一、试述 Hadoop 生态系统以及每个部分的具体功能。（10 分）

1. HDFS,分布式文件系统，提供大数据文件存储功能
2. MapReduce，提供分布式并行编程模型支持
3. YARN，集群资源调度
4. HBase，分布式数据库
5. Hive，数据集管理工具
6. Flume，日志采集系统
7. Sqoop，hadoop 与关系数据库交换组件
8. Mahout，机器学习领域的算法实现
9. Zookeeper，提供分布式系统的基础服务
10. Ambari，hadoop 集群的 web 工具

二、试述 HDFS 中名称节点运行的基本原理以及只设置唯一的名称节点的局限性表现在哪些方面。（10 分）

1. 命名空间的局限性：名称节点是保存在内存中的，因此名称节点能够容纳对象（文件、块）的个数会受到内存空间大小的限制。
2. 性能的瓶颈：整个分布式文件系统的吞吐量受限于单个名称节点的吞吐量。
3. 隔离问题：由于集群中只有一个名称节点，只有一个命名空间，因此无法对不同应用程序进行隔离。
4. 集群的可用性：一旦这个唯一的名称节点发生故障，会导致整个集群变得不可用。

三、试述 HBase 各功能组件及其作用。（10 分）

1. Client，HBASE 功能的使用者
2. Zookeeper，提供集群管理功能，管理每个 region 服务器
3. Master，管理用户丢表的操作，管理 region 服务器，实现负载均衡
4. Region，响应客户读写请求

四、（1）试述 Hlog 的工作原理。（5 分）

HBase 系统为每个 Region 服务器配置了一个 HLog 文件，它是一种预写式日志（Write Ahead Log），用户更新数据必须首先写入日志后，才能写入 MemStore 缓存，并且，直到 MemStore 缓存内容对应的日志已经写入磁盘，该缓存内容才能被刷写到磁盘。

（2）在 HBase 中，每个 Region 服务器维护一个 Hlog，而不是为每个 Region 都单独维护一个 Hlog。请说明这种做法的优点和缺点。（5 分）

优点： 多个 Region 对象的更新操作所发生的日志修改，只需要不断把日志记录追加到单个日志文件中，不需要同时打开、写入到多个日志文件中。

缺点：如果一个 Region 服务器发生故障，为了恢复其上次的 Region 对象，需要将 Region 服务器上的对象，需要将 Region 服务器上的 HLog 按照其所属的 Region 对象进行拆分，然后分发到其他 Region 服务器上执行恢复操作。

五、试述 CAP 理论的具体含义并举例说明不同产品在设计时是如何运用 CAP 理论的。（10 分）

一致性（Consistency）可用性（Availability）分区容错性（Partition tolerance）
(1) CA: 优先保证一致性和可用性，放弃分区容错。 这也意味着放弃系统的扩展性，系统不再是分布式的，有违设计的初衷。

(2) CP: 优先保证一致性和分区容错性，放弃可用性。在数据一致性要求比较高的场合(譬如:zookeeper,Hbase) 是比较常见的做法，一旦发生网络故障或者消息丢失，就会牺牲用户体验，等恢复之后用户才逐渐能访问。

(3) AP: 优先保证可用性和分区容错性，放弃一致性。NoSQL 中的 Cassandra 就是这种架构。跟 CP 一样，放弃一致性不是说一致性就不保证了，而是逐渐的变得一致。

六、Redis 数据库操作。（10 分）

Student 键值对如下:

```
zhangsan: {
        English: 69
        Math: 86
        Computer: 77
```

```
}
lisi: {
        English: 55
        Math: 100
        Computer: 88
}
```

根据上面给出的键值对，完成如下操作：

（1）用 Redis 的哈希结构设计出学生表 Student（键值可以用 student.zhangsan 和 student.lisi 来表示两个键值属于同一个表）；

```
127.0.0.1:6379> hmset student.zhangsan English 69 Math 86 Computer 77
OK
127.0.0.1:6379> hmset student.lisi English 55 Math 100 Computer 88
OK
```

（2）用 hgetall 命令分别输出 zhangsan 和 lisi 的成绩信息；

```
127.0.0.1:6379> hgetall student.lisi
1) "English"
2) "55"
3) "Math"
4) "100"
5) "Computer"
6) "88"
127.0.0.1:6379> hgetall student.zhangsan
1) "English"
2) "69"
3) "Math"
4) "86"
5) "Computer"
6) "77"
```

（3）用 hget 命令查询 zhangsan 的 Computer 成绩；

```
127.0.0.1:6379> hget student.zhangsan Computer
"77"
```

（4）修改 lisi 的 Math 成绩，改为 95。

```
127.0.0.1:6379> hset student.lisi Math 95
(integer) 0
127.0.0.1:6379> hget student.lisi Math
"95"
127.0.0.1:6379>
```

七、MongoDB 数据库操作。

Student 文档如下:

```
{
    “name”: “zhangsan”,
    “score”: {
        “English”: 69,
        “Math”: 86,
        “Computer”: 77
    }
}
{
    “name”: “lisi”,
    “score”: {
        “English”: 55,
        “Math”: 100,
        “Computer”: 88
    }
}
```

1.根据上面给出的文档，完成如下操作：（20 分）

（1）用 MongoDB Shell 设计出 student 集合;

use student

```
> db.createCollection("student")
{ "ok" : 1 }
>
```

```
> db.student.insert(
... {
... name: "zhangsan",
... score:{
... English: 69,
... Math: 86,
... Computer: 77
... }
... })
WriteResult({ "nInserted" : 1 })
> db.student.insert(
... {
... name: "lisi",
... score: {
... English: 55,
... Math: 100,
... Computer: 88
... }
... })
WriteResult({ "nInserted" : 1 })
>
```

（2）用 find()方法输出两个学生的信息；

```
> db.student.find({name:"lisi"}).pretty()
{
        "_id" : ObjectId("61a1a40d5d752e61810d5736"),
        "name" : "lisi",
        "score" : {
                "English" : 55,
                "Math" : 100,
                "Computer" : 88
        }
}
> db.student.find({name:"zhangsan"}).pretty()
{
        "_id" : ObjectId("61a1a38f5d752e61810d5735"),
        "name" : "zhangsan",
        "score" : {
                "English" : 69,
                "Math" : 86,
                "Computer" : 77
        }
}
>
```

（3）用 find()方法查询 zhangsan 的所有成绩(只显示 score 列)；

db.student.find({name:”zhangsan”},{“score”:1});

```
> db.student.find({name:"zhangsan"},{"score":"score"}).pretty()
{
        "_id" : ObjectId("61a1a38f5d752e61810d5735"),
        "score" : {
                "English" : 69,
                "Math" : 86,
                "Computer" : 77
        }
}
> db.student.find({name:"lisi"},{"score":"score"}).pretty()
{
        "_id" : ObjectId("61a1a40d5d752e61810d5736"),
        "score" : {
                "English" : 55,
                "Math" : 100,
                "Computer" : 88
        }
}
>
```

（4）修改 lisi 的 Math 成绩，改为 95。

mongodb 有两种更新操作，分别是$set 和$unset，set 表示添加或修改，unset 表示删除

db.student.update({name:”lisi”},{$set:{math:95}});

db.student.update({name=”lisi”,{$unset:{math:””}}});

```
> db.student.update({name:"lisi"},{score:{math:95}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

2.根据上面已经设计出的 Student 集合，用 MongoDB 的 Java 客户端编程，实现如下操作:

（1）添加数据：English 为 45 ，Math 为 89，Computer 为 100。（10 分）

与上述数据对应的文档形式如下:

```
{
    “name”: “scofield”,
    “score”: {
        “English”: 45,
        “Math”: 89,
        “Computer”: 100
    }
```

```
}
```

```
    static String uri = "mongodb://java_client:8218630@localhost:27017/student";
  public static void main(String[] args) {
    try (MongoClient mongoClient = MongoClients.create(uri)) {
      MongoDatabase database = mongoClient.getDatabase("student");
      MongoCollection<Document> collection = database.getCollection("student");
//       Document doc = collection.find(eq("name", "scofield")).first();
//       System.out.println(doc.toJson());
      Document score = new Document();
      score.put("English",45);
      score.put("Math",89);
      score.put("Computer",89);
      Document doc = new Document();
      doc.put("name","scofield");
      doc.put("score",score);
      collection.insertOne(doc);
    }
  }
```

（2）获取 scofield 的所有成绩信息(只显示 score 列)。（10 分）

```
MongoDatabase database = mongoClient.getDatabase("student");
MongoCollection<Document> collection = database.getCollection("student");
Document doc = collection.find(eq("name", "scofield"),).first();
System.out.println(doc.toJson());
```