问答题（100分）：

（问答题，共13道题目，第1题4分，第2题6分，第3题10分，第4题10分，第5题16分，第6题6分，第7题12分，第8题6分，第9题6分，第10题8分，第11题4分，第12题6分，第13题6分）

1.(4%)请阐述大数据、云计算和物联网三者之间的关系。

答案：

近几年来，云计算受到学术界和工业界的热捧，随后，大数据横空出世，更是炙手可热。那么，大数据和云计算之间是什么关系呢？

（1）从整体上看，大数据与云计算是相辅相成的

大数据着眼于“数据”，关注实际业务，提供数据采集分析挖掘，看重的是信息积淀，即数据存储能力。云计算着眼于“计算”，关注 IT 解决方案，提供 IT 基础架构，看重的是计算能力，即数据处理能力。没有大数据的信息积淀，则云计算的计算能力再强大，也难以找到用武之地；没有云计算的处理能力，则大数据的信息积淀再丰富，也终究只是镜花水月。

（2）从技术上看，大数据根植于云计算

云计算关键技术中的海量数据存储技术、海量数据管理技术、MapReduce编程模型，都是大数据技术的基础。

（3）物联网借助于云计算存储数据，借助于大数据技术分析数据。

2.(6%)下图是Hadoop生态系统图，请分别阐述Hadoop生态系统的各个组成部分（Avro、Zookeeper、HDFS、HBase、MapReduce、Pig、Hive、Sqoop）的主要功能。

答案：

Avro是用于数据序列化的系统。它提供了丰富的数据结构类型、快速可压缩的二进制数据格式、存储持久性数据的文件集、远程调用RPC的功能和简单的动态语言集成功能。

HDFS是一种分布式文件系统，运行于大型商用机集群，HDFS为HBase提供了高可靠性的底层存储支持。

HBase位于结构化存储层，是一个分布式的列存储数据库。

Mapreduce是一种编程模型，用于大规模数据集（大于1TB）的并行运算。

Zookeeper是一个分布式的、高可用性的协调服务，提供分布式锁之类的基本服务，用于构建分布式应用，为HBase提供了稳定服务和失败恢复机制。

Hive最早是由Facebook设计的，是一个建立在Hadoop基础之上的数据仓库，它提供了一些对存储在Hadoop文件中的数据集进行数据整理、特殊查询和分析的工具。

Pig是一种数据流语言和运行环境，用以检索非常大的数据集，大大简化了Hadoop常见的工作任务。

Sqoop为HBase提供了方便的RDBMS数据导入功能，使得传统数据库数据向HBase中迁移变得非常方便。

3. （10%）

（1）请论述HDFS1.0中SecondaryNameNode的作用和工作原理；（6%）

（2）请阐述HDFS的名称节点中关于数据块到数据节点的映射信息是如何维护的？（2%）

（3）HDFS HA解决了HDFS1.0中的什么问题？（2%）

答案：

（1）作用是减少Editlog大小，缩短名称节点重启时间。

SecondaryNameNode的工作情况：

（1）SecondaryNameNode会定期和NameNode通信，请求其停止使用EditLog文件，暂时将新的写操作写到一个新的文件edit.new上来，这个操作是瞬间完成，上层写日志的函数完全感觉不到差别；

　　（2）SecondaryNameNode通过HTTP GET方式从NameNode上获取到FsImage和EditLog文件，并下载到本地的相应目录下；

　　（3）SecondaryNameNode将下载下来的FsImage载入到内存，然后一条一条地执行EditLog文件中的各项更新操作，使得内存中的FsImage保持最新；这个过程就是EditLog和FsImage文件合并；

　　（4）SecondaryNameNode执行完（3）操作之后，会通过post方式将新的FsImage文件发送到NameNode节点上

　　（5）NameNode将从SecondaryNameNode接收到的新的FsImage替换旧的FsImage文件，同时将edit.new替换EditLog文件，通过这个过程EditLog就变小了

（2）

请阐述HDFS的名称节点中关于数据块到数据节点的映射信息，是由数据节点不断向名称节点发送心跳信息来维护的，数据节点把自己维护的数据块列表发送给名称节点，名称节点根据各个数据节点汇报结果生成映射信息。

（3）HDFS HA解决了HDFS1.0中的单点故障问题。

4.(10%) 在MapReduce作业中，combine操作是可选的：

（1）请指出combine操作的作用；（2分）

（2）请指出combine操作在哪些场景下才能使用？（2分）

（3）请指出在MapReduce作业的Shuffle过程中，哪些环节可以使用combine操作？（6分）

答案：

（1）Combiner是可选择的，它的主要作用是在每一个Map执行完分析以后，在本地优先做Reduce的工作，减少在Reduce过程中的数据传输量。

（2）Combiner会优化MapReduce的中间结果，所以它在整个模型中会多次使用。那么，哪些场景才能使用Combiner呢？从这里分析，Combiner的输出是Reducer的输入，Combiner绝不能改变最终的计算结果。所以，一般而言，Combiner只应该用于那种Reduce的输入key/value与输出key/value类型完全一致、且不影响最终结果的场景，比如累加、最大值等。Combiner的使用一定得慎重，如果用好，它对job执行效率有帮助，反之，则会影响reduce的最终结果。

（3）

第一个使用combine的地方：在map作业执行过程中，每个map task都有一个内存缓冲区，存储着map的输出结果，当缓冲区快满的时候，需要将缓冲区的数据以一个临时文件的方式存放到磁盘，如果client设置过Combiner，那么现在就是使用Combiner的时候了。将有相同key的key/value对的value加起来，减少溢写到磁盘的数据量。

第二个使用combine的地方：每次溢写会在磁盘上生成一个溢写文件，如果map的输出结果真的很大，有多次这样的溢写发生，磁盘上相应的就会有多个溢写文件存在。当map task真正完成时，内存缓冲区中的数据也全部溢写到磁盘中形成一个溢写文件。因为最终的文件只允许有一个，所以需要将这些溢写文件归并到一起，这个过程就叫做Merge。因为merge是将多个溢写文件合并到一个文件，所以可能也有相同的key存在，在这个过程中如果client设置过Combiner，也会使用Combiner来合并相同的key。

第三个使用combine的地方：在Reduce作业执行时，Reduce进程启动一些数据复制线程(Fetcher)，通过HTTP方式请求map task所在的TaskTracker获取map task的输出文件。复制过来的数据会先放入内存缓冲区中，当内存中的数据量到达一定阈值，就启动内存到磁盘的merge。与map 端类似，这也是溢写的过程，这个过程中如果你设置有Combiner，也是会启用的，然后在磁盘中生成了众多的溢写文件。

5.(16%)（1）请阐述HBase三层结构；（6分）

（2）请阐述在HBase三层结构下，客户端是如何访问到数据的？（2分）

（3）在HBase中，每个HRegion Server维护一个HLog，而不是每个HRegion一个。请说明这种做法的优点和缺点。（3分）

（4）当一台HRegionServer意外终止时，主服务器HMaster如何发现这种意外终止情况？为了恢复这台发生意外的HRegionServer上的HRegion，HMaster应该做出哪些处理（包括如何使用HLog进行恢复）？（5分）

答案：

（1）HBase使用类似B+树的三层结构来保存HRegion位置信息：

第一层是Zookeeper文件：它记录了-ROOT-表的位置信息，即root region的位置信息；

第二层是-ROOT-表：只包含一个root region，记录了.META.表中的region信息。通过root region，我们就可以访问.META.表的数据。

第三层是.META.表：是一个特殊的表，记录了用户表的HRegion信息，.META.表可以有多个HRegion，保存了HBase中所有数据表的HRegion位置信息。

图7-7 HBase的三层结构

（2）Client访问用户数据之前，需要首先访问Zookeeper，然后访问-ROOT-表，接着访问.META.表，最后才能找到用户数据的位置去访问，中间需要多次网络操作，不过client端会做cache缓存。

（3）每个HRegion Server维护一个HLog，而不是每个HRegion一个。这样不同HRegion(来自不同表)的日志会混在一起，这样做的目的是，不断追加单个文件相对于同时写多个文件而言，可以减少磁盘寻址次数，因此，可以提高对表的写性能。带来的麻烦是，如果一台HRegionServer下线，为了恢复其上的HRegion，需要将HRegionServer上的HLog进行拆分，然后分发到其它HRegionServer上进行恢复。

（4）当HRegionServer意外终止后，HMaster会通过Zookeeper感知到，HMaster首先会处理遗留的HLog文件，将其中不同HRegion的HLog数据进行拆分，分别放到相应HRegion的目录下，然后再将失效的HRegion重新分配，领取到这些HRegion的HRegionServer在加载HRegion的过程中，会发现有历史HLog需要处理，因此会Replay HLog中的数据到HMemStore中，然后刷新到HStoreFiles，完成数据恢复。

6.(6%)在Storm框架设计中，可以kill（杀）掉nimbus进程和supervisor进程，然后重启，它们将恢复状态并继续工作，表现出了极其好的稳定性。请问这种保证Storm稳定性的机制是如何设计的？

答案：

Storm采用了Zookeeper来作为分布式协调组件，一个Zookeeper集群负责Nimbus和多个Supervisor之间的所有协调工作（一个完整的拓扑可能被分为多个子拓扑，并由多个supervisor完成）。

Nimbus后台程序和Supervisor后台程序都是快速失败（fail-fast）和无状态的，所有状态维持在Zookeeper或本地磁盘中。

在这种设计中，master节点并没有直接和worker节点通信，而是借助中介Zookeeper，这样一来可以分离master和worker的依赖，将状态信息存放在Zookeeper集群内以快速回复任何失败的一方。

这意味着你可以kill杀掉nimbus进程和supervisor进程，然后重启，它们将恢复状态并继续工作，这种设计使得Storm极其稳定。

7.(12%)（1）Pregel的容错是通过检查点来实现的，请阐述Pregel的容错机制。（6分）

（2）请描述用Pregel解决单源最短路径问题的基本过程。（6分）

答案：

（1）容错是通过检查点来实现的。在每个超步的开始阶段，master命令worker让它保存它上面的partitions的状态到持久存储设备，包括顶点值、边值以及接收到的消息。Master自己也会保存aggregator的值。

worker的失效是通过master发给它的周期性的ping消息来检测的。如果一个worker在特定的时间间隔内没有收到ping消息，该worker进程会终止。如果master在一定时间内没有收到worker的反馈，就会将该worker进程标记为失败。

当一个或多个worker发生故障，被分配到这些worker的partitions的当前状态信息就丢失了。Master重新分配图的partition到当前可用的worker集合上，所有的partition会从最近的某超步S开始时写出的检查点中，重新加载状态信息。该超步可能比在失败的worker上最后运行的超步 S’要早好几个阶段，此时，失去的几个超步将需要被重新执行。Pregel对检查点频率的选择，是基于某个故障模型的平均时间的，从而平衡检查点的开销和恢复执行的开销。

（2）在该算法中，我们假设与顶点关联的那个值被初始化为INF。在每个超步中，每个顶点会首先接收到来自邻居传送过来的消息，该消息包含更新过的、从源顶点到该顶点的潜在的最短距离。如果这些更新里的最小值小于该顶点当前关联值，那么顶点就会更新这个值，并发送消息给它的邻居。在第一个超步中，只有源顶点会更新它的关联值，然后发送消息给它的直接邻居。然后这些邻居会更新它们的关联值，然后继续发送消息给它们的邻居，如此循环往复。当没有更新再发生的时候，算法就结束，之后所有顶点的关联值就是从源顶点到它的最短距离，若值为INF表示该顶点不可达。如果所有的边权重都是非负的，就可以保证该过程肯定会结束。

8.(6%)请描述作为NoSQL数据库的基石之一的BASE的含义。

答案：

说起来很有趣，BASE的英文意义是碱，而ACID的英文含义是酸，看起来二者似乎是“水火不容”。BASE的基本含义如下：

Basically Availble：基本可用，支持分区失败；

Soft-state：软状态/柔性事务，可以理解为"无连接"的，可以有一段时间不同步；而 "Hard state"是"面向连接"的；

Eventual Consistency：最终一致性，也是 ACID 的最终目的，最终数据是一致的就可以了，而不是时时一致。

BASE模型是反ACID模型的，完全不同于ACID模型，牺牲了高一致性，从而获得可用性或可靠性。BASE思想主要强调基本的可用性，如果你需要高可用性，也就是纯粹的高性能，那么就要以一致性或容错性为牺牲，BASE思想的方案在性能上还是有潜力可挖的。

9.(6%) 假设关系R(A，B)和S(B,C)都存储在一个文件中。请阐述如何用MapReduce实现R和S这两个关系的连接(join)操作。

答案：

总的来说，在MapReduce环境下执行两个关系的联接操作的方法如下：假设关系R(A，B)和S(B,C)都存储在一个文件中。为了联接这些关系，必须把来自每个关系的各个元组都和一个key关联，这个key就是属性B的值。可以使用一个Map进程集合，把来自R的每个元组(a,b)转换成一个key-value对，其中的key就是b，值就是(a,R)。注意，这里把关系R包含到value中，这样做使得我们可以在Reduce阶段，只把那些来自R的元组和来自S的元组进行匹配。类似地，可以使用一个Map进程集合，把来自S的每个元组(b,c)，转换成一个key-value对，key是b，value是(c,S)。这里把关系名字包含在属性值中，可以使得在Reduce阶段只把那些来自不同关系的元组进行合并。Reduce进程的任务就是，把来自关系R和S的具有共同属性B值的元组进行合并。这样，所有具有特定B值的元组必须被发送到同一个Reduce进程。假设使用k个Reduce进程。这里选择一个哈希函数h，它可以把属性B的值映射到k个哈希桶，每个哈希值对应一个Reduce进程。每个Map进程把key是b 的key-value对，都发送到与哈希值h(b)对应的Reduce进程。Reduce进程把联接后的元组(a,b,c)，写到一个单独的输出文件中。

10.(8%)

（1）请画出MapReduce1.0体系结构，并说明各个组件的功能；（4%）

（2）指出MapReduce1.0体系结构存在的缺陷。（4%）

答案：

MapReduce主要有以下4个部分组成：

1）Client

用户编写的MapReduce程序通过Client提交到JobTracker端

用户可通过Client提供的一些接口查看作业运行状态

2）JobTracker

JobTracker负责资源监控和作业调度

JobTracker 监控所有TaskTracker与Job的健康状况，一旦发现失败，就将相应的任务转移到其他节点

JobTracker 会跟踪任务的执行进度、资源使用量等信息，并将这些信息告诉任务调度器（TaskScheduler），而调度器会在资源出现空闲时，选择合适的任务去使用这些资源

3）TaskTracker

TaskTracker 会周期性地通过“心跳”将本节点上资源的使用情况和任务的运行进度汇报给JobTracker，同时接收JobTracker 发送过来的命令并执行相应的操作（如启动新任务、杀死任务等）

TaskTracker 使用“slot”等量划分本节点上的资源量（CPU、内存等）。一个Task 获取到一个slot 后才有机会运行，而Hadoop调度器的作用就是将各个TaskTracker上的空闲slot分配给Task使用。slot 分为Map slot 和Reduce slot 两种，分别供MapTask 和Reduce Task 使用

4）Task

Task 分为Map Task 和Reduce Task 两种，均由TaskTracker 启动

11.（4%）YARN的目标就是实现“一个集群多个框架”，所有计算框架采用YARN来提供统一的资源管理调度服务，请问这样做可以具有哪些优点？

答案：

YARN的目标就是实现“一个集群多个框架”，即在一个集群上部署一个统一的资源调度管理框架YARN，在YARN之上可以部署其他各种计算框架

由YARN为这些计算框架提供统一的资源调度管理服务，并且能够根据各种计算框架的负载需求，调整各自占用的资源，实现集群资源共享和资源弹性收缩

可以实现一个集群上的不同应用负载混搭，有效提高了集群的利用率

不同计算框架可以共享底层存储，避免了数据集跨集群移动

12（6%）

（1）相对于MapReduce而言，在执行迭代计算方面，为什么Spark具有更好的性能？（2%）

答案：每次迭代结束，MapReduce需要写入磁盘，而Spark写内存。

（1）为什么说Spark的设计具有天生的容错性？（2%）

答案：RDD之间形成血缘关系，任何一个RDD分区失效，都可以从其父节点来重新计算生成。

（2）Spark有哪三种部署方式？（2%）

答案：

Standalone（类似于MapReduce1.0，slot为资源分配单位）

Spark on Mesos（和Spark有血缘关系，更好支持Mesos）

Spark on YARN

13（6%） 试述两种典型的推荐算法UserCF和ItemCF算法的区别，以及各自优缺点

UserCF算法和ItemCF算法的思想、计算过程都相似

两者最主要的区别：

UserCF算法推荐的是那些和目标用户有共同兴趣爱好的其他用户所喜欢的物品

ItemCF算法推荐的是那些和目标用户之前喜欢的物品类似的其他物品

UserCF算法的推荐更偏向社会化，而ItemCF算法的推荐更偏向于个性化

UserCF算法的推荐更偏向社会化：适合应用于新闻推荐、微博话题推荐等应用场景，其推荐结果在新颖性方面有一定的优势

UserCF缺点：随着用户数目的增大，用户相似度计算复杂度越来越高。而且UserCF推荐结果相关性较弱，难以对推荐结果作出解释，容易受大众影响而推荐热门物品

ItemCF算法的推荐更偏向于个性化：适合应用于电子商务、电影、图书等应用场景，可以利用用户的历史行为给推荐结果作出解释，让用户更为信服推荐的效果

ItemCF缺点：倾向于推荐与用户已购买商品相似的商品，往往会出现多样性不足、推荐新颖度较低的问题