Aakrist Goyal
2201331540002

(Big Data Analytics Assignment)

Ques(1): List down the tools related with hadoop?

→ Hadoop has several ecosystem tools that ehance its functionality:

HDFS (Hadoop Distributed File system): storage layer of hadoop.
MapReduce: Data processing framework
YARN (Yet Another Resource Negotiator): Manages cluster resources
Apache Hive: SQL-like querying for Big data
Apache Pig: scripting tool for data processing
Apache HBase: NoSQL database that runs on hadoop.
Apache spark: fast data processing engine.
Apache sqoop: transfer data between hadoop and relational databases
Apache flume: collects and transfers large amount of log data.
Apache oozie: workflow scheduler for hadoop tasks.

Ques(2):- Explain the Anatomy of MapReduce job run?

→ the Execution of a MapReduce job follows through several phase:

- client job submission: the user submits the MapReduce job using the hadoop command inline interface.
- Job tracker assigns tasks: YARN schedules the job and assigns Mapper and Reducer tasks to worker Node.
- Data processing in mapper phase: input data is split and proceeded in parallel.
- shuffling and sorting: Intermediate results are grouped by key
- Reducer phase: the final aggregation takes place.
- Results are stored in HDFS: the output is saved in hadoop's distributed storage.

Ques(3): Describe a Case study and full architecture of Map reduce functioning?

→ Case study : Analyzing Social Media sentiments

A Company wants to analyze twitter data to determine positive and negative sentiment.

The workflow will be that the Mapper will Extract words (Ex! Happy or sad) from tweets and assigns the sentiment score and then the reducer aggregates scores for Each keyword.

Full Architecture:

- Input splitting : stores the log data, split the dataset into smaller chunks
- Map phase : processes the subset of data . It Extracts the Ip addresses and emits them as key value pairs.
- Shuffle and Sort phase : the framework groups all key value pairs by key. Sorting ensures that all occurences of an IP address are sent to the same reducer
- Reducer phase : sums up all the values for Each IP address to get final count.
- final Output : that is stored in HDFS.

Ques(4):- List all the differences between regular file systems and HDFS?

	Regular file systems	HDFS
storage type :	Local hard drives	Distributed across a cluster
fault tolerance :	No automatic replication	Data is replicated
Data processing :	Limited to single Machine	Parallel processing using MapReduce
scalability :	Dificult to scale	Easily scale by adding Nodes
write Mechanism :	supports Modification	write-once, read many models
Data access speed :	fast for small files	optimized for large files
use case :	Regular Applications	Big data processing

Aakrist Goyal

2201331540002

Ques (5) :- Describe the working of MapReduce with suitable Examples and also present the Example of word count program in hadoop and Explain precisely (1) Mapper code ; (2) Reducer code ; (3) driver code .

→ MapReduce is a distributed data processing model used in hadoop to process large datasets in paralled across multiple Nodes in a cluster. It consists of 2 Main phases i.e.- Map Phase and Reduce phase.

Map phase processes input data and converts it into key-value pair and Reduce phase aggregates the key value pairs to generate the final result.

Hadoop word Count program :

Let's consider a text file containing the lines like Hadoop is powerful ; Hadoop is scalable ; Hadoop is fast.

- Mapper phase splits the lines into words and Each words is emitted as a key with value 1

  Ex:- (Hadoop, 1) ; (is, 1) ; (Powerful, 1)

- shuffle & sort phase groups by key before being sent to reducers

  Ex:- (Hadoop) → (1, 1, 1)
  (is) → (1, 1, 1)
  (powerful) → (1)
  (scalable) → (1)
  (fast) → (1)

- Reducer phase sums up the values for Each word.

  Ex:- (Hdoop → 3) ; (is → 3) ; --etc.

Mapper code :

```
Public class WordCountMapper Extends Mapper {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(LongWritable key, Text value, Context context)
                throws IoException, InterruptedException {
        string line = Value.toString();
        stringTokenizer = new stringtokenizer(line);
        while (tokenizer.has more tokens ()) {
                word.set (tokenizer.next token ());
                context.write (word, one);
    } } }
```

Reducer Code:

```
Public class WordCountReducer Extends Reducer {
        Private IntWritable result = new IntWritable ();
        Public void reduce (Text key, Iterable <> values, Context context)
              throws IoException {
        Int sum = 0;
        for (IntWritable val : values) {
              sum += val.get ();
        }
        result.set (sum)
        context.write (key, result);
  } }
```