**Q1:** Draw a figure showing the differences between shared-nothing, shared-memory, and shared disk architectures.



**Q2:** The figure below shows the matrix multiplication problem covered in class. Assume each
matrix (M and N) is stored in a separate file (with the same matrix name). Each line in the files is in the format (*i, j, v*) where “i" is the row number, “j” is the column number, and “v” is the cell value. Assume we want to solve the problem using Hadoop

(a) **[1 Point]** How many mapreduce jobs are needed?

2

(b) **[4 Points]** Describe the map input, map output, reduce input, reduce output of the job(s).

The first job is to Join M and N, and join key is column#j in M and row#j in N:

Map function:

From M, Key-j , Value-(flagM, row#i, Mij)

From N, Key-j, Value-(flagN, column#k, Njk)

 Reduce function (For each j)

 compute v = Mij \* Njk

Output (i, k, v)

The second job is to aggregate:

 Map function(i,k,v):

 Output Key-(i,k), Value-v

 Reduce function((i,k),v-list):

 Sum (i,k, sum(v-list))

**Q3:** In Spark, what is the difference between Action operations and Transformation operations?
Give two examples of each.

Action: Perform a computation on existing RDDs producing a result. Examples: count, collect, reduce, save. 🡺 Action operation triggers execution

Transformation: Operations on existing RDDs that can return a new RDD. Examples: map, filter, join. 🡺 Transformation operations just prepare things without triggering the execution

**Q4:** In Spark, draw a figure that shows the difference between a Narrow Dependency and a Wide Dependency? And Then, Give one example operation in each case.

. Example: filter, map. . Example: join, grouping.

**Q5:** Given the following operations/algorithms:
*K-Means Clustering, Logistic Regression, Aggregation operation, Selection operation, Building the model of Naïve Bayes Classifier, Page Rank*State which one(s) will have similar performance (and also mention why) if executed on either
Hadoop or Spark infrastructures. Assume the input data in all cases is initially stored on HDFS.

Aggregation operation, Selection operation and Building the model of Naive Bayes Classifier.

Reason: They only need one iteration. Hadoop consumes extra time of loading and reading data for every iteration in multiple iteration tasks (e.g. page rank, K-means, logistic regression), but spark has build-in mechanism to cache and reuse data for iterations. So hadoop and spark will have similar execution time for one iteration tasks while spark will be much better on multiple iteration tasks.

**Q6:** Specify the data model of each of the following databases?



Redis: Key-value Store

Hbase: Column Family

Cassandra: Column Family

mongoDB: Document Store

**Q7:** In NoSQL Databases, we studied two concepts CAP & BASE.
**(a) [3 Points]** The following figure shows the CAP theorem. Put the following databases on the triangle edge matching with the DB characteristics.

*Relational DB, MongoDB, HBase, Cassandra, CouchDB*



Relational DB: Consistency and Availability;

MongoDB: Consistency and Partition Tolerance;

HBase: Availability and Partition Tolerance;

Cassandra: Availability and Partition Tolerance;

CouchDB: Availability and Partition Tolerance;

**(b) [2 Points]** The keyword “BASE” is an abbreviation of what?

Basically Available – all data is distributed, even when there is a failure the system continues to work.

Soft state – there is no consistency guarantee

Eventually consistent – system guarantees that even when data is not consistent, eventually it will be.

**Q8:** MongoDB has the concept of a “ReplicaSet”, which is a group of machines controlled by one primary plus multiple secondaries. What does each of the following properties mean? And give a scenario in which such property setting is useful?
(a) Priority = 0

Meaning: Cannot be elected as primary. Cannot accept write. Still has data & accept reads.

Scenario: May want some data centers not to accept write ops.

(b) Hidden = True

Meaning: Cannot be elected as primary. Imply Priority = 0. But also cannot accept reads from clients.

Scenario: Good for dedicated offline tasks, e.g., reporting.

(c) SlaveDelay = 100

Meaning: Waits 100 msec before getting the updates from the primary site. Should be Hidden = True.

Scenario: Good to recover from bad transactions.

**Q9: (a) [2 Points]** Give two reasons on why Hadoop and Spark are not considered as database
engines unlike RDBMS, MongoDB, HBase?

**This is because an engines qualifies as a “Database” if it has a data model & allows record-level operations, e.g., insertion, deletion, and updates. Hadoop & Spark lack these two features.**

**(b) [3 Points]** Which of the following systems rely on Zookeeper? And state why it does (or does not) rely on Zookeeper.
Relational DB, MongoDB, HBase, Cassandra, CouchDB
 (b) Relational DB does not rely on zookeeper, since it already has built-in Concurrency Control.

 **MongoDB, Cassandra, CouchDB also do not need zookeeper as they have a built-in concurrency control component.**

**HBase needs a Zookeeper since it is a distributed system that needs some mechanisms for coordination. And HBase does not have its own concurrency control component.**

**Q10:** In Apache Hive, answer the following questions:
**(a) [3 Points]** The following Hive statement creates a partitioned table “R”. Draw a figure that
shows how R will be stored in HDFS (that is, the structure that hive will maintain to store R)
Create Table R(a int, b int, c string) Partitioned By
(country string, date string);

A table in Hive is an HDFS directory in Hadoop. Partitioned tables have “sub-directories”, one for each partition.

***This is not correct. It should be a 2-level hierarchy. The first level is based on “Country” and under each country there are sub directories based on “date”…..Fix the figure accordingly***



**(b) [2 Points]** Apache Hive has a unique component in its architecture that does not exist in Apache Pig. What is this component and why it exists in Hive alone?

**The compoenent is the “Metastore”. It stores in it the schema of the tables and the partitioning information. This is not needed in Pig since it does not have table definitions or schema.**

**Q11: [This problem is given to you in HW3 (Spark Project)]**Assume a two-dimensional space that extends from 1…10,000 in each dimension as shown in the figure below. There are points scattered all around the space. The space is divided into pre-defined grid-cells, each of size 20x20. That is, there is 500,000 grid cell in the space. Each cell has a unique ID as indicated in the Figure. Given an ID of a grid cell, you can calculate the row and the column it belongs to using a simple mathematical equation.


**Neighbor Definition:** For a given grid cell X, N(X) is the set of all neighbor cells of X, which are the cells with which X has a common edge or corner. The Figure illustrates different examples of neighbors. Each non-boundary grid cell has 8 neighbors. However, boundary cells will have less number of neighbors (See the figure). Since the grid cell size is fixed, the IDs of the neighbor cells of a given cell can be computed using a formula (mathematical equations) in a short procedure.
Example: N(Cell 1) = {Cell 2, Cell 501, Cell 502}
N(Cell 1002) = {Cell 501, Cell 502, Cell 503, Cell 1001, Cell 1003, Cell 1501, Cell1502, Cell 1503}

**Relative-Density Index:** For a given grid cell X, I(X) is a decimal number that indicates the
relative density of cell X compared to its neighbors. It is calculated as follows.

I(X) = X.count / Average (Y1.count, Y2.count, …Yn.count) Where “X.count” means the count of points inside grid cell X, and {Y1, Y2, …, Yn} are the neighbors of X. That is N(X) = {Y1, Y2, …, Yn}. If “Average (Y1.count, Y2.count, …Yn.count)” =0, then I(X) should be set to zero.

***The question is:*** Assume we are using Spark RDDs. The first initial RDD has lines, where each line is one point in the space represented as (x,y). As discussed in class, draw a diagram showing the sequence of RDDs that need to be generated to find the TOP 50 I(X) grid cells in a scalable way. For each transformation (E.g., from RDDi to RDDi+1) indicate the output of both the map and the reduce functions (assuming each transformation is done in the map-reduce style).

1. RDD1 (x, y) → RDD2 (cellID, count)

map output: (cellID, 1)

reduce output: (cellID, count)

1. RDD2 → RDD3 (cellID, count, neighborCellIDList)

 map output: (cellID, count, neighborCellIDList)

1. RDD3 → RDD4 (cellID, neighborCellIDAndCountList)

 map output: (cellID, count, neighborCellIDAndCountList)

1. RDD4 → RDD5 (cellID, RDI)

 map output: (cellID, RDI)

1. RDD5 → Top50