r/Place Project Report

Architecture:

The architecture of our r/place is that we have web servers and websockets server containers running in Docker on ECS, and they have are connected to DynamoDB, which is a database that chose for this assignment. The client would connect to the web server and obtain the webpage, which would make a socket connection to the Nodejs server to obtain the board and updates. We are using Redis for our cache, and it will store our entire canvas in a bit field.

When the client first connects, the server would send the board to the user, by reading it from Redis, as a bit field, and send directly for the client to translate to the canvas. The database we are using is DynamoDB, and we are using it to store information about the entire canvas, and the users that are on it. Whenever a user, attempts to paint a tile, the request would be sent it the server, and the server will perform a query on the database to see if either the user doesn’t exist (which means the user never painted before) or if the user does exist, check if the last time they painted is not within the last 5 minutes.

If either of the conditions hold, then we put the entries in the database, and we update Redis. Afterwards we send the request back to user as an acknowledgement that their request is accepted, and the change is propagated to all users.

Strengths and Weaknesses:

Our strength is that our service is scalable since we are using Docker, and our service is highly available. We have multiple instances of our application running, so there isn’t one single source of failure when the user tries to enter the application. We can also generate replicas of our Redis server to scale up our application, and same goes with our web servers and nodejs servers, which has been shown to handle 600,000 connections concurrently. Our DynamoDB server is based on the same architecture as Apache Cassandra, so we have the benefit of our data being safely replicated and partitioned among its internal cluster. We can of course scale up the read and write capacity when needed. Because we are using a reliable DBMS to manage our data, and we are only reading from Redis in one place and writing to both Redis and DynamoDB in the same request, our system is strongly consistent.

Our weakness is that we need to prepare Redis when starting up, by setting the board all to 0’s first manually, for the first time. Also if it ever goes down, then it becomes problematic since we’re unable to transfer everything that we have from our database to our Redis cache as we don’t have an automated way of doing that.