课程编号：100076105　北京理工大学 2022-2023 学年第一学期

2020 级大数据系统开发期末试题 A 卷

班级_____　学号_____　姓名_____　成绩

一、单项选择（30分，每题2分）

1. 处理大数据的计算环境需要有很好的可扩展性，其主要原因是_____。
   A. 数据量大　C. 保护数据
   B. 数据增长快　D. 实时计算
2. 分布式的计算环境能为大数据处理带来的好处是_____。
   A. 可扩展、高可靠　C. 可扩展、高可用
   B. 高可用、高可靠　D. 可扩展、高可靠、高可用
3. 大数据 4V 特征中的 Value 指的是_____。
   A. 价值高　C. 价值高、价值密度低
   B. 价值密度低　D. 价值低、价值密度高
4. 大数据的大体量特征 Volume 指的是数据的规模大，一般大数据规模的体量级别是_____。
   A. EB、ZB　C. PB、ZB
   B. GB、TB、PB、EB　D. TB、PB、EB、ZB
5. 下列哪种应用更具备大数据的特征_____。
   A. 银行储蓄业务系统　C. 搜索引擎
   B. 医院收费系统　D. 短视频分享平台
6. Hadoop 是用于分布式环境下处理大数据的_____。
   A. 操作系统　C. 集群管理软件
   B. 软件平台　D. 编程语言
7. Hadoop 的主要贡献者是_____。
   A. Microsoft　C. Google
   B. Amazon　D. Facebook
8. Hadoop HDFS 是一个分布式的_____。
   A. 文件系统　C. 数据库管理系统
   B. 操作系统　D. 计算平台
9. 从 Hadoop HDFS 2.7.3 版本开始，其默认块大小为 128MB，与一般 Linux 文件系统块大小 512 字节相比很大，这样设计的原因是 Hadoop HDFS 处理的文件_____。
   A. 数量多　C. 节约存储空间
   B. 大　D. 数量多、文件大
10. HBase 的高可靠性主要来自_____。
    A. WAL、HDFS 的高可靠性　D. WAL、Replication、HDFS 的高可靠性
    B. Replication、HDFS 的高可靠性
    C. WAL、Replication
11. 部署在 HDFS 上的 HBase 中，负责读取 HDFS 的是_____。
    A. HMaster　B. RegionServer

C. Client　　D. Client、RegionServer

12. HBase 中，决定 Region 划分的是_____。

A. RowKey　　C. Block

B. Column Family　　D. Cloumn Key

13. MapReduce 适用于下列大数据应用场景的是_____。

A. 离线分析　　C. 流数据分析

B. 在线分析　　D. 实时数据分析

14. 使用 MapReduce 处理文档的倒排索引，决定一个 MapReduce 的作业中 map task 的数量的是_____。

A. 文档大小　　C. Data Node 的数量

B. 文档中每个单词的最大数量　　D. 运行作业的计算机的内存

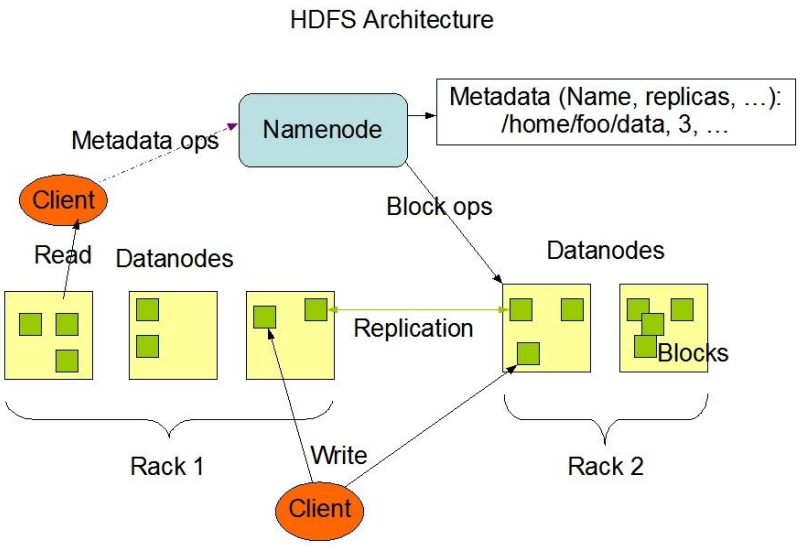
15. 默认情况下，定义 HDFS 数据块的冗余度的配置文件是

A. core-site.xml　　C. mapred-site.xml

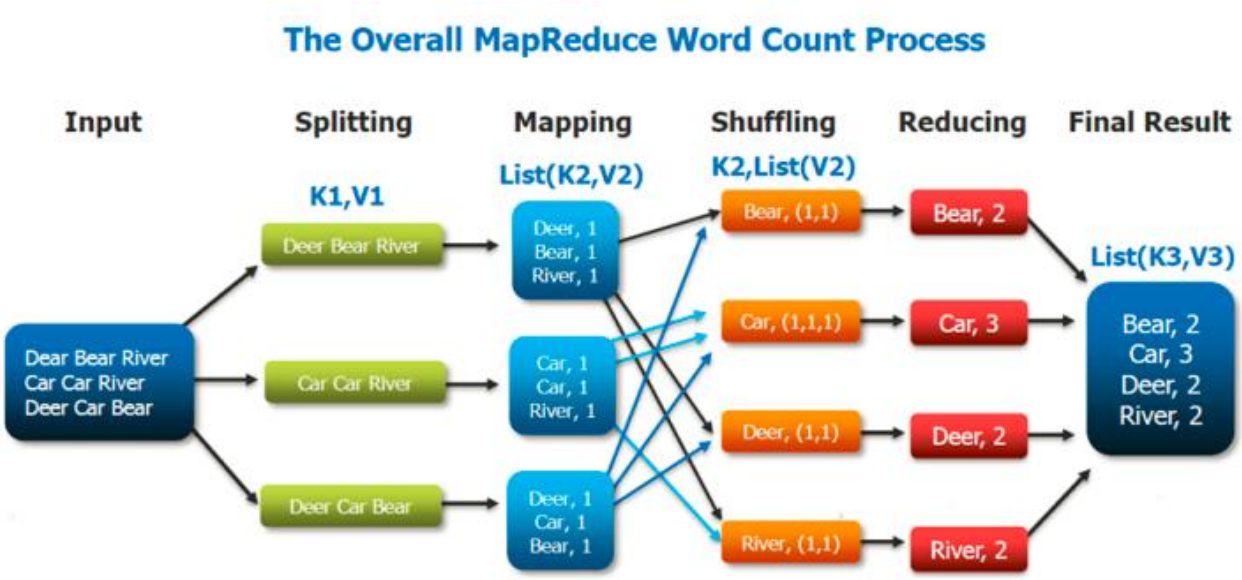
B. hdfs-site.xml　　D. yarn-site.xml

二、（30分，每题10分）简述题。

1. 结合下图说明 HDFS 的体系结构，说明中须包含 Client、Namenode、Datanodes、Metadata、Replication 的作用及相互关系。



2. 结合下图 MapReduce 处理词频统计的过程说明 MapReduce 中 Input、Splitting、Mapping、Shuffling、Reducing、Final Result 各阶段。



3. 简要说明在分布式环境（名称节点 Master、数据节点 1：Node1、数据节点 2：Node2）下安装 Hadoop 的流程，不要求写分布式环境的安装配置，只说明 Hadoop（不包括数据库）安装、配置的有关内容，不要求写具体命令（*注：如果实验中选择了项目二，且使用了其它大数据平台，可以选择相应的大数据平台进行说明*）。

三、（10分，每题2分）程序分析。根据下面关于程序的说明和代码，回答下面的问题，说明问题时可以引用代码中的行号。

1．代码第90行中为什么要使用job.setJarByClass(DataFromHdfs.class)设置jar包的类？
2．最终的结果是如何写入HBase的？
3．程序中是否设置了Combiner的类？如果设置了，是哪个类？如果没有，说明原因。
4．下面的程序是否可以在分布式环境下运行？说明原因。
5．说明map()输出的key、value

1. 环境描述：
   （1） 一系列的文本文件，文件名为 Txxx（后面的 xxx 表示数字，例如，T1，T2，T3），文件的内容包含一系列的英文句子，单词的分隔可能有空格、逗号、句号。
   （2） 输出结果写到已经创建好的 HBase 的 index 表中，包含单词，文件名：单词出现的数量，例如： count,T1:5。
2. Java 程序代码：

```
1  package com.pro.bq;
2
3  import java.io.IOException;
4  import java.util.StringTokenizer;
5
6  import org.apache.hadoop.conf.Configuration;
7  import org.apache.hadoop.fs.Path;
8  import org.apache.hadoop.hbase.HBaseConfiguration;
```

```
9  import org.apache.hadoop.hbase.client.Put;
10 import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
11 import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
12 import org.apache.hadoop.hbase.mapreduce.TableReducer;
13 import org.apache.hadoop.hbase.util.Bytes;
14 import org.apache.hadoop.io.Text;
15 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
16 import org.apache.hadoop.mapreduce.Job;
17 import org.apache.hadoop.mapreduce.Mapper;
18 import org.apache.hadoop.mapreduce.Reducer;
19 import org.apache.hadoop.mapreduce.lib.input.FileSplit;
20 import org.apache.hadoop.util.GenericOptionsParser;
21
22
23 public class DataFromHdfs {
24     public static class LocalMap extends Mapper<Object, Text, Text, Text>
25     {
26         private FileSplit split=null;
27         private Text keydata=null;
28         public void map(Object key, Text value,Context context)
29                 throws IOException, InterruptedException {
30
31             split=(FileSplit) context.getInputSplit();
32             StringTokenizer tokenStr=new StringTokenizer(value.toString());
33             while(tokenStr.hasMoreTokens())
34             {
35                 String token=tokenStr.nextToken();
36                 if(token.contains(",")||
                           token.contains(".")||token.contains(";")||token.contains("?"))
37                 {
38                     token=token.substring(0, token.length()-1);
39                 }
40                 String filePath=split.getPath().toString();
41                 int index=filePath.indexOf("T");
42                 keydata=new Text(token+":"+filePath.substring(index));
43                 context.write(keydata, new Text("1"));
44             }
45         }
46     }
47     public static class LocalCombiner extends Reducer<Text, Text, Text, Text>
48     {
49
50         public void reduce(Text key, Iterable<Text> values,Context context)
51                 throws IOException, InterruptedException {
```

```
52          int index=key.toString().indexOf(":");
53          Text keydata=new Text(key.toString().substring(0, index));
54          String filename=key.toString().substring(index+1);
55          int sum=0;
56          for(Text val:values)
57          {
58              sum++;
59          }
60          context.write(keydata, new Text(filename+":"+String.valueOf(sum)));
61      }
62  }
63  public static class TableReduce extends TableReducer<Text, Text, ImmutableBytesWritable>
64  {
65
66      public void reduce(Text key, Iterable<Text> values,Context context)
67              throws IOException, InterruptedException {
68          for(Text val:values)
69          {
70              int index=val.toString().indexOf(":");
71              String filename=val.toString().substring(0, index);
72              int sum=Integer.parseInt(val.toString().substring(index+1));
73              String row=key.toString();
74              Put put=new Put(Bytes.toBytes(key.toString()));
75
76              put.add(Bytes.toBytes("filesum"),
                    Bytes.toBytes("filename"), Bytes.toBytes(filename));
77              put.add(Bytes.toBytes("filesum"), Bytes.toBytes("count"),
                    Bytes.toBytes(String.valueOf(sum)));
78              context.write(new ImmutableBytesWritable(Bytes.toBytes(row)), put);
79          }
80
81      }
82  }
83  public static void main(String[] args) throws IOException,
                ClassNotFoundException, InterruptedException {
84      Configuration conf=new Configuration();
85      conf=HBaseConfiguration.create(conf);
86      String hdfsPath="hdfs://localhost:9000/user/haduser/";
87      String[] argsStr=new String[]{hdfsPath+"input/reverseIndex"};
88      String[] otherArgs=new GenericOptionsParser(conf, argsStr).getRemainingArgs();
89      Job job=new Job(conf);
90      job.setJarByClass(DataFromHdfs.class);
91
92      job.setMapperClass(LocalMap.class);
```

```
93          job.setCombinerClass(LocalCombiner.class);
94          job.setReducerClass(TableReduce.class);
95
96          job.setMapOutputKeyClass(Text.class);
97          job.setMapOutputValueClass(Text.class);
98
99          TableMapReduceUtil.initTableReducerJob("index", TableReduce.class, job);
100
101         FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
102         System.exit(job.waitForCompletion(true)?0:1);
103     }
104 }
```

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
import org.apache.hadoop.hbase.mapreduce.TableReducer;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;
import org.apache.hadoop.util.GenericOptionsParser;
```

```
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-client</artifactId>
    <version>${hbase.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-common</artifactId>
    <version>${hbase.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.hbase</groupId>
```

<artifactId>hbase-shaded-mapreduce</artifactId>
<version>${hbase.version}</version>

```
for(Text val:values)
    {
       int index=val.toString().indexOf(":");
       String filename=val.toString().substring(0, index);
       int sum=Integer.parseInt(val.toString().substring(index+1));
       String row=key.toString();

       Put put=new Put(Bytes.toBytes(key.toString()));
       put.addColumn(Bytes.toBytes("filesum"),
              Bytes.toBytes("filename"), Bytes.toBytes(filename));
       put.addColumn(Bytes.toBytes("filesum"), Bytes.toBytes("count"),
              Bytes.toBytes(String.valueOf(sum)));

       context.write(NullWritable.get(), put);
    }
```