ABSTRACT
The main work of our project is a Facebook based application which makes recommendation to Facebook users, on which business entities can publish coupons and customer can get recommendation based on their facebook profile, subscription and friendlist etc. Distributed architecture, NoSQL database and cloud computing technologies are applied in our implementation to make the application highly available and scalable. An IT infrastructure to hold this facebook application is designed and then we deploy application, Hadoop[1] file system, and HBase[2] cluster on Amazon EC2 and evaluate the performance of our work.

Keywords
Facebook application, coupon recommendation, distributed architecture, cloud computing.

1. INTRODUCTION
Along with the popularity of Facebook, more and more third party applications are emerging based on Facebook platform, which provide complementary functionality that has not been implemented by Facebook. The function of these Facebook application usually take advantage of users’ data to better understand user thus serve a better service and the friendship on facebook is applied to make the application more interactive.

Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software more attractive as a service and shaping the way IT hardware is designed and purchased[3]. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. The garage innovator creates new web applications which may rocket to popular success or sink when the flash crowd that arrives melts the web server. In the web context, cloud computing provides a way on which the innovator can, with minimal capital, prepare for overwhelming popularity[4].

Inspired by existing all sorts Facebook applications, we designed an application to recommend coupons to Facebook users and implemented a demo. To handle the flash crowds, we apply distributed database and cloud computing technologies to make the system scalable and highly available.

2. Design and Implementation
2.1 Overview of application

As shown in Fig. 1 there are two types of users using our application. Business Owners and the End users.

2.1.1 Business Owners
Business owners are the business partners who have a certain type, location associated with them. The business owners provides the application their Facebook page, using which the application gathers the relevant information such as geo information, the type of business etc. After registering the business with the application, the business owners can add deals and discounts (referred to as Coupon) within the application. While adding the coupon, the business user can target it towards a particular target audience by providing information such as Age range, Gender etc.

2.1.2 End Users
These are the main users of the application who access our application via Facebook app interface. The user has to provide the application with a certain level of permissions to access her Facebook data. We use the recommendation logic which will be described later, to provide users with the coupons she may be interested in.

2.2 Architecture
There are three layers in our coupon recommendation system: Proxy server is the interface to the public internet, which accepts Http requests from the clients and dispatches the requests to the backend application servers. Also it works as the load balancer for the application server cluster. And we adopt the Hadoop to provide a reliable, scalable, distributed computing and data storage layer, upon which HBase is installed as the distributed database, zookeeper is configured to provide distributed synchronization which is also exposed as the interface to access the data storage layer.

Actually we make cluster for each layer except for the proxy server, so it is redundancy that achieves high availability.
2.3 Application Design

The application is implemented using a 3-tiered architecture. The Presentation layer which is presented to the user inside Facebook is composed of JSP’s. The Database layer is implemented using HBase on top of Hadoop and the application layer is implemented using JAVA. The application logic is split into three parts, the first part deals with the business logic and interacting with the front end, the second part deals with the data access layer and the third part deals with extracting data using Facebook APIs.

2.3.1 Recommendation Logic

One of the important things in our design is about how we make recommendation to customers. Basically recommendation can be regarded as making association between published coupons and customers, helping customer to get the coupon they may be interested in. The question here is how to do recommendation reasonably.

We designed and implemented four different strategies to do the recommendation:

- **Filter based (profile based) recommendation:** when the business users publish the coupon, they need specify the target customer by choosing the value of certain filters. For example, a café or bar can publish a coupon targeted to the customers who are single, or a luxurious hotel may specify a coupon is for the customers whose age is between 40-50, because they may have strong purchase ability.

- **Nowadays, social network is becoming one of the important part of the internet. Thus people in the network can be better profiled using their statuses in the social networks (friendship) Since all the customers of our system extract user profile from Facebook, we design an algorithm to do the recommendation as follows:** After a customer register in our system, we maintain the list of his friends in our database. Every time he likes a coupon, the system will recommend this coupon to his friends. The same action will happen when he checks in a place with a coupon.

- **Subscription:** customer can choose which category to subscribe and which business entity he is interested in. After subscription, he will get all the coupons published by the corresponding business user.

- **Location based recommendation:** system will recommend the coupons published by nearby business entities, based on his current checkin place.

2.3.2 Tools and Technology

We have used Idea IntelliJ IDE for development of the JAVA web application. IntelliJ offers simplified web development and is light-weight when compared to other IDE’s such as Eclipse for web development.restFB is used as for accessing Facebook API. restFB provides JSON objects directly into JAVA objects and hence are easier to use within a JAVA application. We decided to use HBase for database because.

2.4 Database Design

To achieve the scalability of the application we use HBase as the backend data storage. We consider the choice of the NoSQL database for several reasons:

- HBase is column oriented, which is good for storing sparse data. User’s profile in Facebook is not complete, some data field may be blank, so storing it in HBase is space economical.

- HBase is schema-less, so it makes the application more flexible and scalable. We can easily add new columns to the database once the functionality of the application changes in the future.

- HBase is on the top of Hadoop which is an open source implementation of GFS. Therefore, by keeping replicas in multiple regional server, HBase is a distributed fault tolerant database system, which facilities the high availability of our system.

- By designing the storage schema and row key carefully, queries from the application can be optimized.

<table>
<thead>
<tr>
<th>Table 1. Database schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table name, rowkey:</strong> User: id</td>
</tr>
<tr>
<td>Family</td>
</tr>
<tr>
<td>column</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Table name, rowkey:</strong> business_user, id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
</tr>
<tr>
<td>Column</td>
</tr>
</tbody>
</table>

| **Table name, rowkey:** checkin, id |
The design of the HBase schema is different from traditional RDBMS tables. In RDBMS we need consider the design according to 3NF. There are conventional strategies to map from ER diagram to data columns for 1-m relation and n-m relation, which is mediate tables. In NoSQL database, we can break n-m relation into multiple 1-m relation and even store rows as columns according to the queries of applications.

Rowkey design is another important issue of NoSQL database, because the default index of a table is built based on the rowkey. Therefore the rowkey can be well designed for retrieving bulk data without scanning the whole table. it can be applied to optimize the query.

Table user stores the user profile information, it store friendlist as a column family so that the table will grow horizontally, and the whole column family works as an index for querying one’s friendlist.

In table business_user, we design it in the similar way for 1-m relation, storing published coupons and the subscribe_user in corresponding column families. So it is especially indexed for corresponding queries to get business users’ published coupons and recommend coupons to subscribing users.

In coupon table the rowkey is designed as the id concatenate by latitude and longitude, which is designed to match the latitude and longitude in location based recommendation without scanning the whole table.

2.5 Physical Deployment

We initialize three VM instances on Amazon EC2 and deploy our system on these VMs.

Proxy server: Nginx works as the Proxy server, which dispatches the incoming requests to the application server, it also balances the work load based on Round-robin strategy.

Application server: two Tomcat servers are running on VM2 and VM3, which implement the business logic.

Data storage layer: all of three virtual machines run as the Hadoop DataNode, HBase regional server and zookeeper instances. And at the same time VM1 works as the HMaster and Hadoop NameNode. In this configuration Therefore, all three VMs form a high available, fault tolerant data storage backend with three replicas.

3. Evaluation

We evaluate our work based on three aspects: Performance of our system, availability and scalability.

3.1 Performance

We use Jmeter to make high pressure test for our application servers cluster and the zookeeper cluster. The configuration is as follows: We configure 1000 threads to simulate the concurrent requests, and the testing process lasts for 1 minutes, during which it loops in the following pattern: the client sends requests for 10 seconds and stop for 5 seconds.

Figure 4. Response time of a single application server

Figure 5. Response time of Zookeeper cluster
Fig. 4 and Fig. 5 indicate that while the high pressure test is going on, the response time of a single server increases rapidly. Also we find the zookeeper server exhibits some adaptation to the crowds after a short break of 5 seconds, and the response time becomes stable and does not increase any more.

3.2 Availability
In our implementation we replicate application servers and adopt a load balancer, also we adopt HBase as the back-end data storage layer, which maintains multiple replicas can tolerate the failure of a single node. The weak point of the system is the master node, once it crashes, the system can not serve the requests any more, through we keep a second copy of HMaster node on VM3, it is not a running copy, thus it takes time to recover. Therefore our system is generally available but has a weak point.

3.3 Scalability
Our application is scalable in three aspects:

Application level. The basic functionality of our application is recommendation. As described in previous section, we design the recommendation algorithm based on different strategies so that the recommendation criteria is easily extended. One the other hand we separate the recommendation action into different time phases (eg. make corresponding recommendation when new user comes and when new coupon is published.) therefore the computation can be spread over different time period to average the workload. Once the recommendation becomes intensive, we may introduce MapReduce into our system to spread the computation workload over multiple servers.

Database storage level. We take advantage of NoSQL database to easily handle new added data field therefore when the application is extended and new function is added, more related data field from facebook can be extracted and persistent into the database.

Infrastructure level. We have currently three VM instances running on Amazon EC2. Among which, VM1 works as both a manager and a slave of the whole cluster, and VM2 can be regarded as a slave. Therefore to handle more workload, more instances like VM2 can be launched, and simply notify the Master the insertion of new node. So the system is easily scale up. It is similar for scaling down.

4. REFERENCES