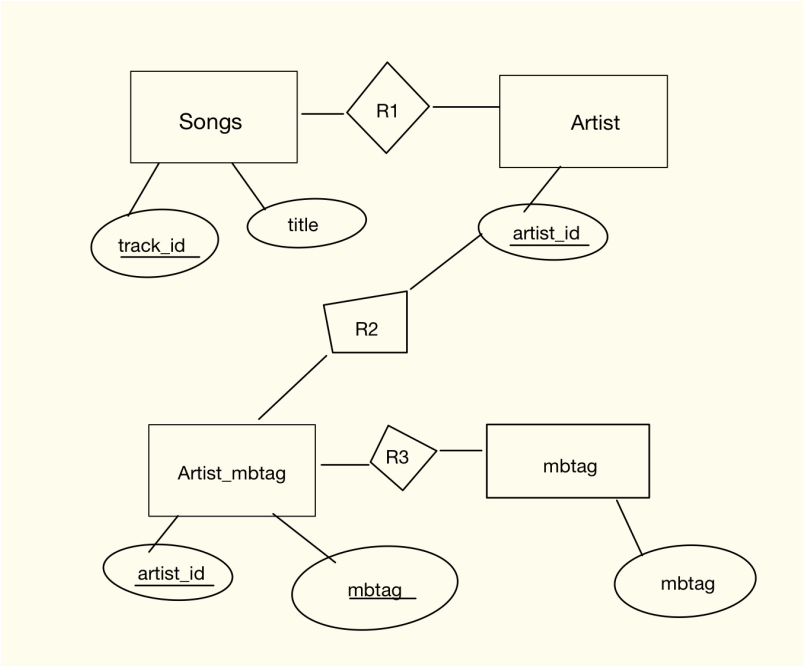
Artists: stores all information about artists.

Artist_mbtag: stores foreign keys referenced to artists and mbtag.

Mbtag: stores all information about genres.

Songs: stores all information about songs, and it also includes foreign keys referenced to other tables.

The relational schema looks like this:



Performance Evaluation:

A few consideration on performance of the system is considered:

- Fault Tolerance: Our frontend has only two nodes, it is quite possible that some error is thrown. We tries to filter input and catch errors so a small error will not crash the whole web service.
- Speed: We discovered that AWS EC2 is reasonably fast, but the limiting factor is SQL query execution. We indexed several key field based on our query to make it faster. For example, in our relational database, we created indexes like idx_artist_id, idx_artist_name, idx_hotttnesss, idx_year. We removed some unuseful attributes and some records with excessive null values, except those

added through “Addition” function in our web page. At this moment, all webpage has a reasonable opening time <300ms.

Key Features:

Core features:

- **Search Music**: it allows users to search the music information stored in the database
- **Play and download free music**: also allows users to play and download musics using HTML5 objects referenced to Free Music Archive;
- **Recommendation**: users also can see the information of songs recommended by our website based on their popularities;

Extended features: (for extra credit)

- **Bing Search**: it allows users to search the music information directly through bing search api through the website;
- **Twitter feeds**: users are able to see the twitter feeds about the music and singer they want to learn about;
- **Facebook Login**: users are able to login through their Facebook accounts and see their personal information.

Potential Future Extension:

- Currently we don’t have a friend-adding system on our web app, but if given chance, we would like to enable users to add friends and also to communicate with friends on our web app.
- We may implement a more personalized recommendation system based on user search history.
- Enhance search database process, including supporting Fuzzy Search. Or FYZ search.
- We lack a load balancer to distribute user requests onto two nodes.