CSE 490H: Scalable Systems: Design, Implementation and Use Of Large Scale Clusters

Autumn Quarter 2008

Term Test

Your name: ____

Question 1	**_____ / 6**
Question 2	**_____ / 6**
Question 3	**_____ / 9**
Question 4	**_____ / 6**
Question 5	**_____ / 6**
Question 6	**_____ / 4**
Question 7	**_____ / 8**
Question 8	**_____ / 16**
Question 9	**_____ / 9**
Question 10	**_____ / 6**
Question 11	**_____ / 10**
Question 12	**_____ / 14**
Total	**_____ /100**

Question 1 (6 points)

Google has built a fault-tolerant, highly-available, recoverable, scalable search application using software techniques.

A (3 points): Identify a situation in which the Google search application uses data partitioning to achieve scalability.

B (3 points): Identify a situation in which the Google search application uses replication to achieve reliability.

Question 2 (6 points)

Describe two ways in which BigTable has less functionality than a traditional relational database system.

Question 3 (9 points)

A (3 points): What does the IP network protocol accomplish – what does it do? (One sentence)

B (3 points): What does the TCP network protocol accomplish – what does it do? (One sentence)

C (3 points): Are TCP packets encapsulated within IP packets, or are IP packets encapsulated within TCP packets?

Question 4 (6 points)

A Google File System cluster has a single Master, which holds metadata, and a large number of Chunkservers, which hold file data. GFS uses the Chubby coarse-grained lock service to elect a new Master in the event of a failure.

A (3 points): How is Chubby used for this purpose? That is, how do GFS computers determine a new Master using Chubby? (Just a sentence or two.)

B (3 points): Like GFS, Chubby uses replication. In the event of the failure of the Master in a Chubby cell, how is a new Master determined? (Just a sentence or two.)

Question 5 (6 points)

You don't "need" MapReduce (and the components on which it is built, such as GFS and Chubby) to build applications on top of a huge cluster of commodity computers. MapReduce eases the task, though, by taking care of a large number of headaches that you would otherwise have to write code to deal with yourself. Identify two different major headaches that MapReduce takes care of.

Question 6 (4 points)

The major components of a computer are CPU, RAM, network, and disk. Which component(s) are typically the rate-limiting ones in a MapReduce process? Explain why. (Just a few sentences.)

Question 7 (8 points)

Given the following web link graph:

```
A -> [B, C]
B -> [A, C]
C -> [D, B, A]
D -> [A]
```

where `X -> [Y, Z]` means page `X` links to pages `Y` and `Z`, show one iteration of the PageRank algorithm. Assume d = .85. Before iteration 1, initialize each pagerank to 0.15.

Question 8 (16 points)

Given these input data structures:

```
class Foo implements Writable {
   int fooIdentificationKey;
   int someFooData;
   float importantFooMagic;

   void write(DataOutput out) { } // elided
   void readFields(DataInput in) { } // elided
}

class Bar implements Writable {
   int barIdentificationKey;
   String barString;
   int relatedFooItem;

   void write(DataOutput out) { } // elided
   void readFields(DataInput in) { } // elided
}
```

A (4 points): Create a datatype that has the following properties:

- It can represent the contents of either a `Foo` or a `Bar` object.
- A `Bar` object should be able to be joined with the `importantFooMagic` field of the corresponding `Foo` object it references.
- We must be able to distinguish between `Bar` objects that have been through this join process and those that have not.

Show all the fields the object requires; also show the `write()` method body. (You do *not* need to show the `readFields()`, `compareTo()`, `equals()`, `toString()`, or `hashCode()` methods.) For reference, assume the following interface:

```
interface DataOutput {
   public void writeInt(int x);
   public void writeLong(long x);
   public void writeFloat(float x);
   public void writeDouble(double x);
   public void writeString(String x);
   public void writeBoolean(bool x);
   public void writeChar(char x);
}
```

B (6 points): Write the mapper and reducer code which reads in objects of your combined data type, and emits them back out; `Foo` objects should be unchanged, but `Bar` objects should have had the magic data from their related `Foo` objects joined in.

Assume that `Foo`-style values (magically) always arrive "first in line" at a reducer ahead of any `Bar`-style values.

Assume that the key arriving at the mapper is irrelevant.

C (3 points): Why is it important for the `Foo`-style values to arrive at the reducers before the `Bar`-style values?

D (3 points): What is the general relationship (the "contract") between the implementations of the `compareTo`, `equals`, and `hashCode` methods? Why is this important for MapReduce? (Just a few sentences.)

Question 9 (9 points)

A (6 points): Why is data not lost when a single machine fails in an HDFS cluster? Describe the steps the system takes to ensure this.

B (3 points): Under what conditions could HDFS lose data permanently?

Question 10 (6 points)

A (3 points): Assuming a Paxos cluster of 7 nodes, at most how many nodes can fail and leave the system remaining consistent (functioning correctly)?

B (3 points): Why can Paxos not support more failures than this?

Question 11 (10 points)

Virtual machine monitors have recently found a “new life” for server consolidation (multiple services on a single server).

A (2 points): Identify one key characteristic of VMM’s that makes them particularly suitable for this task.

B (8 points): Trace the steps that occur when an application running on a guest operating system in a virtual machine attempts to do a file operation – identify each transition among application, guest OS, VMM, and hardware, and identify the mechanism that causes each transition.

Question 12 (14 points)

Implement Variance(X) using MapReduce.

The Variance of n values of the variable X is defined as

$$\text{Variance(X)} = \sum_{i=1}^{n} (x_i - \mu)^2$$

where μ is the arithmetic mean of the values.

The input to your program is a file including several intermixed datasets. A dataset is the multiple values for a single variable. Each line in the file consists of a key (the name of the variable) and a single value. The same values may repeat within a dataset. Thus, the input file looks like:

```
K1   Value1_for_K1
K1   Value2_for_K1
K2   Value1_for_K2
K1   Value3_for_K1
K2   Value2_for_K2
Etc.
```

The output of your program should have a (Key, Variance) pair for each key (each variable) in the input dataset.

What are the scalability limits, if any, of your solution?