Evan Donald

April 23, 2019

1. How to solve race condition (multiple clients in one seat)

To solve the problem of multiple clients reserving the same seat, I decided to use a seat map to send back and forth between servers. I also used an array to keep a lock on seats. If the seat is unlocked the value is -1, if the seat is locked the value is 0, and if the seat is reserved the value is 1. I created an object that contains the seat map and the array of locks. This object is used by each server to keep track of who has reserved each seat and it is also used to check the lock of a seat before it is reserved. If a server receives a request to reserve a seat, the server sends a request to all the other servers to lock that seat. If that seat is already locked or reserved the new request to reserve that seat is denied. If there is no lock on the seat, the server completes the reservation and locks the seat on all the other servers.

1. How to make the system fault tolerant

To make the system fault tolerant, I created a primary server that contains a list of secondary servers. The clients are only connected to the secondary servers. I created a class that makes sure that the client is connected to a server by continuously pinging. If a server fails, the user is reconnected to a new server.