Health.Mate

Joshua Antonson, Aayush Agarwal, Mikhail Kutsovsky, Priya Narasimhan

Introduction and Problem Statement

- As the world becomes more flat, individuals are accessing several more hospitals than before. This leads to scattered personal health records which is unpractical for self-management.
- Health.Mate allows users to maintain a centralized location to view and add to their collection of personal health files.

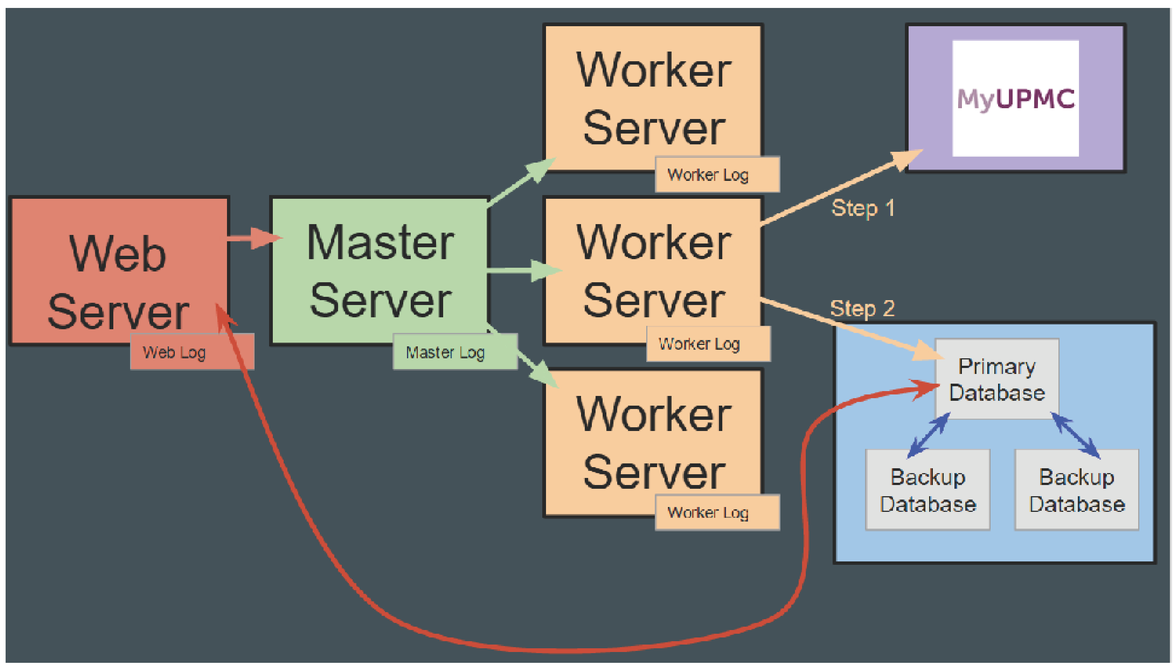
- Goals
  - Usability – create a secure account, collect health records from medical providers, view records whenever
  - Fault-Tolerance – a system which is able to withstand servers or databases going down in the back-end of the applciation

- Non-goals
  - We did not replicate our back-end workflow coordinator.

Demonstration

- Access your own health files
  - If you are a myUPMC user, you can create an account on health.mate and enter your myUPMC credentials to view your current health files

- Kill one of our servers
  - Start a request to collect health files and kill the machine which is doing the work



Architecture and Fault Tolerance

- Our system is entirely deployed on AWS and has the following components:
  - Web Server – Serves as the user interaction layer to our application, this is the only box which allows for “public” requests
  - Master Server – our single point of failure, acts as the coordinator for requests from front-end to workers
  - Worker Server – communicates with the hospital’s severs to access files and then writes files to our own secure database
  - Database – passively replicated MongoDB instances in AWS which get files from workers and sends PDFs to front end

Experimental Evaluation

- The top graph shows average time of all operations required for case when a worker server crashes in the middle of processing a request
- The bottom graph shows the performance of each of the servers when handling concurrent requests