# Yuchen Peng

1. **What are the Hadoop daemons? Explain their roles. (10’)**
   - **NameNode** – stores meta-data for HDFS.
   - **Secondary NameNode** - While secondary name node is standby, it keeps merging edits log to fsimage. If active namenode was down, secondary namenode immediately be switched to take the responsibility of an active node. Secondary share same storage with active namenode such that secondary name node solves the problem of single point of failure.
   - **DataNode** – Stores actual HDFS data blocks
   - **Job Tracter** – Manages MapReduce jobs, distribute individual tasks to machines running the Task Tracker.
   - **Task Tacker** – Each DataNode will have one task tracker. Task trackers communicate with Job trackers to send status of the jobs.

2. **What are the different complex data types in Pig? (10’)**
   - **tuple** – An ordered set of fields
   - **bag** – An collection of tuples
   - **map** – A set pf key value pairs.

3. **What are managed and external tables in Hive? (10’)**
   - **Managed table** - Managed table is also called as Internal table. This is the default table in Hive. When we create a table in Hive without specifying it as external, by default we will get a Managed table. If we create a table as a managed table, the table will be created in a specific location in HDFS.
   - **External table** - External table is created for external use as when the data is used outside Hive. Whenever we want to delete the table’s meta data and we want to keep the table’s data as it is, we use External table. External table only deletes the schema of the table.

4. **What are benefits of Spark over MapReduce? (10’)**
   - Spark is easy to program and does not require any abstractions.
   - Programmers can perform streaming, batch processing and machine learning ,all in the same cluster.
   - It has interactive mode whereas in MapReduce there is no built-in interactive mode.
   - park executes batch processing jobs about 10 to 100 times faster than Hadoop MapReduce.
   - Spark uses lower latency by caching partial/complete results across distributed nodes whereas MapReduce is completely disk-based.
   - Spark uses an abstraction called RDD which makes Spark feature rich, whereas MapReduce doesn't have any abstraction

5. **What are transformations and actions in the context of RDDs? (10')**
   - **Spark Transformation** is a function that produces new RDD from the existing RDDs. After the transformation, the resultant RDD is always different from its parent RDD. It can be smaller (e.g. filter, count, distinct, sample), bigger (e.g. flatMap(), union(), Cartesian()) or the same size (e.g. map).
   - **Transformations create RDDs** from each other, but when we want to work with the actual dataset, at that point action is performed. When the action is triggered after the result, new RDD is not formed like transformation. Thus, Actions are Spark RDD operations that give non-RDD values. The values of action are stored to drivers or to the external storage system.

6. **What are the components of Presto Architecture? (10')**
   - **Coordinator** - The Presto coordinator is the server that is responsible for parsing statements, planning queries, and managing Presto worker nodes.
   - **Worker** - A Presto worker is a server in a Presto installation which is responsible for executing tasks and processing data.

**7.**

**(1)** **bookMapper**'s job is to process the input data. The input file books.txt is passed to the bookMapper function line by line. bookMapper processes the data and creates several small chunks of data. Each chunk of data has a value of 1.

**(2)** The functionality of the **bookReducer** is a combination of the shuffle stage. It processes the data that comes from the bookMapper. For each chunk of data, bookReducer accumulate and combine the values from each unique key. After processing, it produces a new set of output, which will be stored in the HDFS.

**(3)** **pig**

a = load 'books' as (ISBN:int, Book-Title:chararray, Year-Of-Publication:int, Publisher:chararray, image-URL-S:chararray, image-URL-m=M:chararray, image-URL-L:chararray);

b = group a by Year-Of-Publication;

c = foreach b generate COUNT(a);

**(4)** **Hive**

CREATE TABLE books(ISBN int, Book-Title chararray, Year-Of-Publication int, Publisher chararray, image-URL-S chararray, image-URL-m=M chararray, image-URL-L chararray);

SELECT Year-Of-Publication, COUNT(*) FROM books GROUP BY Year-Of-Publication;

**8.**

**(1)** This is scala code statement. peopleDF was defined previously val peopleDF = sqlCtx.createDataFrame(rowRDD, schema). peopleDF was filtered by gender that equals to “F” then followed by the another filter that filtering height > 180, and eventually show 50 results. In a word, the output is to select all the female people whose height are larger than 180cm.

**(2)**

```
var people = spark.sql(“SELECT * from peopleData”)

var grouped_gender = people.groupBy(“gender”).count().show()
```

**(3)**

```
var people = spark.sql(“SELECT * from peopleData”)

var highest_Female = people.select(“gender”).where(“gender =
F”).orderBy(desc(“gender”)).show(1)
```

**(4)**

```
var people = spark.sql(“SELECT * from peopleData”)

var average_male_hight = people.select(“gender”).where(“gender =
F”).agg(avg(“height”)).show()
```

**(5)**

```
as.data.frame.matrix(peopleDataRDD)
```