**<Platforms>**

201533661 이승수’s Machine Learning HW#9

**Task1) Write a report on the detailed architectures of Google File System and Google BigTable.**

**GFS(Google File System)** is distensible distributed file system developed by Google. It is optimized for enhancing Google’s core data storage and search engine. It’s focused on enhancing throughput instead of minimizing latency.

It’s used in large and distributed applications which need to access mass data. However, difference from traditional file system is that GFS is available and designed to runs on cheap, common hardware.

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/GoogleFileSystemGFS.svg/300px-GoogleFileSystemGFS.svg.png | Google File System Architecture |
| **Figure1) GFS map-reduce flow** | **Figure2) System Structure of GFS** |

GFS is designed for system-to-system interaction, increasing credibility and distribute processing. GFS clusters data into 64MB block-sized data ‘chunks’, which can be read and write(cannot overwrite) only.

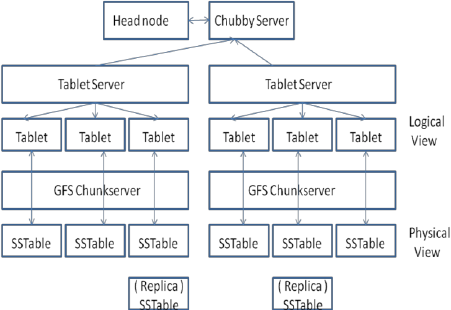
GFS clusters includes master server and many chunk servers. When creating a block, chunk server save chunks at local hard disks with Linux file system. Then for reliability, each chunks will be copied to several block servers. Master server manages all metadata of file system for handling(read/write) data from chunks.

**Google Bigtable** is a compressed, high performing proprietary data storage system built on GFS, and so on. It’s initially released at February 2005, public version was made available as a service on May 2015 on Google Cloud Platform.

It’s one of NoSQL database, which maps two arbitrary string values(row key, column key) and timestamp(for 3-dimensional mapping) into an associated arbitrary byte array.

For example, for Google’s web copy, each row key holds domain-reversed URL, and columns key holds various properties of a web page. Each page columns can have several timestamped versions describing different copies of the web page .

BigTable is somewhat like a mapreduce worker pool in that thousands to hundreds of tablet shards may be served by same number of BigTable servers.



**Figure3) Google BigTable Architecture**

BigTable has three different types of servers named Master, Tablet, and Lock Servers. Master server do assigning tablets to tablet servers, balancing load, detecting the loss or addition of Tablet Servers and garbage collection. The master keep communications with Tablet Servers to ensure that they continue to hold on to their corresponding locks.

**Task2) Write a report on how to create a collection, populate it with data, search data, and update data using MongoDB. Use a concrete example. Do the same with an RDBMS.**

For MongoDB, First install MongoDB at homepage and start cmd command. After settings, put queries as below to create, insert, search, and update data into DB.

Tables below is comparing RDBMS and NoSQL queries doing same functions for each actions for building database.

**<Create Collection>**

-To Create new database and table/collection.

|  |  |
| --- | --- |
| **Database type** | **query** |
| RDBMS(mysql) | ●CREATE DATABASE <db\_name>  ●USE <db\_name>  ●CREATE TABLE books(  Id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,  Name VARCHAR(10) NOT NULL,  Author VARCHAR(20) NOT NULL,  View INT NOT NULL ) //create ‘books’ table |
| NoSQL(MongoDB) | ●Use <db\_name>  ●db.createCollection(“books”,<options>)  //create collection(books), options(optional) |

To check collection list at MongoDB, use ‘show collections’ command.

**<Populate Data>**

-To insert new records/documents for created table/collection.

|  |  |
| --- | --- |
| **Database type** | **query** |
| RDBMS(mysql) | ●INSERT INTO books(id,name,author,view) VALUES(1,”Book1”,”Seungsoo1”,0)  ●INSERT INTO books(id,name,author,view) VALUES(2,”Book2”,”Seungsoo2”,0)  ●INSERT INTO books(id,name,author,view) VALUES(3,”Book3”,”Seungsoo2”,0) //add 3 records into table |
| NoSQL(MongoDB) | ●Db.books.insert([  {“name”: “Book1”, “author”: “Seungsoo1”, “view”: 0},  {“name”: “Book2”, “author”: “Seungsoo2” , “view”: 0},  {“name”: “Book3”, “author”: “Seungsoo2” , “view”: 0},  ]); //add 2 documents on collection |

**<Search Data>**

-display result in conditions from table/collections.

|  |  |
| --- | --- |
| **Database type** | **query** |
| RDBMS(mysql) | ●SELECT \* FROM books //find all records in ‘books’ table  ●SELECT \* FROM books WHERE(name=”Book1”)  //find ‘Book1’ record  ●SELECT \* FROM books WHERE(view<30) //find records with view columns less than 30  ●SELECT \* FROM books WHERE(view=0) //find records with view=0 |
| NoSQL(MongoDB) | ●Db.books.find() //find all documents in collection(books)  ●Db.books.find({“name”: “Book1”}) //find document(Book1)  ●Db.books.find({“view”:{$lte:30}}) //find documents with “view”<30  ●Db.books.find({$where:”this.view.length==0”})  //find empty(length=0) documents by javascript expression |

'$lte’ means ‘less than’ in MongoDB. Comparison operators used in MongoDB is listed as below.

**<operators used in MongoDB query>**

|  |  |  |
| --- | --- | --- |
| Operator | Definition | symbol |
| $eq | equals |  |
| $gt | Greater than |  |
| $gte | Greater than or equals |  |
| $lt | Less than |  |
| $lte | Less than or equals |  |
| $ne | Not equall |  |

**<Update Data>**

-update table/collections by add, delete, or change records/documents.

|  |  |
| --- | --- |
| **Database type** | **<Update Data> query** |
| RDBMS(mysql) | ●UPDATE books SET view=1 WHERE name=”Book1”  //update record ’Book1’ view to 1  ●UPDATE books SET name=”Book2nd” WHERE name=”Book2”  //change record ‘Book2’ name to ‘Book2nd’  ●DELETE FROM books WHERE name=”Book2nd”  //delete ‘Book2nd’ record from table |
| NoSQL(MongoDB) | ●Db.books.update({“name”: “Book1”},  {$set: {“view”: 1}}) //update Books document’s view=1  ●Db.books.update({“name”: “Book2”},  {“name”: ”Book2nd”, ”author”: ”LSS2” })//replace Book2 document  ●Db.books.update({“name”: ”Book2nd”},{$unset: {score: 1}})  //delete Book2nd document from collection |

At MongoDB, ‘db.books.remove({“name”: “Book2nd”})’ command works same as last command above.

Finally result of both queries will be as below.

|  |
| --- |
| Db(Books) |
| Books{[Name(Book1), author(Seungsoo1),view(1)],  [Name(Book3), author(Seungsoo2),view(0)]} |

Comparing MongoDB and mysql, mysql’s grammar looks like acquiring more strict rules than MongoDB for simple works. However for complicated works, mysql query looks more intuitive and applicable for many use-case.