EXPERIMENT 06 & 07

Aim (06)

AirPhone Corp is a famous telecom company. They have customers in all locations. Customers use

AirPhone Corp’s network to make calls. Government has brought in a regulation that all telecom

companies should store call details of their customers. This is very important from a security point of

view and all telecom companies have to retain this data for 15 years. AirPhone Corp already stores all

customer details data, for their analytics team. But due to a surge in mobile users in recent years, their

current database cannot handle huge amounts of data. Current database stores only six months of data.

AirPhone Corp now wants to scale their database and wants to store 15 years of data.

Data contains following columns – Source: Phone number of caller; Destination: Phone number of call

receiver; Source\_location: Caller’s city; Destination\_location: Call receiver’s city; Call\_duration: phone

call duration; Roaming: Flag to check if caller is in roaming; Call\_charge: Money charged for call.

Sample Data:

{ source: “+919612345670”, destination: “+919612345671”, source\_location: “Delhi”,

destination\_location: “Mumbai”, call\_duration: 2.03, roaming: false, call\_charge: 2.03 }

After discussing the requirements with database and architecture team, it has been decided that they

should use MongoDb. You have been given the task to Setup a distributed system (database) such that

data from different locations go to different nodes (to distribute the load):

 Import data to sharded collection

 Check data on each shard for distribution

Aim (07)

Execute below sets of problem by taking reference of Experiment Number 06 and find out:

 Add additional node to existing system (to test if we can add nodes easily when data increases)

 Check the behaviour of cluster (data movement) on adding a shard.

 Check the behaviour of query for finding a document with source location Mumbai.x

Procedure

Server creation:

• First we create multiple folders for the servers we’re creating through the command prompt,

‘C:\>mkdir \dataS\rs1 \dataS\rs2 \dataS\rs3’.

• We then configure each of them:

◦ ‘C:\>start mongod --replSet kau --logpath \dataS\rs1\first.log --dbpath \dataS\rs1 --port

27017 --smallfiles --oplogSize 64’

◦ ‘C:\>start mongod --replSet kau --logpath \dataS\rs1\first.log --dbpath \dataS\rs1 --port

27017 --smallfiles --oplogSize 64’

◦ ‘C:\>start mongod --replSet kau --logpath \dataS\rs1\first.log --dbpath \dataS\rs1 --port

27017 --smallfiles --oplogSize 64’

Starting the first server:

• In the command prompt we then start the first MongoDB server, ‘C:\>mongo --port 27017’, and

restart it with ‘db.adminCommand({shutdown: 1});’.

• Then configure the server by setting up configuration attributes with, ‘config = { \_id: "kau",

members: [ { \_id: 0, host: "localhost: 27017"}, { \_id: 1, host: "localhost: 27018"}, { \_id: 2,

host: "localhost: 27019"}]};’, and configure with the command, ‘rs.initiate(config);’.

• This is out PRIMARY server, and we can check its status using ‘rs.status();’.

• After this we insert data into the server, ‘db.airphone.insert([{source: "+919612345670",

destination: "+919612345671", source\_location: "Delhi", destination\_location: "Mumbai",

call\_duration: 2.03, roaming: false, call\_charge: 2.03 }, {source: "+919612345670",

destination: "+919345371345", source\_location: "Delhi", destination\_location: "Chennai",

call\_duration: 1.37, roaming: false, call\_charge: 1.37 }, {source: "+919345371345",

destination: "+918293977188", source\_location: "Chennai", destination\_location:

"Bangalore", call\_duration: 5.55, roaming: false, call\_charge: 5.55 }]);’.

• We can check the data ‘db.airphone.find().pretty();’.

Start the second server:

• In a new command prompt window we start the second MongoDB server, ‘mongo --port 27018’.

• We need to make this the secondary server, which we can do with, ‘rs.slaveOk();’.

• We can now see the data from the primary server on the secondary one,

‘db.airphone.find().pretty();’.

Output

The following are some of the output for the above commands:

