Zayd Hammoudeh

24/25 +3

CS286 Exam I

1. What are the *3 V's* of big data?
   ~~a.~~ Value, velocity, volume
   ~~b.~~ Velocity, volume, veracity
   [✓ c.] Velocity, volume, variety
   ~~d.~~ Veracity, value, volume

C

2.* Which of the following is considered *unstructured* data?
   [✓ a.] video
   ~~b.~~ email
   c. log
   ~~d.~~ table

A

3. Which of the following scaling strategies introduces *error*?
   ~~a.~~ Scale-up
   ~~b.~~ Scale-out
   ~~c.~~ parallelizing
   [✓ d.] sampling

D

4. The `reduce()` method *reads one*
   ~~a.~~ Split at a time
   ~~b.~~ Block at a time
   [✓ c.] Key-value pair at a time
   ~~d.~~ Spill at a time

C

5. The combiner
   ~~a.~~ Always improves performance
   ~~b.~~ Is called before the `map()` method
   ~~c.~~ Is called after the `reduce()` method
   [✓ d.] Is an optional optimization

D

6. Which statement is *true* about MapReduce? may
   a. MapReduce was invented at Google
   [✓ b.] MapReduce programs must ~~be~~ written in Java
   c. MapReduce programs are scale-dependent
   ~~d.~~ MapReduce is not supported by YARN

B

7.* Which of the following *extends* `InputFormat` class?
   ~~a.~~ `LineRecordReader`
   b. `FileInputFormat` ☆
   [✓ c.] `TextInputFormat`
   ~~d.~~ `InputSplit`

C

7

8. The *number of mappers* that run for a job

D

a. Is equal to the number of blocks
b. Is equal to the 1 by default
c. Is equal to the number of reducers
[✓ d.] Is equal to the number of input splits

9. The *number of reducers* that run for a job

C

a. Depends on the number of input splits
b. Is equal to the number of mappers
[✓ c.] Is 1 by default
d. Depends on the amount of RAM installed

10. Which of the following are ***NOT*** part of the Hadoop strategy?

B

a. Distribute data
[✓ b.] Synchronize data
c. Tolerate failure
d. Distribute computation

11. What does a *virtual file system* provide?

D

~~a.~~ Read-only access
~~b.~~ Append-only access
c. Read-write access
[✓ d.] POSIX compliance

12. What is included as part of the output from *every* MapReduce job?

B ☆

~~a.~~ `_SUCCCESS` file
[✓ b.] `_logs` directory
c. `part-r-xxxxx` files
d. `part-m-xxxxx` files

13. Which statement is *true* of the `Mapper` class?

A

[✓ a.] The output of the Mapper must match the input of the Reducer.
b. The input of the Mapper must match the input of the Reducer.
c. The input of the Mapper must match the output of the Reducer.
d. The output of the Mapper must match the output of the Reducer.

14. Which statement is *true* of the `Reducer` class?

D

~~a.~~ The Reducer object calls the `reduce()` method once per split.
b. The Reducer object calls the `run()` method once per key.
c. The Reducer object calls the method once per key
[✓ d.] The Reducer object calls the `reduce()` ~~`cleanup()`~~ method once per key.

Cleanup

7

Zayd Hammoudeh

15. Which statement is true of the *driver* class?

D

~~a.~~ The driver must check the command-line syntax.
~~b.~~ The driver must parse the command-line arguments.
c. The driver must use the ToolRunner interface.
[✓ d.] The driver defines the Mapper and Reducer classes.

16. Which Java statement converts a `Text` parameter (`value`) to a *list* of tokens?

B

a. `new StringTokenizer(value, "\\s+");`
[✓ b.] `new String(value, "\\s+");`
c. `value.toString();`
d. `new String(value);`

17. How many total copies of a block are created *by default* in HDFS and MapR-FS?

C

a. 1
b. 2
[✓ c.] 3
d. HDFS and MapR-FS do not have the same default number of replicas

18. Which Java statement launches a MapReduce job *synchronously*?

A

[✓ a.] `return job.waitForCompletion(true) ? 0:1;`
b. `return job.waitForCompletion(false) ? 0:1;`
c. `job.submit(false);`
d. `job.submit(true);`

19. Which of the following is *different* between `mapred` and `mapreduce` packages?

B

~~a.~~ YARN support
[✓ b.] `map()` method signature
c. `reduce()` method signature
~~d.~~ Number of slots or containers required

20. Which statement is *true* of the Hadoop MapReduce data types?

C

a. Values must implement the `WritableComparable`
b. The `Text` class implements the `Serializable` interface
[✓ c.] Keys must implement the `WritableComparable` interface
d. The `BooleanWritable` implements the `Interruptable` interface

21. Which of the following is ***NOT*** a feature of HDFS?

X A

[✓ a.] Compression
b. Replication
c. Authorization
d. Encryption

6

22. How many *mappers* will the following configuration instantiate at the job runtime? : 4 files (10KB, 250MB, 600MB, 1000MB); block size = 256MB

↑ ↑ ↑ ↑
1 1 3 4

B
- a. 8
- [✓ b.] 9
- c. 4
- d. 5

23. Which of the following statements is *true* regarding how records are read by the mappers?

B
- ~~a.~~ There is one mapper instantiated per record
- [✓ b.] There is one `map()` method called per record
- ~~c.~~ Records must fall on input split boundaries
- ~~d.~~ End users must define the record terminator

24. Which of the following is a counter that the Hadoop framework will track by default?

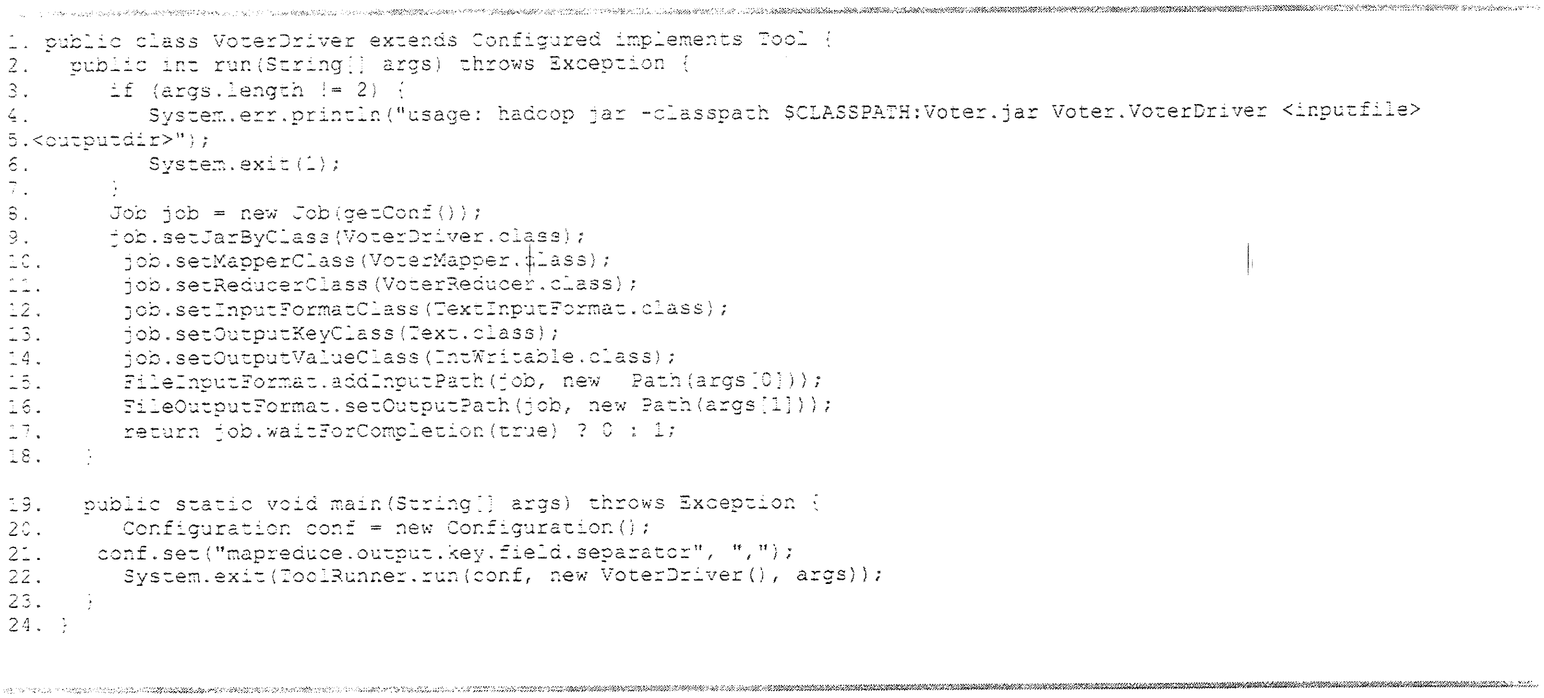
B
- a. Number of RPC packets sent
- [✓ b.] Physical memory consumption
- c. Number of RPC packets received
- d. Number of CPU pipeline stalls

25. Which statement is ~~true~~ False regarding job priority?

D
- ~~a.~~ Job priority determines how quickly a job is scheduled
- ~~b.~~ You can modify a job priority in the driver class
- ~~c.~~ You can modify a job priority when you submit the job
- [✓ d.] You cannot modify a job priority in Hadoop

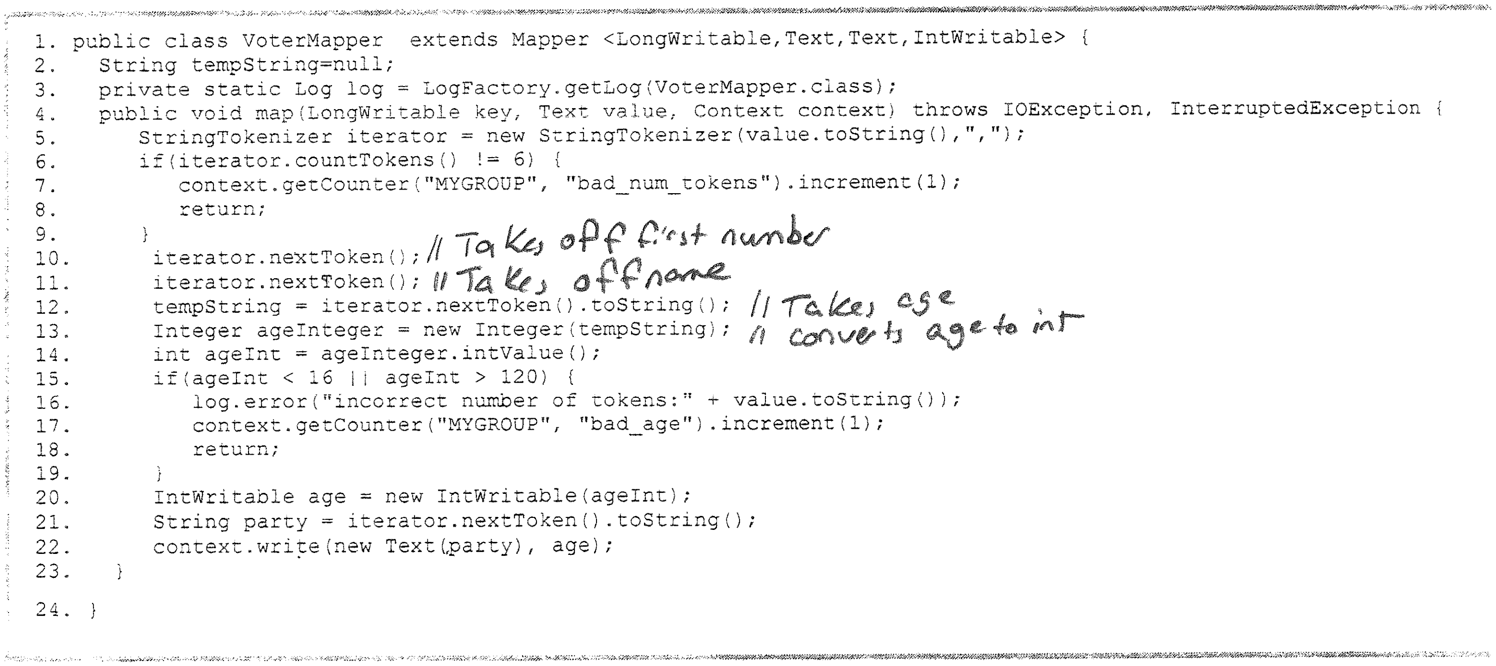
4

Zayd Hammoudeh



```
public class VoterDriver extends Configured implements Tool {
  public int run(String[] args) throws Exception {
     if (args.length != 2) {
        System.err.println("usage: hadoop jar -classpath $CLASSPATH:Voter.jar Voter.VoterDriver <inputfile>
<outputdir>");
        System.exit(1);
     }
     Job job = new Job(getConf());
     job.setJarByClass(VoterDriver.class);
      job.setMapperClass(VoterMapper.class);
      job.setReducerClass(VoterReducer.class);
      job.setInputFormatClass(TextInputFormat.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new  Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      return job.waitForCompletion(true) ? 0 : 1;
   }

   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
    conf.set("mapreduce.output.key.field.separator", ",");
      System.exit(ToolRunner.run(conf, new VoterDriver(), args));
   }
}
```



```
public class VoterMapper  extends Mapper <LongWritable,Text,Text,IntWritable> {
  String tempString=null;
  private static Log log = LogFactory.getLog(VoterMapper.class);
  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
     StringTokenizer iterator = new StringTokenizer(value.toString(),",");
     if(iterator.countTokens() != 6) {
        context.getCounter("MYGROUP", "bad_num_tokens").increment(1);
        return;
     }
      iterator.nextToken(); // Takes off first number
      iterator.nextToken(); // Takes off name
      tempString = iterator.nextToken().toString(); // Takes age
      Integer ageInteger = new Integer(tempString); // converts age to int
      int ageInt = ageInteger.intValue();
      if(ageInt < 16 || ageInt > 120) {
         log.error("incorrect number of tokens:" + value.toString());
         context.getCounter("MYGROUP", "bad_age").increment(1);
         return;
      }
      IntWritable age = new IntWritable(ageInt);
      String party = iterator.nextToken().toString();
      context.write(new Text(party), age);
  }
}
```

```
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
     long sum = 0L;
     int count = 0;
     int tempValue = 0;
     int max=Integer.MIN_VALUE;
     int min=Integer.MAX_VALUE;
     for (IntWritable value: values) {
         tempAge = value;
          tempValue = (new Integer(value.toString())).intValue();
        if(tempValue < min) {
             min=tempValue;
          }
          if(tempValue > max) {
            max=tempValue;
          }
          sum+=tempValue;
          count++;
      }
      float mean=sum/count;
   context.write(key, new FloatWritable(mean));
   }
}
```

missing final field

```
999991,mike falkner,38,socialist,703.09
999992,luke falkner,61,libertarian,965.43,15683
999993,calvin xylophone,20,democrat,794.84,21569
999994,wendy underhill,18,democrat,343.46,4758
999995,nick  young,57,democrat,324.15,3956
999996,ethan brown,37,democrat,117.57,19228
999997,ulysses nixon,55,independent,819.34,27477
999998,calvin laertes,22,democrat,542.47,16730
999999,irene thompson,70,green,158.22,10006
1000000,priscilla zipper,13,libertarian,862.71,7137
```

Bad

Socialist.
liberterian.
democrat
Independent
green.

Zayd Hammadeh

Based on the sample data, `VoterDriver`, `VoterMapper`, and `VoterReducer` classes provided above, write a short 1-2 sentence answer for each question below.

1. What is the record delimiter for the mapper?
   "\n" (I do not see you changing the default).
2. What is the field delimiter for the mapper?
   Comma "," (from StringTokenizer declaration - line #5)
3. What is the output key type and output value type in the mapper?
   Key: Text Value: IntWritable
4. What is the input key type and input value type in the reducer?
   Key: Text Value: IntWritable for each value. Collection is Iterable<IntWritable>
5. Assuming there is only one reducer and based strictly on the data provided, since using MapReduce package
   +1 how many calls to the `reduce()` method would be made?
   Four (socialist is excluded as a bad record)
6. What would be the output from the `map()` method for the following data point?

+1 `999991,mike falkner,38,socialist,703.09,20560`

Key value pair. key is "socialist" (type Text) and value is 38 (type int writable)

7. What would be the output from the `reduce()` method for the key
   +1 `democrat`?
   22 + 37+57+18+20 = 97+57=154/5=30 (integer division is what I think Java uses)
   Key value pair: Key: "democrat" (Text) Value: 30 (FloatWritable)
   "democrat, 30.0" to file
8. In the `VoterReducer` code line 20
   a. What is the precision of the `mean`?
      0 - long divided by int
   b. Rewrite line 20 so that you will see significant digits after the decimal point. float mean = sum;
      mean = mean/count; (Two lines but this is guaranteed to work. Not most elegant).
9. What is the impact of defining the `sum` and `count` variables as instance variables rather than local variables in the `map()` method?

10. What is the value of the counters `MYGROUP.bad_num_tokens` and `MYGROUP.bad_age` based on the data provided?

One bad age (Priscilla zipper since less than 16). No one is older than 120.

One bad num token (Mike falkner) as he only has five fields. Did not see any others missing a field.

```
 1
 97
 57
----
154
```