Handshake

A Mobile Application for Contact Sharing

Rohit Malhotra
Prateek Jain
Gaurav Gupta

Motivation & Goals

The main idea herein is to develop a mobile application that would allow people to easily exchange contact information among each other by leveraging near field communication between their mobile phones, in conjunction with push notifications from a remote server to the phones.
Contact information may refer to -

Phone Number

Facebook Profile’s Url

Twitter Handle

LinkedIn Profile’s Url

Personal Website’s Url

Github Profile Url

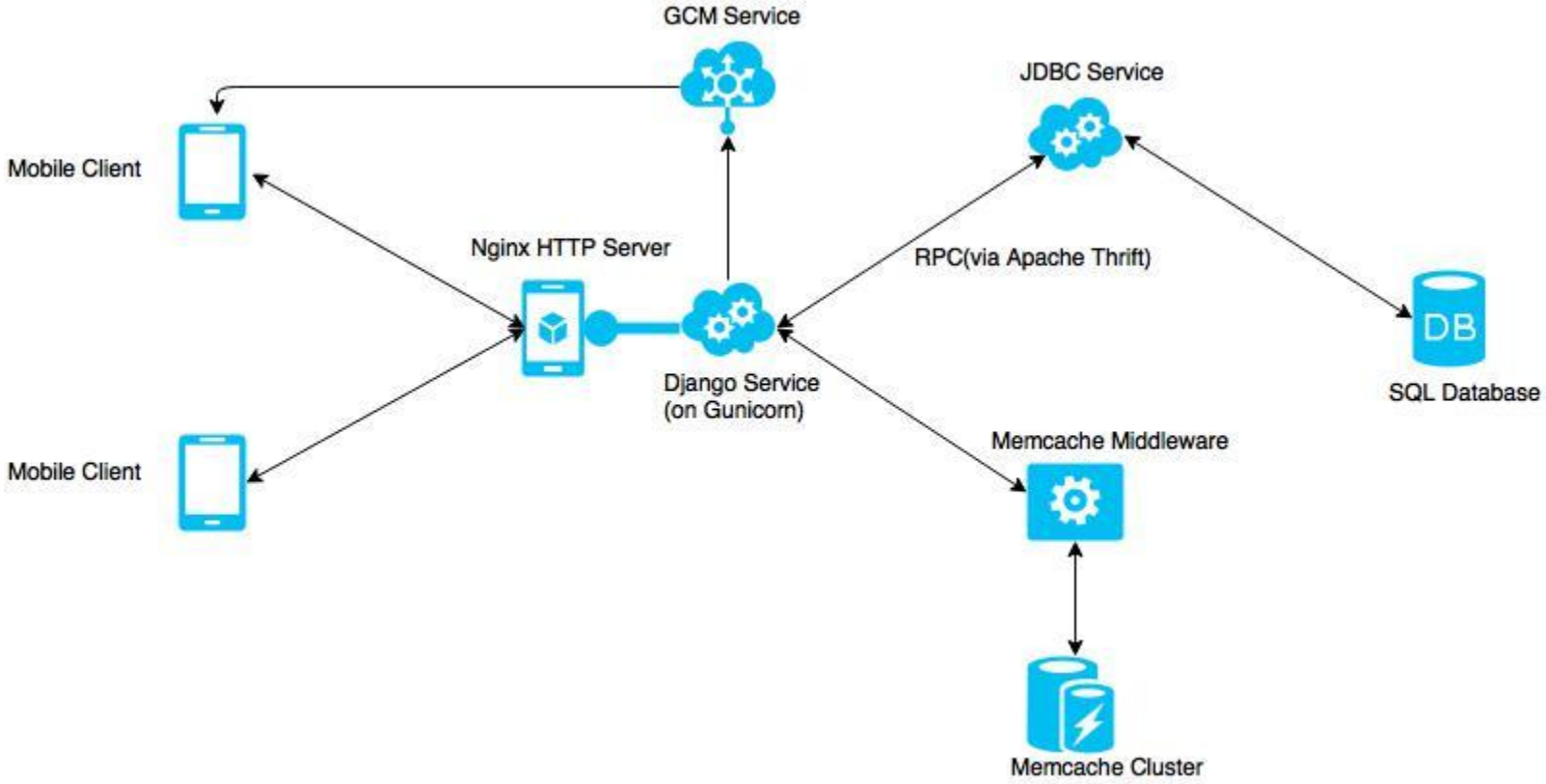
Related Work

- Bump - A motion sensor based android app that enables users to share any data between two phones by tapping them together and sending location data to a server. (requires proximity)
- Share Contact - Share contacts with other people over SMS or internet. (proximity not required)
- Slip - Instant Contact Sharing (for iOS only)

Middleware Concepts/Technologies

1) IAAS**:** We use **Amazon EC2** from AWS as our infrastructure to host the backend of our application
2) RPC Runtime **:** We use **Apache Thrift** as our RPC runtime. It provides an IDL (Similar to CORBA).
3) Distributed Shared Memory**:** We use two **memcached** servers running on two different ec2 instances forming a memcached cluster. Our application uses this as a layer of caching over the database interactions.
4) Memcache Middleware **:** We use **'mcrouter'** as a middleware which provides Qos such as high availability, fault tolerance and pool based allocation of memcached servers. This can help us horizontally scale up the application in case we need to add more memcached servers.
5) Java Based Middleware**:** We use **JDBC** to connect to the MySQL database from our Java thrift service.
6) HTTP Server**:** We use **Nginx** as our front end HTTP server for interactions with the mobile client.
7) Asynchronous message Delivery Service **: Google Cloud Messaging** platform
8) Middleware Framework: Python **Django** Framework

System Architecture



Design Rationale

We have designed the system in order to focus on the following non-functional properties of the system.

1) **Runtime Performance:** For our query processing from MySQLDB, we use a Java service running over an RPC runtime. Since Java has better runtime performance than python, we get less turnaround time for query processing.
2) **Fast Read Performance:** We use memcache demand-filled look-aside cache on top of our MySQL database. This helps us save database I/O cost for frequently issues queries.
3) **Reliability:** We use **mcrouter** (memcache middleware) which provides the option of server pooling of memcached servers and provides QoS like fault tolerance and high availability.
4) **Interoperability:** We use Django as our frontend web framework since python is suited to fast paced development. For interacting with services written in other programming languages, we use Apache Thrift which provides an IDL to write interoperable services.