1.在Hadoop中定义的主要公用InputFormat中，默认是哪一个？（A）

|  |
| --- |
| A、 TextInputFormat  B、 KeyValueInputFormat  C、 SequenceFileInputFormat |

1. 下面哪个程序负责 HDFS 数据存储？（C）

|  |
| --- |
| A.NameNode  B.JobTracker  C.DataNode  D.SecondaryNameNode  E.tasktracker |

2.HDFS 中的 block 默认保存几份？（A）

|  |
| --- |
| A.3 份  B.2 份  C.1 份  D.不确定 |

3.下列哪个程序通常与 NameNode 在一个节点启动？（D）

|  |
| --- |
| A.SecondaryNameNode  B.DataNode  C.TaskTracker  D.JobTracker |

|  |
| --- |
| hadoop的集群是基于master/slave模式，namenode和jobtracker属于master，datanode和tasktracker属于slave，master只有一个，而slave有多个.  SecondaryNameNode内存需求和NameNode在一个数量级上，所以通常secondary NameNode（运行在单独的物理机器上）和 NameNode 运行在不同的机器上。  JobTracker对应于NameNode,TaskTracker对应于DataNode.  DataNode和NameNode是针对数据存放来而言的.JobTracker和TaskTracker是对于MapReduce执行而言的.  mapreduce中几个主要概念，mapreduce 整体上可以分为这么几条执行线索：  jobclient，JobTracker与TaskTracker。  1、JobClient会在用户端通过JobClient类将已经配置参数打包成jar文件的应用存储到hdfs，并把路径提交到Jobtracker,然后由JobTracker创建每一个Task（即 MapTask 和 ReduceTask） 并将它们分发到各个TaskTracker服务中去执行。  2、JobTracker是一master服务，软件启动之后JobTracker接收Job，负责调度Job的每一个子任务。task运行于TaskTracker上，并监控它们，如果发现有失败的task就重新运行它。一般情况应该把JobTracker 部署在单独的机器上。  3、TaskTracker是运行在多个节点上的slaver服务。TaskTracker主动与JobTracker通信，接收作业，并负责直接执行每一个任务。 TaskTracker 都需要运行在HDFS的DataNode上。 |

4.关于 SecondaryNameNode 哪项是正确的？（C）

|  |
| --- |
| A.它是 NameNode 的热备  B.它是内存没有要求  C.他的目的使帮助 NameNode 合并编辑日志，减少 NameNode 启动时间  D. SecondaryNameNode 应与 NameNode 部署到一个节点 |

HDFS 默认 BlockSize 是（B）

|  |
| --- |
| A.32MB  B.64MB  C.128MB |

下列哪项通常是集群的最主要瓶颈（C）

|  |
| --- |
| A.CPU  B.网络  C.磁盘 IO  D.内存 |

1. 下列哪项可以作为集群的管理？（ABC）

|  |
| --- |
| A.Puppet  B.Pdsh  C.ClouderaManager  D.Zookeeper |

2.Client 端上传文件的时候下列哪项正确？（BC）

|  |
| --- |
| A.数据经过 NameNode 传递 DataNode  B.Client 端将文件切分为 Block，依次上传  C.Client 只上传数据到一台 DataNode，然后由 NameNode 负责 Block 复制工作 |

3.下列哪个是 Hadoop 运行的模式？（ABC）

|  |
| --- |
| A.单机版  B.伪分布式  C.分布式 |

1.列举几个hadoop生态圈的组件并做简要描述

|  |
| --- |
| Zookeeper:是一个开源的分布式应用程序协调服务,基于zookeeper可以实现同步服务，配置维护，命名服务。  Flume:一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统。  Hbase:是一个分布式的、面向列的开源数据库, 利用Hadoop HDFS作为其存储系统.  Hive:基于Hadoop的一个数据仓库工具，可以将结构化的数据档映射为一张数据库表，并提供简单的sql 查询功能，可以将sql语句转换为MapReduce任务进行运行。  Sqoop:将一个关系型数据库中的数据导进到Hadoop的 HDFS中，也可以将HDFS的数据导进到关系型数据库中。 |

Hadoop中需要哪些配置文件，其作用是什么？

|  |
| --- |
| 1>.Core-site.xml：  (1)fs.defaultFS:hdfs://cluster1(域名)，这里的值指的是默认的HDFS路径 。  (2)hadoop.tmp.dir:/export/data/hadoop\_tmp,这里的路径默认是NameNode、DataNode、JournalNode等存放数据的公共目录。用户也可以自己单独指定这三类节点的目录。  (3)ha.zookeeper.quorum:hadoop101:2181,hadoop102:2181,hadoop103:2181,这里是ZooKeeper集群的地址和端口。注意，数量一定是奇数，且不少于三个节点 。  2>.Hadoop-env.sh: 只需设置jdk的安装路径，如：export JAVA\_HOME=/usr/local/jdk 本机j位置  3>.Hdfs-site.xml：  (1) dfs.replication:他决定着系统里面的文件块的数据备份个数，默认为3个。  (2) dfs.data.dir:datanode节点存储在文件系统的目录 。  (3) dfs.name.dir:是namenode节点存储hadoop文件系统信息的本地系统路径 。  4>.Mapred-site.xml：  (1)mapred.job.tracker:是jobtracker的主机和端口配置 。 |

4. 请列出正常工作的Hadoop集群中Hadoop都分别需要启动哪些进程，它们的作用分别是什么?

|  |
| --- |
| a) NameNode它是hadoop中的主服务器，管理文件系统名称空间和对集群中存储的文件的访问，保存有 metadate.  b).SecondaryNameNode它不是namenode的冗余守护进程，而是提供周期检查点和清理任务。帮助NN合并editslog，减少NN启动时间。  c)DataNode它负责管理连接到节点的存储（一个集群中可以有多个节点）。每个存  储数据的节点运行一个datanode守护进程。  d)ResourceManager（JobTracker）JobTracker负责调度DataNode上的工作。每个DataNode有一个TaskTracker，它们执行实际工作。  e) NodeManager（TaskTracker）执行任务  f) DFSZKFailoverController高可用时它负责监控NN的状态，并及时的把状态信息写入ZK。它通过一个独立线程周期性的调用NN上的一个特定接口来获取NN的健康状态。FC也有选择谁作为Active NN的权利，因为最多只有两个节点，目前选择策略还比较简单（先到先得，轮换）。  g) JournalNode 高可用情况下存放namenode的editlog文件. |

HDFS数据写入实现机制

|  |
| --- |
| 3. HDFS的存储机制？  写入HDFS过程：  Client调用DistributedFileSystem对象的create方法，创建一个文件输出流（FSDataOutputStream）对象，通过DistributedFileSystem对象与Hadoop集群的NameNode进行一次RPC远程调用，在HDFS的Namespace中创建一个文件条目（Entry），该条目没有任何的Block，通过FSDataOutputStream对象，向DataNode写入数据，数据首先被写入FSDataOutputStream对象内部的Buffer中，然后数据被分割成一个个Packet数据包，以Packet最小单位，基于Socket连接发送到按特定算法选择的HDFS集群中一组DataNode（正常是3个，可能大于等于1）中的一个节点上，在这组DataNode组成的 Pipeline上依次传输 Packet， 这组 DataNode 组成的 Pipeline反方向上，发送 ack，最终由 Pipeline 中第一个 DataNode 节点将 Pipeline ack发送给Client，完成向文件写入数据，Client 在文件输出流（FSDataOutputStream）对象上调用close方法，关闭流.调用DistributedFileSystem对象的complete 方法，通知 NameNode 文件写入成功.  读取文件过程：  使用HDFS提供的客户端开发库Client，向远程的Namenode发起RPC请求；Namenode会视情况返回文件的部分或全部block列表，对于每个block，Namenode都会返回有该block拷贝的DataNode地址；客户端开发库Client会选取离客户端最接近的DataNode来读取block；如果客户端本身就是DataNode,那么将从本地直接获取数据.读取完当前block的数据后，关闭与当前的DataNode连接，并为读取下一个block寻找最佳的DataNode；当读完列表的block后，且文件读取还没有结束，客户端开发库会继续向Namenode获取下一批的block列表。读取完一个block都会进行 checksum 验证，如果读取 datanode 时出现错误，客户端会通知 Namenode，然后再从下一个拥有该 block 拷贝的 datanode 继续读。 |

2.HDFS的存储机制是什么

|  |
| --- |
| 客户端通过把请求发送给NameNode active，NN会把文件切成1个到N个固定大小的block（一般默认为128M）并上传到DN中。当所有block拷贝结束时，NN会立即通知客户端上传结果。但此时上传的流程还未结束。DN还需要根据配置信息的副本数量，在不同的机架节点上通过局域网作数据拷贝工作。 |

6Hadoop 是由哪几个组件组成

|  |
| --- |
| hadoop 是一套处理大数据的生态系统，包括文件存储HDFS ，计算框架MapReduce调度框架Yarn，数据库Hbase ，数据仓库Hive，协调与锁服务zookeeper，关系型数 据库与Hbase转换工具sqoop，工作流模块Oozie,机器学习模块mahout. |

1.6 hadoop 节点动态上线下线怎么操作?

|  |
| --- |
| 节点上线操作：  当要新上线数据节点的时候，需要把数据节点的名字追加在 dfs.hosts 文件中  一，关闭新增节点的防火墙  二，在 NameNode 节点的 hosts 文件中加入新增数据节点的 hostname  三，在每个新增数据节点的 hosts 文件中加入 NameNode 的 hostname  四，在 NameNode 节点上增加新增节点的 SSH 免密码登录的操作  五，在 NameNode 节点上的 dfs.hosts 中追加上新增节点的 hostname,  六，在其他节点上执行刷新操作：hdfs dfsadmin -refreshNodes  七，在 NameNode 节点上，更改 slaves 文件，将要上线的数据节点 hostname 追加  到 slaves 文件中  八，启动 DataNode 节点  九，查看 NameNode 的监控页面看是否有新增加的节点  节点下线操作：  一，修改/conf/hdfs-site.xml 文件  二，确定需要下线的机器，dfs.osts.exclude 文件中配置好需要下架的机器，这个是阻  止下架的机器去连接 NameNode  三，配置完成之后进行配置的刷新操作./bin/hadoop dfsadmin -refreshNodes,这个  操作的作用是在后台进行 block 块的移动  四，当执行三的命令完成之后，需要下架的机器就可以关闭了，可以查看现在集  群上连接的节点，正在执行 Decommission，会显示：  Decommission Status : Decommission in progress 执行完毕后，会显示：  Decommission Status : Decommissioned  五，机器下线完毕，将他们从 excludes 文件中移除。 |

10.Hadoop 中 job 和 task 之间的区别是什么？

|  |
| --- |
| JobTracker 是一个 master 服务，软件启动之后 JobTracker 接收 Job，负责调度 Job  的每一个子任务 task 运行于 TaskTracker 上，并监控它们，如果发现有失败的 task 就重新  运行它。一般情况应该把 JobTracker 部署在单独的机器上。  TaskTracker 是运行在多个节点上的 slaver 服务。TaskTracker 主动与 JobTracker 通信，  接收作业，并负责直接执行每一个任务11.Hadoop 中通过拆分任务到多个节点运行来实现并行计  算，但某些节点运行较慢会拖慢整个任务的运行，Hadoop  采用何种机制应对这个情况？  答: 推测执行机制是 Hadoop 对“拖后腿”的任务的一种优化机制，当一个作业的某些任务  运行速度明显慢于同作业的其他任务时，Hadoop 会在另一个节点 上为“慢任务”启动一  个备份任务，这样两个任务同时处理一份数据，而 Hadoop 最终会将优先完成的那个任务  的结果作为最终结果，并将另一个任务杀掉。 |

2. mapred.job.tracker 命令的作用

|  |
| --- |
| 连接jobtrack服务器的配置项，默认不写是local，在本地运行，默认 map 数1，reduce数1。 |

7 如何决定一个job的map和reduce的数量?

|  |
| --- |
| splitSize=max{minSize,min{maxSize,blockSize}}  map数量由处理的数据分成的block数量决定default\_num = total\_size / split\_size;  reduce的数量job.setNumReduceTasks(x);x 为reduce的数量. |

14.如何为一个Hadoop任务设置mappers的数量？

|  |
| --- |
| 具体的数据分片是这样的，InputFormat在默认情况下会根据hadoop集群HDFS块大小进行分片，每一个分片会由一个map任务来进行处理，当然用户还是可以通过参数mapred.min.split.size参数在作业提交客户端进行自定义设置。还有一个重要参数就是mapred.map.tasks，这个参数设置的map数量仅仅是一个提示，只有当InputFormat决定了map任务的个数比mapred.map.tasks值小时才起作用。同样，Map任务的个数也能通过使用JobConf的conf.setNumMapTasks(int num)方法来手动地设置。这个方法能够用来增加map任务的个数，但是不能设定任务的个数小于Hadoop系统通过分割输入数据得到的值。 |

15.如何为一个Hadoop任务设置要创建的 reducer的数量？

|  |
| --- |
| 纯粹的 mapreduce task 的 reduce task 数很简单，就是参数 mapred.reduce.tasks  的值，hadoop-site.xml 文件中和 mapreduce job 运行时。不设置的话默认为 1。 |

10 yarn 的新特性

|  |
| --- |
| ResourceManager HA：在apache hadoop 4.4 或者CDH5.0.0 版本之后，增加了ResourceMangerHA 特性，支持基于Zookeeper 的热主备切换。  磁盘容错：增加了几个对多磁盘非常友好地参数，这些参数允许YARN 更好地使用NodeManager 上的多块磁盘.  资源调度器：Fair Scheduler 增加了允许用户在线将一个应用程序从一个队列转移到另外一个队列。 |

11 如何使用mapReduce实现两个表的join?

|  |
| --- |
| reduce side join : 在map 阶段，map 函数同时读取两个文件File1 和File2，为了区分  两种来源的key/value 数据对，对每条数据打一个标签（tag）,比如：tag=0 表示来自文件File1，  tag=2 表示来自文件File2。  map side join : Map side join 是针对以下场景进行的优化：两个待连接表中，有一个表非常大，而另一个表非常小，以至于小表可以直接存放到内存中。这样，我们可以将小表复制多份，让每个map task 内存中存在一份（比如存放到hash table 中），然后只扫描大表：对于大表中的每一条记录key/value，在hash table 中查找是否有相同的key 的记录，如果有，  则连接后输出即可。  Semi Join : Semi Join，也叫半连接，是从分布式数据库中借鉴过来的方法。它的产生动机是：对于reduce side join，跨机器的数据传输量非常大，这成了join 操作的一个瓶颈，  如果能够在map 端过滤掉不会参加join 操作的数据，则可以大大节省网络IO。  reduce side join + BloomFilter : BloomFilter 最常见的作用是：判断某个元素是否在一个集合里面。它最重要的两个方法是：add() 和contains()。最大的特点是不会存在 falsenegative，即：如果contains()返回false，则该元素一定不在集合中，但会存在一定的 falsepositive，即：如果contains()返回true，则该元素一定可能在集合中。  参考网址：http://my.oschina.net/leejun2005/blog/95186?fromerr=cDqkUyup |

8 Hadoop的sequencefile的格式，并说明下什么是java序列化，如何实现java序列化?

|  |
| --- |
| SequenceFile文件是Hadoop用来存储二进制形式的key-value 对而设计的一种平面文件;Hadoop 的HDFS 和MapReduce子框架主要是针对大数据文件来设计的，在小文件的处理上不但效率低下，而且十分消耗磁盘空间(每一个小文件占用一个Block,HDFS 默认block大小为64M)。  解决办法通常是选择一个容器，将这些小文件组织起来统一存储。HDFS提供了两种类型的容器，分别是SequenceFile和MapFile。  SequenceFile的每条记录是可序列化的字符数组。  序列化是指将结构化的对象转化为字节流以便在网络上传输或写入到磁盘进行永久存 储的过程，反序列化是指将字节流转回结构化对象的过程. |

9 简述hadoop1与hadoop2 的架构异同.

|  |
| --- |
| 加入了yarn 解决了资源调度的问题。  加入了对zookeeper 的支持实现比较可靠的高可用.  http://blog.csdn.net/fenglibing/article/details/32916445 |

13 请描述mapReduce二次排序原理

|  |
| --- |
| 二次排序:就是首先按照第一字段排序，然后再对第一字段相同的行按照第二字段排序，注意不能破坏第一次排序的结果。  在Hadoop中，默认情况下是按照key进行排序。对于同一个key，reduce函数接收到的value list是按照value 排序的。  有两种方法进行二次排序，分别为：buffer and in memory sort和value-to-key conversion。  对于buffer and in memory sort，主要思想是：在reduce()函数中，将某个key对应的所有value保存下来，然后进行排序。 这种方法最大的缺点是：可能会造成 out of memory。  对于value-to-key conversion，主要思想是：将key和部分value拼接成一个组合key（实现WritableComparable接口或者调用setSortComparatorClass函数），这样reduce获取的结果便是先按key排序，后按value排序的结果，需要注意的是，用户需要自己实现Paritioner，以便只按照key进行数据划分。Hadoop显式的支持二次排序，在Configuration 类中有个 setGroupingComparatorClass()方法，可用于设置排序 group的 key 值。  具体参考：http://www.cnblogs.com/xuxm2007/archive/2011/09/03/2165805.html  参考网址：http://my.oschina.net/leejun2005/blog/95186?fromerr=cDqkUyup |

14 请描述mapReduce中排序发生的几个阶段

|  |
| --- |
| 一个是在map side发生在spill后partition前。  一个是在reduce side发生在copy后 reduce前。  mapReduce的四个阶段：  Splitting :在进行map 计算之前，mapreduce 会根据输入文件计算输入分片（inputsplit），每个输入分片（input split）针对一个map 任务。输入分片（input split）存储的并非数据本身，而是一个分片长度和一个记录数据位置的数组，输入分片（input split）往往和hdfs 的block（块）关系很密切，假如我们设定hdfs 的块的大小是64mb，如果我们输入有三个文件，大小分别是3mb、65mb 和127mb，那么mapreduce 会把3mb 文件分为一个输入分片（input split），65mb 则是两个输入分片（input split）而127mb 也是两个输入分片（input split），换句话说我们如果在map 计算前做输入分片调整，例如合并小文件，那么就会有5 个map 任务将执行，而且每个map 执行的数据大小不均，这个也是mapreduce 优化计算的一个关键点。  Mapping:就是程序员编写好的map 函数了，因此map 函数效率相对好控制，而且一般map 操作都是本地化操作也就是在数据存储节点上进行；  Shuffle:描述着数据从map task 输出到reduce task 输入的这段过程。  Reduce:对Map阶段多个文件的数据进行合并。 |

15 请描述mapReduce中shuffle阶段的工作流程，如何优化shuffle阶段

|  |
| --- |
| 分区，排序，溢写，拷贝到对应reduce机器上，增加combiner，压缩溢写的文件。 |

16 请描述mapReduce 中combiner 的作用是什么，一般使用情景，哪些情况不需要？

|  |
| --- |
| 在MR作业中的Map阶段会输出结果数据到磁盘中。  Combiner只应该适用于那种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，也可以用于过滤数据，在 map端将无效的数据过滤掉。  在这些需求场景下，输出的数据是可以根据key值来作合并的，合并的目的是减少输出的数据量，减少IO的读写，减少网络传输,以提高MR的作业效率。    1.combiner的作用就是在map端对输出先做一次合并,以减少传输到reducer的数据量.  2.combiner最基本是实现本地key的归并,具有类似本地reduce,那么所有的结果都是reduce完成,效率会相对降低。  3.使用combiner,先完成的map会在本地聚合,提升速度. |

3. 简述Hadoop的几个默认端口及其含义

|  |
| --- |
| dfs.namenode.http-address:50070  dfs.datanode.address:50010  fs.defaultFS:8020  yarn.resourcemanager.webapp.address:8088 |

5. 谈谈你对MapReduce的优化建议。

|  |
| --- |
| 一.mapper调优  mapper调优主要就一个目标：减少输出量。  我们可以通过增加combine阶段以及对输出进行压缩设置进行mapper调优。  1>combine合并：      实现自定义combine要求继承reducer类。比较适合map的输出是数值型的，方便进行统计。  2>压缩设置：      在提交job的时候分别设置启动压缩和指定压缩方式。    二.reducer调优   reducer调优主要是通过参数调优和设置reducer的个数来完成。   reducer个数调优：    要求：一个reducer和多个reducer的执行结果一致，不能因为多个reducer导致执行结果异常。  规则：一般要求在hadoop集群中的执行mr程序，map执行完成100%后，尽量早的看到reducer执行到33%，可以通过命令hadoop job -status job\_id或者web页面来查看。     原因：map的执行process数是通过inputformat返回recordread来定义的；而reducer是有三部分构成的，分别为读取mapper输出数据、合并所有输出数据以及reduce处理，其中第一步要依赖map的执行，所以在数据量比较大的情况下，一个reducer无法满足性能要求的情况下，我们可以通过调高reducer的个数来解决该问题。  优点：充分利用集群的优势。  缺点：有些mr程序没法利用多reducer的优点，比如获取top n的mr程序。 |

hadoop家族如此强大，为什么还要结合R语言？

|  |
| --- |
| hadoop和R语言在各自的领域都相当的强大，由RevolutionAnalytics发起的一个开源项目RHadoop将R语言与Hadoop结合在一起，很好发挥了R语言特长。广大R语言爱好者借助强大工具RHadoop，可以在大数据领域大展拳脚。  1. Hadoop家族的强大之处，在于对大数据的处理，让原来的不可能（TB,PB数据量计算），成为了可能。  2.R语言的强大之处，在于统计分析，在没有Hadoop之前，我们对于大数据的处理，要取样本，假设检验，做回归，长久以来R语言都是统计学家专属的工具。  3. hadoop重点是全量数据分析，而R语言重点是样本数据分析。 两种技术放在一起，刚好是取长补短。     4.用R语言，通过分析少量数据，对业务目标回归建模，并定义指标，用Hadoop从海量日志数据中，提取指标数据，用R语言模型，对指标数据进行测试和调优  5.R和Hadoop分别都起着非常重要的作用，以计算机开发人员的思路，所有有事情都用Hadoop去做，没有数据建模和证明，”预测的结果”一定是有问题的。以统计人员的思路，所有的事情都用R去做，以抽样方式，得到的“预测的结果”也一定是有问题的。所以让二者结合，是产界业的必然的导向，也是产界业和学术界的交集，同时也为交叉学科的人才提供了更广阔的发展空间。 |

5.Mapreduce的工作原理，请举例子说明mapreduce是怎么运行的？

|  |
| --- |
| 离线计算框架，过程分为split map shuffle reduce四个过程 。  架构节点有：Jobtracker TaskTracker 。  Split将文件分割，传输到mapper，mapper接收KV形式的数据，经过处理，再传到shuffle过程。  Shuffle先进行HashPartition或者自定义的partition，会有数据倾斜和reduce的负载均衡问题；再进行排序，默认按字典排序；为减少mapper输出数据，再根据key进行合并，相同key的数据value会被合并；最后分组形成（key,value{}）形式的数据，输出到下一阶段 。  Reduce输入的数据就变成了，key+迭代器形式的数据，再进行处理。 |

2.两个类TextInputFormat和KeyValueInputFormat的区别是什么？

|  |
| --- |
| 相同点：  TextInputformat和KeyValueTextInputFormat都继承了FileInputFormat类，都是每一行作为一个记录；  区别：  TextInputformat将每一行在文件中的起始偏移量作为 key，每一行的内容作为value。默认以\n或回车键作为一行记录。  KeyValueTextInputFormat 适合处理输入数据的每一行是两列，并用 tab 分离的形式。 |

3.在一个运行的 Hadoop 任务中，什么是 InputSplit？

|  |
| --- |
| 在执行 mapreduce 之前，原始数据被分割成若干个 split，每个 split 作为一个 map  任务的输入，在 map 执行过程中 split 会被分解成一个个记录（key-value 对）。 |

4.Hadoop 框架中文件拆分是怎么被调用的？

|  |
| --- |
| InputFormat 是 MapReduce 中一个很常用的概念是文件拆分必须实现的一个接口，包含了两个方法：  public interface InputFormat<K, V> {  InputSplit[] getSplits(JobConf job, int numSplits) throws IOException;  RecordReader<K, V> createRecordReader(InputSplit split, TaskAttemptContext  context) throws IOException;  }  这两个方法分别完成以下工作：  方法 getSplits 将输入数据切分成 splits，splits 的个数即为 map tasks 的个数，  splits 的大小默认为块大小，即 64M  方法 getRecordReader 将每个 split 解析成 records, 再依次将 record 解析成  <K,V>对，也就是说 InputFormat 完成以下工作：InputFile --> splits --> <K,V>  系 统 常 用 的 InputFormat 又 有 哪 些 呢 ？ TextFileInputFormat ，  KeyValueTextFileInputFormat，SequenceFileInputFormat<key,value>,  NLineInputFormat 其中 Text InputFormat 便是最常用的，它的 <K,V>就代表 <行偏移,  该行内容>  然而系统所提供的这几种固定的将 InputFile 转换为 <K,V>的方式有时候并不能满足我  们的需求：  此时需要我们自定义 InputFormat ，从而使 Hadoop 框架按照我们预设的方式来将  InputFile 解析为<K,V>  在领会自定义 InputFormat 之前，需要弄懂一下几个抽象类、接口及其之间的关系：  InputFormat(interface), FileInputFormat(abstract class), TextInputFormat(class),  RecordReader (interface), Line RecordReader(class)的关系FileInputFormat implements InputFormat  TextInputFormat extends FileInputFormat  TextInputFormat.get RecordReader calls Line RecordReader  Line RecordReader implements RecordReader  对于 InputFormat 接口，上面已经有详细的描述  再看看 FileInputFormat，它实现了 InputFormat 接口中的 getSplits 方法，而将  getRecordReader 与 isSplitable 留给具体类(如 TextInputFormat )实现， isSplitable 方  法通常不用修改，所以只需要在自定义的 InputFormat 中实现  getRecordReader 方 法 即 可 ， 而 该 方 法 的 核 心 是 调 用 Line RecordReader( 即 由  LineRecorderReader 类来实现 " 将每个 s plit 解析成 records, 再依次将 record 解析成  <K,V>对" )，该方法实现了接口 RecordReader  public interface RecordReader<K, V> {  boolean next(K key, V value) throws IOException;  K createKey();  V createValue();  long getPos() throws IOException;  public void close() throws IOException;  float getProgress() throws IOException;  }  定义一个 InputFormat 的核心是定义一个类似于LineRecordReader的RecordReader |

5.参考下面的M/R系统的场景：HDFS 块大小为64MB；输入类型为FileInputFormat；有三个文件大小分别是：

64KB65MB 127MB。Hadoop框架会把这些文件拆分为多少块？

|  |
| --- |
| 分别为 1 块，2 块，2 块。 |

6.Hadoop 中 RecordReader 的作用是什么?

|  |
| --- |
| Hadoop 的 MapReduce 框架来处理数据，主要是面向海量大数据，对于这类数据，Hadoop 能够使其真正发挥其能力。对于海量小文件，不是说不能使用 Hadoop 来处理，只不过直接进行处理效率不会高，实际应用中，我们在使用 Hadoop 进行计算的时候，需要考虑将小数据转换成大数据，比如通过合并压缩等方法，我们通过自定义 InputFormat和RecordReader来实现对海量小文件的并行处理。 |

8.如果没有定义 partitioner，那数据在被送达 reducer 前

是如何被分区的？

|  |
| --- |
| 如果没有自定义的 partitioning，则默认的 partition 算法，即根据每一条数据的 key  的 hashcode 值摸运算（%）reduce 的数量，得到的数字就是“分区号“。 |

7.Map 阶段结束后， Hadoop 框架会处理： Partitioning，Shuffle 和 Sort，在这个阶段都发生了什么？

|  |
| --- |
| map task上的洗牌（shuffle）结束，此时 reducer task 上的洗牌开始，抓取 fetch 所  属于自己分区的数据，同时将这些分区的数据进行排序sort（默认的排序是根据每一条数据的键的字典  排序），进而将数据进行合并merge，即根据key相同的，将其 value 组成一个集合，最后输出结果。 |

5. 请列出你所知道的Hadoop调度器，并简要说明其工作方法。

|  |
| --- |
| （1）默认的调度器 FIFO  Hadoop中默认的调度器，它先按照作业的优先级高低，再按照到达时间的先后选择被执行的作业。  （2）计算能力调度器 Capacity Scheduler  支持多个队列，每个队列可配置一定的资源量，每个队列采用FIFO调度策略，为了防止同一个用户的作业独占队列中的资源，该调度器会对同一用户提交的作业所占资源量进行限定。调度时，首先按以下策略选择一个合适队列：计算每个队列中正在运行的任务数与其应该分得的计算资源之间的比值，选择一个该比值最小的队列；然后按以下策略选择该队列中一个作业：按照作业优先级和提交时间顺序选择，同时考虑用户资源量限制和内存限制。  （3）公平调度器 Fair Scheduler  同计算能力调度器类似，支持多队列多用户，每个队列中的资源量可以配置，同一队列中的作业公平共享队列中所有资源。实际上， Hadoop的调度器远不止以上三种，最近，出现了很多针对新型应用的Hadoop调度器。  （4）适用于异构集群的调度器LATE  现有的 Hadoop 调度器都是建立在同构集群的假设前提下，具体假设如下：  1）集群中各个节点的性能完全一样  2）对于 reduce task，它的三个阶段：copy、 sort 和 reduce，用时各占 1/3  3）同一 job 的同类型的 task 是一批一批完成的，他们用时基本一样。  现有的Hadoop调度器存在较大缺陷，主要体现在探测落后任务的算法上：如果一个task 的进度落后于同类型 task进度的20%，则把该task当做落后任务(这种任务决定了job的完成时间，需尽量缩短它的执行时间)，从而为它启动一个备份任务（speculative task）。  如果集群异构的，对于同一个task，即使是在相同节点上的执行时间也会有较大差别，因而在异构集群中很容易产生大量的备份任务。  LATE调度器从某种程度上解决了现有调度器的问题，它定义三个阈值 ：  SpeculativeCap，系统中最大同时执行的speculative task 数目（作者推荐值为总slot数的10%）； SlowNodeThreshold（作者推荐值为25%）：得分（分数计算方法见论文）低于该阈值的node（快节点）上不会启动 speculative task；SlowTaskThreshold（作者推荐值为 25%）：当 task 进度低于同批同类 task 的平均进度的 SlowTaskThreshold 时，会为该 task 启动 speculative task。它的调度策略是：当一个节点出现空闲资源且系统中总的备份任务数小 于 SpeculativeCap 时， a.如果该节点是慢节点（节点得分高 于SlowNodeThreshold），则忽略这个请求。 b.对当前正在运行的 task 按估算的剩余完成时间排序 c.选择剩余完成时间最大且进度低于 SlowTaskThreshold 的 task，为该 task 启动备份任务。  （5）适用于实时作业的调度器Deadline Scheduler和Constraint-based Scheduler  这种调度器主要用于有时间限制的作业（Deadline Job），即给作业一个deadline时间，让它在该时间内完成。实际上，这类调度器分为两种，软实时（允许作业有一定的超时）作业调度器和硬实时（作业必须严格按时完成）作业调度器。  Deadline Scheduler 主要针对的是软实时作业，该调度器根据作业的运行进度和剩余时间动态调整作业获得的资源量，以便作业尽可能的在 deadline 时间内完成。  Constraint-based Scheduler 主要针对的是硬实时作业，该调度器根据作业的deadline和当前系统中的实时作业运行情况，预测新提交的实时作 业能不能在 deadline时间内完成，如果不能，则将作业反馈给用户，让他重调整作业的 deadline。 |

13.有可能使 Hadoop 任务输出到多个目录中么？如果可以，怎么做？

|  |
| --- |
| Hadoop 内置的输出文件格式有：  MultipleOutputs<K,V> 可以把输出数据输送到不同的目录；  在自定义的reduce的函数中首先使用setup函数（注：该函数在task启动后数据处理前就调用一次）new出MultipleOutputs 对象，利用该对象调用 write 方法定义输出的目录。 |

12.Hadoop API 中的什么特性可以使 map reducer 任务以不同语言（如 Perl，ruby，awk 等）实现灵活性？

|  |
| --- |
| Hadoop序列化机制，支持多语言的交互。 |

1.3Hadoop集群有几种角色的节点，每个节点对应的进程有哪些？

|  |
| --- |
| Hadoop 集群从三个角度划分为两个角色。 Hadoop 集群包括 hdfs 作为文件存储系统，mapreduce 作为分布式计算框架。  (1)最基本的划分为主节点和从节点  (2)在 hdfs 中，Namenode 作为主节点，接收客户端的读写服务。 存储文件的的元数据如文件名，文件目录结构，文件属性(生成时间，副本数，文件权限)，以及 block所在的datanode 等等，namenode 的元数据信息会在启动后加载到内存，metadata存储到磁盘文件名为” fsimage” ,block 的信息不会保存到 fsimage,edits 记录对metadata 的操作日志datanode 作为从节点。 Datanode 的作用是存储数据即 block，启动 datanode 线程的时候会向 namenode 汇报 block 信息，通过向 namenode 发送心跳保持与其联系，如果 namenode10 分钟没有收到 datanode 的心跳，则认为 datanode 挂了，并复制其上的 block 到其他 datanode。  (3)在 mapreduce 中的主节点为 jobTracker 和 tasktracker。  JobTracker 作为主节点，负责调度分配每一个子任务 task 运行于 taskTracker上，如果发现有失败的task 就重新分配到其他节点，每个 Hadoop 集群一般有一个JobTaskTracker，运行在 master 节点上。  TaskTracker主动与JobTracker通信，接收作业，并负责直接执行每一个任务，为了减少网络带宽TaskTracker 最好运行在 datanode 上 |

1.7mapreduce 的优化方法有哪些?

|  |
| --- |
| Mapreduce 程序效率的瓶颈在于两点：  一计算机性能  二 I/O 操作优化  优化分为时间和空间两种常见的优化策略如下：  1,输入的文件尽量使用大文件，众多的小文件会导致map的数量众多，每个map任务都会造成一些性能的损失，如果输入的是小的文件可以在进行mapreduce处理之前整合成为大文件，或者直接采用ConbinFileInputFormat来作为输入方式，此时，hadoop 会考虑节点和集群的位置信息，决定将哪些文件打包到一个单元中。  2,合理的分配 map 和 reduce 的任务的数量.  3,压缩中间数据，减少 I/O.  4,在 map 之后先进行 combine 处理，减少 I/O. |

1.10 列举你了解的海量数据的处理方法及适用范围，如果有相关使用经验，可简要说明。

|  |
| --- |
| mapreduce 分布式计算 mapreduce 的思想就是分而治之  倒排索引:一种索引方法，用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射，在倒排索引中单词指向了包含单词的文档。  消息队列:大量的数据写入首先存入消息队列进行缓冲，再把消息队列作为数据来源进行数据读取。  数据库读写分离:向一台数据库写入数据，另外的多台数据库从这台数据库中进行读取。 |

2 编程题MR作业设计: 有两种格式的海量日志文件存放于hdfs上，可以通过文件名前缀区分，其中登录日志格式：

|  |
| --- |
| user,ip,time,oper(枚举值:1 为上线， 2 位下线 )；  访问日志格式:ip,time,url,假设登录日志中user上下线信息完整，且同一上下线时间段内使用的ip唯一，要计算访问日志中独立 user 数量最多的前 10个url，请给出MapReduce设计思路，如需要几个MapReduce，及每个 MapReduce 算法伪代码。  思路:  将 userid ,ip ,url 封装成一个对象，map 类输入数据，按照 userid 为 key 输出，value  为 1（intwritable）,按照 reduce 计算写一个排序类然后按照 value 降序输出前十个。 |

2. 运行一个 hadoop 任务的流程是什么样的。

|  |
| --- |
| 1、 导入数据对需分析的数据进行分片，片的大小默认与 datanode 块大小相同。  2、 每个数据片由一个 mapper 进行分析，mapper 按照需求将数据拆分为一个个 keyvalue 格式的数据。  3、 每个 key-value 数据调用一次 map 方法，对数据进行相应的处理后输出。  4、 将输出的数据复制到对应的分区，默认一个键一个区，相同键放在同一个区中。  5、 将输出的数据进行合并为 key-Iterable 格式。  6、 每个分区有一个 reduce，每个 reduce 将同一个分区的数据进行合并处理为自己所需的数据格式。  7、 将数据输出至 hdfs。 |

3. secondary namenode / HAnamenode 是如何工作的?

|  |
| --- |
| 1、 secondary namenode 在 namenode 的 edits log 文件超过规定值（默认 64M）时，或者每隔3600秒（默认值）时会剪切 edits log 文件，复制 namenode 上的 fsimage文件（若没有 fsimage）。  2、 namenode 节点创建一个新的 edits 文件。  3、 将 edits log 文件及 fsimage 文件进行合并为一个新 fsimage。  4、 将 fsimage 文件回传至 namenode 节点。  5、 Namenode 节点使用 seconday namenode 节点回传的 fsimage 文件覆盖本地的fsimage 文件。  6、 当 namenode 节点启动时可以从 seconday namenode 节点复制 fsimage 文件达到快速启动的目的。 |

4. 什么样的计算不能用 mr 来提速，举 5 个例子。

|  |
| --- |
| 1、 数据量很小。  2、 繁杂的小文件。  3、 索引是更好的存取机制的时候。  4、 事务处理。  5、 只有一台机器的时候。 |

5. 一个 mr 作业跑的比较慢，如何来优化。至少给出 6 个方案。

|  |
| --- |
| mr跑的慢可能有很多原因，如：数据倾斜、map和reduce数设置不合理、reduce等待过久、小文件过多、spill 次数过多、 merge 次数过多等。  1、解决数据倾斜：数据倾斜可能是partition不合理，导致部分partition中的数据过多，部分过少。可通过分析数据，自定义分区器解决。  2、合理设置map和reduce数：两个都不能设置太少，也不能设置太多。太少，会导致task等待，延长处理时间；太多，会导致 map、 reduce 任务间竞争资源，造成处理超时等错误。  3、设置map、reduce共存：调整slowstart.completedmaps参数，使map运行到一定程度后，reduce也开始运行，减少 reduce 的等待时间。  4、合并小文件：在执行mr任务前将小文件进行合并，大量的小文件会产生大量的map任务，增大map任务装载次数，而任务的装载比较耗时，从而导致 mr 运行较慢。  5、减少spill次数：通过调整io.sort.mb及sort.spill.percent参数值，增大触发spill的内存上限，减少spill 次数，从而减少磁盘 IO。  6、减少merge次数：通过调整io.sort.factor参数，增大merge的文件数目，减少merge的次数，从而缩短mr处理时间。 |

6. Hadoop会有哪些重大故障，如何应对？至少给出 5个。

|  |
| --- |
| 1、 namenode 单点故障：通过 zookeeper 搭建 HA 高可用，可自动切换 namenode。  2、ResourceManager单点故障：可通过配置YARN的HA，并在配置的namenode上手动启动ResourceManager作为Slave，在 Master 故障后，Slave 会自动切换为Master。  3、reduce阶段内存溢出：是由于单个reduce任务处理的数据量过多，通过增大reducetasks数目、优化partition 规则使数据分布均匀进行解决。  4、datanode内存溢出：是由于创建的线程过多，通过调整linux的maxuserprocesses参数，增大可用线程数进行解决。  5、 集群间时间不同步导致运行异常：通过配置内网时间同步服务器进行解决。 |

7. 什么情况下会触发 recovery 过程， recover 是怎么做的。

|  |
| --- |
| 当 jobtracker.restart.recover 参数值设置为 true， jobtracker 重启之时会触发recovery。  在JobTracker重启前，会在history log中记录各个作业的运行状态，这样在JobTracker关闭后，系统中所有数据目录、 临时目录均会被保留，待 JobTracker 重启之后，JobTracker 自动重新提交这些作业，并只对未运行完成的 task 进行重新调度，这样可避免已经计算完的 task 重新计算。 |

8. 你认为 hadoop 有哪些设计不合理的地方。

|  |
| --- |
| 1、 不支持文件的并发写入和对文件内容的随机修改。  2、 不支持低延迟、高吞吐的数据访问。  3、 存取大量小文件，会占用 namenode 大量内存，小文件的寻道时间超过读取时间。  4、 hadoop 环境搭建比较复杂。  5、 数据无法实时处理。  6、 mapreduce 的 shuffle 阶段 IO 太多。  7、 编写 mapreduce 难度较高，实现复杂逻辑时，代码量太大。 |