Hope Foundation's

Finolex Academy of Management and Technology

P60, P60-1, MIDC, Mirjole, Ratnagiri, Maharshtra, Pin 415639

Information Technology Department

Remedial Assignment 1 – Apply, Evaluate and Create Level

Big Data Analytics

YEAR / SEM: BE/VIII **Due Date: 28 /02/2020**

BRANCH: IT

Name: Kazi Jawwad A Rahim **Roll No: 28**

Q. N.	Question Details	Marks	Module	Level	CO
Q1.(a)	Create a Hadoop Cluster on four Ubuntu Linux Node and Run a Wordcount Program on the same	[10]	M2	C	CO3
Q1.(b)	Give a Case study for Big Data.	[10]	M1	E	CO811
Q 1.(c)	How Hadoop implements parallelization. Explain with Hadoop high level architecture.	[5]	M3	A	CO813
Q 1.(d)	Explain SQL Join as well as Matrix Multiplication using MapReduce	[5]	M2	AN	CO812
Q 1.(e)	What is DSMS? Compare with DBMS. Explain DSMS Framework	[5]	M2	AN	CO812
Q 1.(f)	Explain NO SQL Data Architectural Pattern	[5]	M2	U	CO812
Q 1.(g)	What is L_r Norm? Explain Euclidean distance as L2 Norm. Prove that the Euclidean distance is a valid distance measure.	[5]	M3	E	CO813
Q 1.(h)	Explain NoSQL Business Drivers	[5]	M3	U	CO813

Ms. Priyanka S. Bandagale

Assistant Professor

Q.1(f)

There are 4 major business drivers for NoSQL as -

1. Volume: In NoSQL volumes specifies the amount of data which is required to be dealt with.
2. Velocity: In NoSQL velocity specifies the rate at which data is stored, retrieved and processed.
3. Variability: In NoSQL variability specifies the type of data supported by NoSQL database viz structured, unstructured and semi-structured.
4. Agility: It is the ability to accept change easily and quickly. Among the variety of agility dimensions such as model agility, operational agility, and programming agility. One of the main ability to quickly and seamlessly scale an application to accomodate large amounts of data, users and connection. The most complex part of building ~~operations~~ applications using RDBMS is the process of data into and getting out of the database. If your data has nested and repeated subgroups of data structures, you need to include an object-relational mapping layer. The responsibility of this layer is to generate the correct combinations of Insert, Update, Delete and Select SQL statement to move object data to and from the RDBMS persistance layer. This process is not simple. and is associated with the largest barriers to rapid change

when developing new or modifying existing applications. Generally, object relational mapping requires experienced software developers. Even with experienced staff, small change requests can cause slowdowns in development and testing schedules. All these hurdles are best overcome by NoSQL database. These databases are schematic and can be scaled down easily. They can accomodate application changes easily and can handle any volume of data efficiently. This agility has become business driver for NoSQL databases.

Q.1(f)

NoSQL data architectural pattern:

NoSQL has mainly 4 architectural patterns as follows—

1) Key-value store — It is a schema-less format. It can be autogenerated or artificially generated while the value can be string, JSON, etc. Key could be a webpage, file path, REST call, Image name, SQL query, etc. The key-value use hash table with unique key and pointer to particular data. A bucket is logical group of keys. So, different buckets can have identical keys.

2) Column Family Store: In column oriented NoSQL db, data is stored in cells grouped in columns rather than as rows of data. Columns are logically grouped into column

family. Instead of storing data in row, for fast query, data is organized for fast column operations. This column centric cube makes column ideal for running aggregate functions or looking up records that match.

3) Document store: It basically expands the idea of key-value store where documents are more complex. In that they contain data and each document assigns a unique key. These are designed for retrieving and managing documents oriented information.

4) Graph Store: It is based on graph theory. These databases are designed for data whose represented as graph and element which are interconnected with undefined no. of relations between them. Data is highly interconnected such as scientific citation, etc. graph store is used.

Q.1(e)

→ DSMS:

Traditional relational DB store and retrieve records of data that is static in nature but it is not feasible to keep transmitting TB of raw data to DBMS. This has resulted in DSMS, which is scalable, flexible, ranging from low latency, adhoc analysis and early reduction on raw data to long term analysis of process data.

	DBMS	DSMS
1)	It stores set of relatively static records with no predefined notion of time.	1) It supports online analysis of rapidly changing data stream.
2)	One-time queries that are evaluated once over point in time snapshot of dataset with answer written to user.	2) Continuous queries evaluated continuously as data stream is continuously arrives.
3)	It has random access to all data of query evaluation of employee of fixed query plan to generate answer.	3) It generate data in sequential pass using limited working memory & adjust the query plan. It may generate approx. or exact answer.
4)	Update rate is relatively low.	4) It is high or busty.
5)	Latency is high.	5) Latency is low.
6)	Arbitary updates can occur in DBMS.	6) Updates are append only.
7)	Good for application that requires persistant storage and complex queries.	7) Good for application where continuous and ordered data stream by arrival time and timestamp and realtime.

Q.1(g)

→ L1 Norm: Also known as manhattan distance or Taxicab norm. L1 norm is the sum of the magnitudes of the vectors in a space. It is the most natural way of measure distance between vectors, that is the sum of absolute

difference of the components of the vectors. In this norm, all the components of the vector are weighted equally.

Euclidean distance or euclidean matric is the "ordinary" straight-line distance between two points in Euclidean space. With this distance, euclidean space becomes a metric space. The associated norm is called the euclidean norm. Older literature refers to the metric as the Pythagorean metric. A generalised term for the Euclidean norm is the L^2 norm or L^2 distance.

Q.1(d) →

Join using MapReduce:

The joins can be done at both Map side and Join side according to nature of data sets to be joined.

Reduced Side join:

Lets take following tables containing employee and department data.

Employees

Name	Age	Dept_id
Alex	26	2
Ben	24	2
Sara	34	5

Department

Dept_id	Name
5	MKT
2	Eng
3	Sales

Lets see how join query below can be achieved using reduce side join.

```
SELECT Employees.Name, Employees.Age,
       Dempartment.Name
```

```
FROM Employees
INNER JOIN Department ON
 Employees.Dept_id = Department.Dept_id
```

Map side is responsible for emitting the join predicate values along with the corresponding record from each table so that records having same department id in both tables will end up at on same reducer which would then do the joining of records having same department id. However, it is also required to tag the each record to indicate from which table the record originated so that joining happens between records of two tables.

Here is the pseudo code for map function for this scenario.

```
map(K table, V rec){
  dept_id = rec.Dept_id;
  tagged_rec.tag = table
  tagged_rec.rec = rec
  emit(dept_id, tagged_rec)
}
```

At reduce side join happens within records having different tags.

```
reduce(K dept_id, list<tagged_rec> tagged_recs){
 for(tagged_rec : tagged_recs){
    for(tagged_rec1 : tagged_recs){
       if(tagged_rec.tag != tagged_rec1.tag){
          joined_rec = join(tagged_rec, tagged_rec1)
       }
```

```
emit(tagged_rec.rec_Dept_Id, joined_rec)
}
}
```

Map side Join (Replicated Join):

Using distributed cache on smaller table - For this implementation to work one relation, has to fit in to memory. The smaller table is replicated to each node and loaded to the memory. The join happens at map side without reducers involvement which significantly speeds up the process since this avoid shuffling all data across the network even though most of the records not matching are later dropped. Smaller table can be populated to a hash-table so look-up by Dept_id can be done. The pseudo code is outlined below -

```
map(K table, V rec){
    list recs = lookup(rec.Dept_id)
    for(small_table_rec : recs){
        joined_rec = join(small_table_rec, rec)
    }
    emit(rec.Dept_id, joined_rec)
}
```

Q.1(c)

→ Parallelization in Hadoop:

Parallel map over input: Input data is processed such that key/value pairs are processed one by one. It is well known that this pattern of a list map is amenable to total data

parallelism. That is, in principle, the list map may be executed in parallel at granularity level of single elements. Clearly, MAP must be a pure function so that the order of processing key/value pairs does not affect the result of the map phase and communication between the different threads can be avoided.

Parallel reduction per group:

Let us assume that REDUCE defines a proper reduction. That is REDUCE reveals itself as an operation that collapses a list into a single value by means of an associative operation and its unit. Then, each application of REDUCE can be associative operation at the nodes. If the binary operation is also commutative, then the order of reduction at the granularity of groups, it is non-obvious that parallel reduction of the values per key could be attractive.

Q.1(a)

→ Steps to create hadoop cluster on 4 Ubuntu nodes-

1) Name the 4 nodes as node-master, node1, node2 and node3 respectively.
2) Add a private IP address to each Linode.
3) Adjust IP address of each nodes as-

node-master:	192.0.2.1
node1:	192.0.2.2
node2:	192.0.2.3
node3:	192.0.2.4

4) Create host file on each node
5) Distribute authentication key-pairs for the Hadoop user.
6) Download and unpack hadoop binaries.
7) set environment variables.
8) Configure the master node.
9) set NameNode Location
10) set path for HDFS
11) set YARN as Job Scheduler
12) Configure YARN
13) Configure Workers
14) Configure Memory Allocation
15) Duplicate Config Files on Each Node
16) Format HDFS
17) Run and monitor HDFS
18) RUN YARN
19) Run the code for word count.

pseudo code for word count using Map Reduce:

Word Count using MapReduce
→ count occurances of each word input

M = large corpus of text

Mapper:
each word in M → ("word", 1)

Reducers:
for all ("word", v-i)
→ ("word", sum-vi-i)
"aggregate" in Hadoop