Project

Twitter Dataset Analysis and Modeling

Student Name: Srinivasa Rao Gorijavolu (SGORIJAV)

**Problem definition:** Testing various data modelling techniques using mongodb and benchmarking performance.

**Software Used:** Python, MongoDB

**NOSQL Data Modelling Using MongoDB:**

There are quite few ways we can do data modelling in NoSQL. Two ways of models have been chosen for bench marking performance.

***Model A (wider table approach)***

The main document collection called “TwitterUser” contains all the users along with their tweets and followers as sub or nested documents. All three sets of Twitter data like users, network and tweets were loaded into this table.

There will be one document for each user and his network (followers) and his tweets are part of this document.

Here is the sample record:

{ "\_id" : 2195791,

"followercount" : 3813,

"name" : "bsandslaw",

"friendcount" : 4589,

"userlocation" : "LOS ANGELES, CALIF",

"joineddate" : "Sun Mar 25 13:30:35 CDT 2007",

"statuscount" : 11,

"favoritecount" : 0,

"followers" : [ 65499120, 115186483, 80374470, 113374955, 156604701, 101533343, 217129209, 172699031, 73817037, 98413321, 42316937, 205986528, 70006588,48171456, 165940419, 184256802, 179295554, 98055163, 206267245, 67976600, 151063632, 173147858,………],

"tweets" : [

{ "Text" : "Brry Snds on you tube - topi - dui defense" },

{ "Text" : "See Brry Snds on you tube - wwwyoutubeom/user/brrysndslwyer" },

{ "Text" : "Former GUNS N ROSES Drummer" },

{ "Text" : "Criminl Defense Attorney Brry Snds with elebrity lient Steve-O" },

{ "Text" : "" },

{ "Text" : "See Criminl Defense Attorney Brry Snds with elebrity lient Steve-O Wth Video" },

{ "Text" : "See Criminl Defense Attorney Brry Snds with elebrity lient Steve-O Wth Video" },

{ "Text" : "Criminl Defense Attorney Brry Snds with elebrity lient Steve-O Wth Video" },

{ "Text" : "Criminl Defense Attorney Brry Snds with elebrity lient Steve-O Wth Video" },

{ "Text" : "See Criminl Defense Attorney Brry Snds with elebrity lient Steve-O Wth Video" },

{ "Text" : "See Criminl Defense Attorney Brry Snds with elebrity lient Steve-O Wth Video" }

]

}

This model has many advantages. Everything in one document and number of documents are equal to number of users. This makes collection wider and table thinner in terms of row counts which is good for NOSQL databases.

**Model B (leaner rows but vertically bigger table)**

In this case each tweet will have one document and corresponding userid is embedded within the document. This makes collection or table very longer but smaller row size.

One of the sample row is like below:

{

"\_id" : ObjectId("56228929d53f0e057aad58d8"),

"Origin" : "“Our greatest weakness lies in giving up The most certain way to succeed is always to try just one more time” -- Thomas A Edison",

"URL" : "",

"Text" : "“Our greatest weakness lies in giving up The most certain way to succeed is always to try just one more time” -- Thomas A Edison",

"Hashtags" : "",

"userid" : 20000024,

"Favorite" : "false",

"RetCount" : "3",

"Time" : "Mon Jul 18 06:58:15 CDT 2011",

"Type" : "status",

"ID" : "92926150908710912",

"MentionedEntities" : ""

}

There are other models that can be used like using separate documents for users and tweets but this requires joins and lot of programming which may result in poor performance.

**Using python to populate data.**

Python programming language is used to connect to mongodb and load the data from files into mongodb collections.

Files were attached in canvas to populate both collection models.

**Benchmarking Performance**

**Search both models for words in their Tweets. And below are the timings for each model.**

**From model I (twitteruser)**

**MongoDB commands:**

**$> db.twitteruser.count()**

**1507814 (total users)**

**$> db.twitteruser.find({tweets: { Text: "good"} }).count()**

**Returned 55 docs in 6 secs**

**$> db.twitteruser.find({tweets: { Text: "sjdfjksdfjdsfjdlsfkl"} }).count()**

**Returned 0 docs in 3 seconds.**

**Model II (Tweets)**

**Mongodb commands**

**$> db.tweets.count()**

**11701482 (total users)**

**$> db.tweets.find({tweets: { Text: "good"} }).count()**

**Returned 57 docs in 32 secs**

**$> db.tweets.find({tweets: { Text: "sjdfjksdfjdsfjdlsfkl"} }).count()**

**Returned 0 docs in 30 seconds.**

**Adding a follower to a user, update all the necessary collections.**

Adding follower (100043628) to user 100049128

**Mongodb code:**

**> use twitter**

**switched to db twitter**

**> db.twitteruser.update({\_id: 100049128}, {"$push": {followers: 100043628}})**

**WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })**

**> db.twitteruser.find({\_id: 100049128})**

**{ "\_id" : 100049128, "followercount" : 2197, "name" : "EliseSandstw12", "friendcount" : 2315, "userlocation" : "", "joineddate" : "28 Dec 2009 21:15:15 GMT", "statuscount" : 13475, "favoritecount" : 0, "followers" : [ 100043628 ] }**

**Findings and Summary:**

For the NoSQL databases like MongoDB, The big table approach works good from both performance and disk space point of view.

I also came up with new data model for NOSQL databases like Mongodb, Cassandra called “Set Relational” model and I wrote a book on this which is available on amazon.com.

<http://www.amazon.com/Cassandra-Practical-Warehousing-Architecture-Modelling-ebook/dp/B013FL04GK/ref=sr_1_1?ie=UTF8&qid=1449207490&sr=8-1&keywords=practical+data+warehousing>