大数据面试题

目录

[大数据面试题 1](#_Toc3299833)

[SparkCore面试篇01 17](#_Toc3299834)

[1.Sparkmaster使用zookeeper进行HA的，有哪些元数据保存在Zookeeper？ 17](#_Toc3299835)

[2.SparkmasterHA主从切换过程不会影响集群已有的作业运行，为什么？ 18](#_Toc3299836)

[3.SparkonMesos中，什么是的粗粒度分配，什么是细粒度分配，各自的优点和缺点是什么？ 18](#_Toc3299837)

[4.如何配置sparkmaster的HA？ 18](#_Toc3299838)

[5.ApacheSpark有哪些常见的稳定版本，Spark1.6.0的数字分别代表什么意思？ 19](#_Toc3299839)

[6.driver的功能是什么？ 20](#_Toc3299840)

[7.spark的有几种部署模式，每种模式特点？ 20](#_Toc3299841)

[8.Spark技术栈有哪些组件，每个组件都有什么功能，适合什么应用场景？ 22](#_Toc3299842)

[9.Spark中Work的主要工作是什么？ 23](#_Toc3299843)

[10.Spark为什么比mapreduce快？ 23](#_Toc3299844)

[11.简单说一下hadoop和spark的shuffle相同和差异？ 24](#_Toc3299845)

[12.Mapreduce和Spark的都是并行计算，那么他们有什么相同和区别 25](#_Toc3299846)

[13.RDD机制？ 26](#_Toc3299847)

[14、spark有哪些组件？ 26](#_Toc3299848)

[15、spark工作机制？ 27](#_Toc3299849)

[16、spark的优化怎么做？ 27](#_Toc3299850)

[17.简要描述Spark分布式集群搭建的步骤 28](#_Toc3299851)

[18.什么是RDD宽依赖和窄依赖？ 28](#_Toc3299852)

[19.spark-submit的时候如何引入外部jar包 29](#_Toc3299853)

[20.cache和pesist的区别 29](#_Toc3299854)

[参考资料 30](#_Toc3299855)

[Sparkcore面试篇02 30](#_Toc3299856)

[1.cache后面能不能接其他算子,它是不是action操作？ 30](#_Toc3299857)

[2.reduceByKey是不是action？ 30](#_Toc3299858)

[3.数据本地性是在哪个环节确定的？ 30](#_Toc3299859)

[4.RDD的弹性表现在哪几点？ 31](#_Toc3299860)

[5.常规的容错方式有哪几种类型？ 31](#_Toc3299861)

[6.RDD通过Linage（记录数据更新）的方式为何很高效？ 31](#_Toc3299862)

[7.RDD有哪些缺陷？ 32](#_Toc3299863)

[8.说一说Spark程序编写的一般步骤？ 32](#_Toc3299864)

[9.Spark有哪两种算子？ 32](#_Toc3299865)

[10.Spark提交你的jar包时所用的命令是什么？ 33](#_Toc3299866)

[11.Spark有哪些聚合类的算子,我们应该尽量避免什么类型的算子？ 33](#_Toc3299867)

[12.你所理解的Spark的shuffle过程？ 33](#_Toc3299868)

[13.你如何从Kafka中获取数据？ 33](#_Toc3299869)

[14.对于Spark中的数据倾斜问题你有什么好的方案？ 34](#_Toc3299870)

[15.RDD创建有哪几种方式？ 35](#_Toc3299871)

[16.Spark并行度怎么设置比较合适 35](#_Toc3299872)

[17.Spark中数据的位置是被谁管理的？ 35](#_Toc3299873)

[18.Spark的数据本地性有哪几种？ 36](#_Toc3299874)

[19.rdd有几种操作类型？ 36](#_Toc3299875)

[20.Spark如何处理不能被序列化的对象？ 36](#_Toc3299876)

[21.collect功能是什么，其底层是怎么实现的？ 37](#_Toc3299877)

[22.Spaek程序执行，有时候默认为什么会产生很多task，怎么修改默认task执行个数？ 37](#_Toc3299878)

[23.为什么SparkApplication在没有获得足够的资源，job就开始执行了，可能会导致什么什么问题发生? 38](#_Toc3299879)

[24.map与flatMap的区别 38](#_Toc3299880)

[25.列举你常用的action？ 38](#_Toc3299881)

[26.Spark为什么要持久化，一般什么场景下要进行persist操作？ 39](#_Toc3299882)

[27.为什么要进行序列化 40](#_Toc3299883)

[28.介绍一下join操作优化经验？ 40](#_Toc3299884)

[29.介绍一下cogrouprdd实现原理，你在什么场景下用过这个rdd？ 40](#_Toc3299885)

[30下面这段代码输出结果是什么？ 41](#_Toc3299886)

[二、参考资料 42](#_Toc3299887)

[Sparkcore面试篇03 42](#_Toc3299888)

[1.Spark使用parquet文件存储格式能带来哪些好处？ 42](#_Toc3299889)

[2.Executor之间如何共享数据？ 43](#_Toc3299890)

[3.Spark累加器有哪些特点？ 43](#_Toc3299891)

[4.如何在一个不确定的数据规模的范围内进行排序？ 44](#_Toc3299892)

[5.sparkhashParitioner的弊端是什么？ 44](#_Toc3299893)

[6.RangePartitioner分区的原理? 44](#_Toc3299894)

[7.介绍parition和block有什么关联关系？ 45](#_Toc3299895)

[8.Spark应用程序的执行过程是什么？ 45](#_Toc3299896)

[9.hbase预分区个数和spark过程中的reduce个数相同么 46](#_Toc3299897)

[10.如何理解Standalone模式下，Spark资源分配是粗粒度的？ 46](#_Toc3299898)

[11.Spark如何自定义partitioner分区器？ 46](#_Toc3299899)

[12.spark中task有几种类型？ 47](#_Toc3299900)

[13.union操作是产生宽依赖还是窄依赖？ 47](#_Toc3299901)

[14.rangePartioner分区器特点？ 47](#_Toc3299902)

[15.什么是二次排序，你是如何用spark实现二次排序的？（互联网公司常面） 48](#_Toc3299903)

[16.如何使用Spark解决TopN问题？（互联网公司常面） 48](#_Toc3299904)

[17.如何使用Spark解决分组排序问题？（互联网公司常面） 48](#_Toc3299905)

[18.窄依赖父RDD的partition和子RDD的parition是不是都是一对一的关系？ 50](#_Toc3299906)

[19.Hadoop中，Mapreduce操作的mapper和reducer阶段相当于spark中的哪几个算子？ 50](#_Toc3299907)

[20.什么是shuffle，以及为什么需要shuffle？ 50](#_Toc3299908)

[21.不需要排序的hashshuffle是否一定比需要排序的sortshuffle速度快？ 50](#_Toc3299909)

[22.Spark中的HashShufle的有哪些不足？ 51](#_Toc3299910)

[23.conslidate是如何优化Hashshuffle时在map端产生的小文件？ 51](#_Toc3299911)

[24.Sort-basesdshuffle产生多少个临时文件 52](#_Toc3299912)

[25.Sort-basedshuffle的缺陷? 52](#_Toc3299913)

[26.Sparkshell启动时会启动derby? 52](#_Toc3299914)

[27.spark.default.parallelism这个参数有什么意义，实际生产中如何设置？ 53](#_Toc3299915)

[28.spark.storage.memoryFraction参数的含义,实际生产中如何调优？ 53](#_Toc3299916)

[29.spark.shuffle.memoryFraction参数的含义，以及优化经验？ 54](#_Toc3299917)

[30.介绍一下你对UnifiedMemoryManagement内存管理模型的理解？ 54](#_Toc3299918)

[二、参考资料 56](#_Toc3299919)

[SparkonYarn面试篇04 56](#_Toc3299920)

[1.MRV1有哪些不足？ 57](#_Toc3299921)

[2.描述Yarn执行一个任务的过程？ 57](#_Toc3299922)

[3.Yarn中的container是由谁负责销毁的，在HadoopMapreduce中container可以复用么？ 58](#_Toc3299923)

[4.提交任务时，如何指定SparkApplication的运行模式？ 58](#_Toc3299924)

[5.不启动Spark集群Master和work服务，可不可以运行Spark程序？ 59](#_Toc3299925)

[6.Spark中的4040端口由什么功能? 59](#_Toc3299926)

[7.sparkonyarnCluster模式下，ApplicationMaster和driver是在同一个进程么？ 59](#_Toc3299927)

[8.如何使用命令查看application运行的日志信息 59](#_Toc3299928)

[9.SparkonYarn模式有哪些优点？ 60](#_Toc3299929)

[10.谈谈你对container的理解？ 60](#_Toc3299930)

[11.运行在yarn中Application有几种类型的container？ 61](#_Toc3299931)

[12.SparkonYarn架构是怎么样的？（要会画哦，这个图） 61](#_Toc3299932)

[13.Executor启动时，资源通过哪几个参数指定？ 62](#_Toc3299933)

[14.为什么会产生yarn，解决了什么问题，有什么优势? 62](#_Toc3299934)

[15.Mapreduce的执行过程? 62](#_Toc3299935)

[16.一个task的map数量由谁来决定？ 62](#_Toc3299936)

[17.reduce后输出的数据量有多大？ 63](#_Toc3299937)

[18.你的项目提交到job的时候数据量有多大？ 63](#_Toc3299938)

[19.你们提交的job任务大概有多少个？这些job执行完大概用多少时间？ 64](#_Toc3299939)

[20.你们业务数据量多大？有多少行数据？ 64](#_Toc3299940)

[22.如何杀死一个正在运行的job 64](#_Toc3299941)

[23.列出你所知道的调度器，说明其工作原理 64](#_Toc3299942)

[24.YarnClient模式下，执行SparkSQL报这个错，Exceptioninthread"Thread-2"java.lang.OutOfMemoryError:PermGenspace，但是在YarnCluster模式下正常运行，可能是什么原因？ 65](#_Toc3299943)

[25.spark.driver.extraJavaOptions这个参数是什么意思，你们生产环境配了多少？ 65](#_Toc3299944)

[26.导致Executor产生FULLgc的原因，可能导致什么问题？ 66](#_Toc3299945)

[27.Combiner和partition的作用 66](#_Toc3299946)

[28.Spark执行任务时出现java.lang.OutOfMemoryError:GCoverheadlimitexceeded和java.lang.OutOfMemoryError:javaheapspace原因和解决方法？ 67](#_Toc3299947)

[29.请列出在你以前工作中所使用过的开发map/reduce的语言 67](#_Toc3299948)

[30.你认为/etc/hosts配置错误，会对集群有什么影响？ 67](#_Toc3299949)

[二、参考资料 68](#_Toc3299950)

[SparkCore面试篇05 68](#_Toc3299951)

[1.scala中private与private[this]修饰符的区别？ 69](#_Toc3299952)

[2.scala中内部类和java中的内部类区别 69](#_Toc3299953)

[3.Spark中standalone模式特点，有哪些优点和缺点？ 69](#_Toc3299954)

[4.FIFO调度模式的基本原理、优点和缺点？ 69](#_Toc3299955)

[5.FAIR调度模式的优点和缺点？ 69](#_Toc3299956)

[6.CAPCACITY调度模式的优点和缺点？ 69](#_Toc3299957)

[7.列举你了解的序列化方法，并谈谈序列化有什么好处？ 69](#_Toc3299958)

[8.常见的数压缩方式，你们生产集群采用了什么压缩方式，提升了多少效率？ 69](#_Toc3299959)

[9.简要描述Spark写数据的流程？ 69](#_Toc3299960)

[10.Spark中Lineage的基本原理 69](#_Toc3299961)

[11.使用shell和scala代码实现WordCount？ 69](#_Toc3299962)

[12.请列举你碰到的CPU密集型的应用场景，你有做哪些优化？ 69](#_Toc3299963)

[13.SparkRDD和MR2的区别 69](#_Toc3299964)

[14.Spark读取hdfs上的文件，然后count有多少行的操作，你可以说说过程吗。那这个count是在内存中，还是磁盘中计算的呢？ 70](#_Toc3299965)

[15.spark和Mapreduce快？为什么快呢？快在哪里呢？ 70](#_Toc3299966)

[16.sparkSql又为什么比hive快呢？ 70](#_Toc3299967)

[17.RDD的数据结构是怎么样的？ 70](#_Toc3299968)

[18.RDD算子里操作一个外部map比如往里面put数据。然后算子外再遍历map。会有什么问题吗。 70](#_Toc3299969)

[19.hadoop的生态呢。说说你的认识。 70](#_Toc3299970)

[20.jvm怎么调优的，介绍你的SparkJVM调优经验？ 70](#_Toc3299971)

[21.jvm结构？堆里面几个区？ 70](#_Toc3299972)

[22.怎么用spark做数据清洗 70](#_Toc3299973)

[23.spark怎么整合hive？ 70](#_Toc3299974)

[24.spark读取数据，是几个Partition呢？ 70](#_Toc3299975)

[25.hbaseregion多大会分区，spark读取hbase数据是如何划分partition的？ 70](#_Toc3299976)

[26.画图，画Spark的工作模式，部署分布架构图 70](#_Toc3299977)

[27.画图，画图讲解spark工作流程。以及在集群上和各个角色的对应关系。 71](#_Toc3299978)

[28.java自带有哪几种线程池。 71](#_Toc3299979)

[29.画图，讲讲shuffle的过程。那你怎么在编程的时候注意避免这些性能问题？ 71](#_Toc3299980)

[30.BlockManager怎么管理硬盘和内存的？ 71](#_Toc3299981)

[SparkCore面试篇 71](#_Toc3299982)

[1.kafka收集数据的原理？ 72](#_Toc3299983)

[2.讲讲列式存储的parquet文件底层格式？ 72](#_Toc3299984)

[3.dataset和dataframe？ 72](#_Toc3299985)

[4scala中trait特征和用法？ 72](#_Toc3299986)

[5.redis和memcache的区别？ 72](#_Toc3299987)

[6.列举Spark中常见的端口，分别有什么功能？ 72](#_Toc3299988)

[7.Sparkmaster如何通过Zookeeper做HA？ 72](#_Toc3299989)

[8.Spark官网中，你常用哪几个模块？ 72](#_Toc3299990)

[9.你有见过哪些原因导致的数据倾斜，怎么解决？ 72](#_Toc3299991)

[10.简要描述宽依赖窄依赖以及各自的特点？ 72](#_Toc3299992)

[11.yarn的原理？ 72](#_Toc3299993)

[12.BlockManager怎么管理硬盘和内存的？ 72](#_Toc3299994)

[13.哪些算子操作涉及到shuffle1 72](#_Toc3299995)

[14.看过源码？你熟悉哪几个部分的源码？ 72](#_Toc3299996)

[15.集群上nodemanager和ResourceManager的数量关系？ 72](#_Toc3299997)

[16.Spark如何处理结构化数据，Spark如何处理非结构话数据？ 73](#_Toc3299998)

[17.Spark性能优化主要有哪些手段？ 73](#_Toc3299999)

[18.简要描述Spark分布式集群搭建的步骤？ 73](#_Toc3300000)

[19.对于Spark你觉得他对于现有大数据的现状的优势和劣势在哪里？ 73](#_Toc3300001)

[20.对于算法是否进行过自主的研究设计？ 73](#_Toc3300002)

[21.简要描述你了解的一些数据挖掘算法与内容 73](#_Toc3300003)

[22.什么时候join不发生shuffle？ 73](#_Toc3300004)

[23.sparkshuffle的具体过程，你知道几种shuffle方式 73](#_Toc3300005)

[24.spark如何防止内存溢出？ 73](#_Toc3300006)

[25.简述hadoop实现join的及各种方式？ 73](#_Toc3300007)

[26rdd转为dataFrame两种方式？ 73](#_Toc3300008)

[27.列举你熟悉的内存系统，各自的优缺点？ 73](#_Toc3300009)

[28.Spark中Master实现HA有哪些方式？ 73](#_Toc3300010)

[29函数式编程特点？ 73](#_Toc3300011)

[30.Sort-basedshuffle的缺陷？ 74](#_Toc3300012)

[面试篇07 74](#_Toc3300013)

[1、简答说一下hadoop的map-reduce编程模型 74](#_Toc3300014)

[2、hadoop的TextInputFormat作用是什么，如何自定义实现 75](#_Toc3300015)

[3、hadoop和spark的都是并行计算，那么他们有什么相同和区别 75](#_Toc3300016)

[4、为什么要用flume导入hdfs，hdfs的构架是怎样的 76](#_Toc3300017)

[5、map-reduce程序运行的时候会有什么比较常见的问题 76](#_Toc3300018)

[6、简单说一下hadoop和spark的shuffle过程 77](#_Toc3300019)

[7、Hive中存放是什么？ 77](#_Toc3300020)

[8、Hive与关系型数据库的关系？ 77](#_Toc3300021)

[9、Flume工作机制是什么？ 78](#_Toc3300022)

[10、Sqoop工作原理是什么？ 78](#_Toc3300023)

[11、Hbase行健列族的概念，物理模型，表的设计原则？ 78](#_Toc3300024)

[12、SparkStreaming和Storm有何区别？ 79](#_Toc3300025)

[13、mllib支持的算法？ 79](#_Toc3300026)

[14、简答说一下hadoop的map-reduce编程模型？ 80](#_Toc3300027)

[15、Hadoop平台集群配置、环境变量设置？ 80](#_Toc3300028)

[16、Hadoop性能调优？ 82](#_Toc3300029)

[17、Hadoop高并发？ 83](#_Toc3300030)

[25、kafka工作原理？ 83](#_Toc3300031)

[26、ALS算法原理？ 83](#_Toc3300032)

[27、kmeans算法原理？ 83](#_Toc3300033)

[28、canopy算法原理？ 84](#_Toc3300034)

[29、朴素贝叶斯分类算法原理？ 84](#_Toc3300035)

[30、关联规则挖掘算法apriori原理？ 84](#_Toc3300036)

[参考资料 84](#_Toc3300037)

[面试篇08-Spark错误日志面试集锦 84](#_Toc3300038)

[1、OperationcategoryREADisnotsupportedinstatestandby是什么原因导致的？org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.ipc.StandbyException):OperationcategoryREADisnotsupportedinstatestandby答：此时请登录Hadoop的管理界面查看运行节点是否处于standby 85](#_Toc3300039)

[2、不配置spark.deploy.recoveryMode选项为ZOOKEEPER，会有什么不好的地方 85](#_Toc3300040)

[3、多Master如何配置 85](#_Toc3300041)

[4、NoSpaceLeftonthedevice（Shuffle临时文件过多） 86](#_Toc3300042)

[5、java.lang.OutOfMemory,unabletocreatenewnativethread 86](#_Toc3300043)

[6、Worker节点中的work目录占用许多磁盘空间 87](#_Toc3300044)

[7、spark-shell提交SparkApplication如何解决依赖库 88](#_Toc3300045)

[8、Spark在发布应用的时候，出现连接不上master问题，如下 88](#_Toc3300046)

[9、开发spark应用程序（和Flume-NG结合时）发布应用时可能出现org.jboss.netty.channel.ChannelException:Failedtobindto:/192.168.10.156:18800 89](#_Toc3300047)

[10、spark-shell找不到hadoopso问题解决 90](#_Toc3300048)

[12、java.lang.IllegalArgumentException:java.net.UnknownHostException:dfscluster 90](#_Toc3300049)

[13、Exceptioninthread“main”java.lang.Exception:Whenrunningwithmaster‘yarn-client’eitherHADOOP\_CONF\_DIRorYARN\_CONF\_DIRmustbesetintheenvironment. 91](#_Toc3300050)

[14、Jobabortedduetostagefailure:Task3instage0.0failed4times,mostrecentfailure:Losttask3.3in 92](#_Toc3300051)

[15、长时间等待无反应，并且看到服务器上面的web界面有内存和核心数，但是没有分配，如下图 93](#_Toc3300052)

[16、内存不足或数据倾斜导致ExecutorLost（spark-submit提交） 94](#_Toc3300053)

[17、SparkStreaming和kafka整合后读取消息报错：OffsetOutOfRangeException 96](#_Toc3300054)

[18、java.io.IOException:Couldnotlocateexecutablenull\bin\winutils.exeintheHadoopbinaries.（sparksqlonhive任务引发HiveContextNullPointerException） 96](#_Toc3300055)

[19、Therootscratchdir:/tmp/hiveonHDFSshouldbewritable.Currentpermissionsare:rwx—— 98](#_Toc3300056)

[20、Exceptioninthread“main”org.apache.hadoop.security.AccessControlException:Permissiondenied:user=Administrator,access=WRITE,inode=”/data”:bdata:supergroup:drwxr-xr-x 98](#_Toc3300057)

[21、运行Spark-SQL报错：org.apache.spark.sql.AnalysisException:unresolvedoperator‘Project 99](#_Toc3300058)

[22.在$SPARK\_HOME/conf/spark-env.sh中设置这些变量好像也只是在terminal中的shell环境中才有效JAVA\_HOMEisnotsetException:Javagatewayprocessexitedbeforesendingthedriveritsportnumber 100](#_Toc3300059)

[23.ValueError:CannotrunmultipleSparkContextsatonce 100](#_Toc3300060)

[24.spark输出太多warningmessages 101](#_Toc3300061)

[25.org.apache.spark.shuffle.FetchFailedException，一般发生在有大量shuffle操作的时候,task不断的failed,然后又重执行，一直循环下去，非常的耗时 102](#_Toc3300062)

[26.Executor&TaskLost因为网络或者gc的原因,worker或executor没有接收到executor或task的心跳反馈WARNTaskSetManager:Losttask1.0instage0.0(TID1,aa.local):ExecutorLostFailure(executorlost) 102](#_Toc3300063)

[27.Master挂掉,standby重启也失效，如Master默认使用512M内存，当集群中运行的任务特别多时，就会挂掉，原因是master会读取每个task的eventlog日志去生成sparkui，内存不足自然会OOM，可以在master的运行日志中看到，通过HA启动的master自然也会因为这个原因失败。 103](#_Toc3300064)

[28.worker挂掉或假死有时候我们还会在webui中看到worker节点消失或处于dead状态，在该节点运行的任务则会报各种lostworker的错误，引发原因和上述大体相同，worker内存中保存了大量的ui信息导致gc时失去和master之间的心跳。 103](#_Toc3300065)

[29.报错：ERRORstorage.DiskBlockObjectWriter:Uncaughtexceptionwhilerevertingpartialwritestofile/hadoop/application\_1415632483774\_448143/spark-local-20141127115224-9ca8/04/shuffle\_1\_1562\_27 104](#_Toc3300066)

[30.报错：ERRORexecutor.CoarseGrainedExecutorBackend:DriverDisassociated[akka.tcp://sparkExecutor@pc-jfqdfx31:48586]->[akka.tcp://sparkDriver@pc-jfqdfx30:41656]disassociated!Shuttingdown. 104](#_Toc3300067)

[面试篇09-综合题 105](#_Toc3300068)

[1.给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url? 105](#_Toc3300069)

[2.有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M，要求返回频数最高的100个词。 106](#_Toc3300070)

[3.现有海量日志数据保存在一个超级大的文件中，该文件无法直接读入内存，要求从中提取某天出访问百度次数最多的那个IP。 107](#_Toc3300071)

[4.LVS和HAProxy相比，它的缺点是什么? 108](#_Toc3300072)

[5.Sqoop用起来感觉怎样? 111](#_Toc3300073)

[6.ZooKeeper的角色以及相应的Zookepper工作原理? 112](#_Toc3300074)

[7.HBase的Insert与Update的区别? 113](#_Toc3300075)

[8.请简述大数据的结果展现方式。 113](#_Toc3300076)

[9.例举身边的大数据。 114](#_Toc3300077)

[10.简述大数据的数据管理方式。 114](#_Toc3300078)

[11.什么是大数据? 115](#_Toc3300079)

[12.海量日志数据，提取出某日访问百度次数最多的那个IP。 115](#_Toc3300080)

[13.搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。 116](#_Toc3300081)

[14.有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。 117](#_Toc3300082)

[15.有10个文件，每个文件1G，每个文件的每一行存放的都是用户的query，每个文件的query都可能重复。要求你按照query的频度排序。 117](#_Toc3300083)

[16.JVM&垃圾回收机制 118](#_Toc3300084)

[17.在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。 119](#_Toc3300085)

[18.腾讯面试题：给40亿个不重复的unsignedint的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中? 119](#_Toc3300086)

[19.怎么在海量数据中找出重复次数最多的一个? 120](#_Toc3300087)

[20.上千万或上亿数据(有重复)，统计其中出现次数最多的钱N个数据。 120](#_Toc3300088)

[21.一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前10个词，请给出思想，给出时间复杂度分析。 120](#_Toc3300089)

[22.WARNTaskSchedulerImpl:Initialjobhasnotacceptedanyresources;checkyourclusteruitoensurethatworkersareregisteredandhavesufficientmemory 121](#_Toc3300090)

[23.Applicationisn’tusingalloftheCores:HowtosettheCoresusedbyaSparkApp 121](#_Toc3300091)

[24.SparkExecutorOOM:HowtosetMemoryParametersonSpark 121](#_Toc3300092)

[25.SharkServer/LongRunningApplicationMetadataCleanup 122](#_Toc3300093)

[26.ClassNotFound:ClasspathIssues 122](#_Toc3300094)

[23.使用mr，spark,sparksql编写wordcount程序 123](#_Toc3300095)

[24.如何为一个hadoop任务设置mappers的数量 123](#_Toc3300096)

[25.有可能使hadoop任务输出到多个目录中么?如果可以，怎么做? 123](#_Toc3300097)

[26.如何为一个hadoop任务设置要创建的reducer的数量 124](#_Toc3300098)

[27.SparkStreaming和Storm有何区别？ 124](#_Toc3300099)

[28.如果公司叫你写hadoop平台设计方案，你会如何规划Hadoop生产集群？ 124](#_Toc3300100)

[29.hadoop集群监控，你会关注哪些监控点？ 124](#_Toc3300101)

[30.实际生产中，传统关系型数据库如何迁移到hadoop平台，迁移过程中，你遇到了哪些问题？ 124](#_Toc3300102)

[Spark大数据针对性问题 124](#_Toc3300103)

[1、海量日志数据，提取出某日访问百度次数最多的那个IP。 125](#_Toc3300104)

[2、搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。假设目前有一千万个记录(这些查询串的重复度比较高，虽然总数是1千万，但如果除去重复后，不超过3百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门)，请你统计最热门的10个查询串，要求使用的内存不能超过1G。 125](#_Toc3300105)

[3、有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。 125](#_Toc3300106)

[4、有10个文件，每个文件1G，每个文件的每一行存放的都是用户的query，每个文件的query都可能重复。要求你按照query的频度排序。 126](#_Toc3300107)

[5、给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url? 126](#_Toc3300108)

[6、在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。 127](#_Toc3300109)

[7、腾讯面试题：给40亿个不重复的unsignedint的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中? 127](#_Toc3300110)

[8、怎么在海量数据中找出重复次数最多的一个? 128](#_Toc3300111)

[9、上千万或上亿数据(有重复)，统计其中出现次数最多的前N个数据。 128](#_Toc3300112)

[10、一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前10个词，请给出思想，给出时间复杂度分析。 128](#_Toc3300113)

[某公司笔试题 128](#_Toc3300114)

[1、如何为一个hadoop任务设置mappers的数量 129](#_Toc3300115)

[2、有可能使hadoop任务输出到多个目录中么？如果可以，怎么做？ 129](#_Toc3300116)

[3、如何为一个hadoop任务设置要创建的reducer的数量 129](#_Toc3300117)

[4、在hadoop中定义的主要公用InputFormats中，哪一个是默认值 130](#_Toc3300118)

[5、两个类TextInputFormat和KeyValueTextInputFormat的区别？ 130](#_Toc3300119)

[6、在一个运行的hadoop任务中，什么是InputSpilt? 131](#_Toc3300120)

[7、Hadoop框架中，文件拆分是怎么被调用的？ 131](#_Toc3300121)

[8、分别举例什么情况下使用combiner,什么情况下不会使用？ 131](#_Toc3300122)

[9、Hadoop中job和Tasks之间的区别是什么？ 131](#_Toc3300123)

[10、Hadoop中通过拆分任务到多个节点运行来实现并行计算，但是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种情况？ 132](#_Toc3300124)

[11、流API中的什么特性带来可以使mapreduce任务可以以不同语言(如perl\ruby\awk等)实现的灵活性？ 132](#_Toc3300125)

[12、参考下面的M/R系统的场景： 132](#_Toc3300126)

[13、Hadoop中的RecordReader的作用是什么？ 133](#_Toc3300127)

[14、Map阶段结束后，Hadoop框架会处理：Partitioning,shuffle和sort,在这个阶段都会发生了什么？ 133](#_Toc3300128)

[15、如果没有定义partitioner,那么数据在被送达reducer前是如何被分区的？ 133](#_Toc3300129)

[16、什么是Combiner? 134](#_Toc3300130)

[公司A： 135](#_Toc3300131)

[1.讲讲你做的过的项目。项目里有哪些难点重点注意点呢？ 136](#_Toc3300132)

[2.讲讲多线程吧，要是你，你怎么实现一个线程池呢？ 136](#_Toc3300133)

[3.讲一下Mapreduce或者hdfs的原理和机制。map读取数据分片。 136](#_Toc3300134)

[4.shuffle是什么？怎么调优？ 136](#_Toc3300135)

[5.项目用什么语言写？Scala？scala的特点？和Java的区别？ 136](#_Toc3300136)

[6.理论基础怎么样，比如数据结构，里面的快速排序，或者，树？讲一讲你了解的树的知识？ 136](#_Toc3300137)

[7.数学怎么样呢？ 136](#_Toc3300138)

[8.讲一下数据库，SQl，左外连接，原理，实现？ 136](#_Toc3300139)

[9.还了解过数据的什么知识？数据库引擎？ 136](#_Toc3300140)

[10.Hadoop的机架怎么配置的？ 136](#_Toc3300141)

[11.Hbase的设计有什么心得？ 136](#_Toc3300142)

[12.Hbase的操作是用的什么API还是什么工具？ 136](#_Toc3300143)

[13.对调度怎么理解.?用什么工具吗？ 136](#_Toc3300144)

[14.用kettle这种工具还是自己写程序？你们公司是怎么做的？ 136](#_Toc3300145)

[15.你们数据中心开发周期是多长？ 137](#_Toc3300146)

[16.你们hbase里面是存一些什么数据。 137](#_Toc3300147)

[二面。三个人。 137](#_Toc3300148)

[1.讲讲你做的项目。 138](#_Toc3300149)

[2.平时对多线程这方面是怎么处理呢？异步是怎么思考呢？遇到的一些锁啊，是怎么做的呢？比如两个人同时操作一样东西。怎么做的呢？一些并发操作设计到一些变量怎么做的呢？ 138](#_Toc3300150)

[3.你们用的最多是http协议吧？有没有特殊的头呢？讲讲你对tcp/ip的理解？ 138](#_Toc3300151)

[4.有没有用过Zookeeper呢？Zookeeper的适用场景是什么？HA状态维护分布式锁全局配置文件管理操作Zookeeper是用的什么？ 138](#_Toc3300152)

[Spark方面： 138](#_Toc3300153)

[5.spark开发分两个方面？哪两个方面呢？ 138](#_Toc3300154)

[6.比如一个读取hdfs上的文件，然后count有多少行的操作，你可以说说过程吗。那这个count是在内存中，还是磁盘中计算的呢？磁盘中。 138](#_Toc3300155)

[7.spark和Mapreduce快？为什么快呢？快在哪里呢？1.内存迭代。2.RDD设计。3,算子的设计。 138](#_Toc3300156)

[8.sparksql又为什么比hive快呢？ 138](#_Toc3300157)

[10.RDD的数据结构是怎么样的？Partition数组。dependence 139](#_Toc3300158)

[11.hadoop的生态呢。说说你的认识。hdfs底层存储hbase数据库hive数据仓库Zookeeper分布式锁spark大数据分析 139](#_Toc3300159)

[公司B： 139](#_Toc3300160)

[1.Spark工作的一个流程。 140](#_Toc3300161)

[2.Hbase的PUT的一个过程。 140](#_Toc3300162)

[3.RDD算子里操作一个外部map比如往里面put数据。然后算子外再遍历map。有什么问题吗。 140](#_Toc3300163)

[4.shuffle的过程。调优。 140](#_Toc3300164)

[5.5个partition里面分布有12345678910.用算子求最大值或者和。不能用广播变量和累加器。或者sortbykey. 140](#_Toc3300165)

[6.大表和小表join. 140](#_Toc3300166)

[7.知道spark怎么读hbase吗？sparkonhbase.。华为的。 140](#_Toc3300167)

[8.做过hbase的二级索引吗？ 140](#_Toc3300168)

[9.sortshuffle的优点？ 140](#_Toc3300169)

[10.stage怎么划分的？宽依赖窄依赖是什么？ 140](#_Toc3300170)

[公司W： 140](#_Toc3300171)

[1.讲讲你做过的项目(一个整体思路) 141](#_Toc3300172)

[2.问问大概情况。公司里集群规模。hbase数据量。数据规模。 141](#_Toc3300173)

[3.然后挑选数据工厂开始详细问。问hbase.。加闲聊。 141](#_Toc3300174)

[4.问二次排序是什么。topn是什么。二次排序要继承什么接口？ 141](#_Toc3300175)

[5.计算的数据怎么来的。 141](#_Toc3300176)

[6.kakfadirect是什么，。为什么要用这个，有什么优点？。和其他的有什么区别。 141](#_Toc3300177)

[7.问了shuffle过程。 141](#_Toc3300178)

[8.怎么调优的，jvm怎么调优的？ 141](#_Toc3300179)

[9.jvm结构？堆里面几个区？ 141](#_Toc3300180)

[10.数据清洗怎么做的？ 141](#_Toc3300181)

[11.怎么用spark做数据清洗 141](#_Toc3300182)

[12.跟我聊了spark的应用，商场里广告投放，以及黄牛检测 141](#_Toc3300183)

[13.spark读取数据，是几个Partition呢？hdfs几个block就有几个Partition？ 141](#_Toc3300184)

[14.sparkonyarn的两种模式?client模式？和cluster模式？ 142](#_Toc3300185)

[15.jdbc？mysql的驱动包名字叫什么？ 142](#_Toc3300186)

[16.region多大会分区？ 142](#_Toc3300187)

[公司Q 142](#_Toc3300188)

[公司M 142](#_Toc3300189)

[1.画图，画Spark的工作模式，部署分布架构图 143](#_Toc3300190)

[2.画图，画图讲解spark工作流程。以及在集群上和各个角色的对应关系。 143](#_Toc3300191)

[3.java自带有哪几种线程池。 143](#_Toc3300192)

[4.数据是怎么收集的。kafka收集数据的原理？ 143](#_Toc3300193)

[5.画图，讲讲shuffle的过程。那你怎么在编程的时候注意避免这些性能问题。 143](#_Toc3300194)

[6.讲讲列式存储的parquet文件底层格式。 143](#_Toc3300195)

[7.dataset和dataframe？ 143](#_Toc3300196)

[8.通过什么方式学习spark的？ 143](#_Toc3300197)

[9.有哪些数据倾斜，怎么解决？ 143](#_Toc3300198)

[10.宽依赖窄依赖？ 143](#_Toc3300199)

[11.yarn的原理？ 143](#_Toc3300200)

[12.BlockManager怎么管理硬盘和内存的。 143](#_Toc3300201)

[13.哪些算子操作涉及到shuffle 143](#_Toc3300202)

[14.看过源码？你熟悉哪几个部分的源码？ 143](#_Toc3300203)

[15.集群上nodemanager和ResourceManager的数量关系？ 144](#_Toc3300204)

[16.spark怎么整合hive？大概这样。sparkonhive。hive还是hive执行引擎是spark。 144](#_Toc3300205)

[其他人的： 144](#_Toc3300206)

[1.Spark如何处理结构化数据，Spark如何处理非结构话数据？ 144](#_Toc3300207)

[2.Spark性能优化主要有哪些手段？ 144](#_Toc3300208)

[3.简要描述Spark分布式集群搭建的步骤 144](#_Toc3300209)

[4.对于Spark你觉得他对于现有大数据的现状的优势和劣势在哪里？ 144](#_Toc3300210)

[5.对于算法是否进行过自主的研究设计？ 144](#_Toc3300211)

[6.简要描述你了解的一些数据挖掘算法与内容基本我有印象的就这几个问题，聊了2个多小时，脑子都差点被问干了 144](#_Toc3300212)

[2.有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M，要求返回频数最高的100个词。 145](#_Toc3300213)

[3.现有海量日志数据保存在一个超级大的文件中，该文件无法直接读入内存，要求从中提取某天出访问百度次数最多的那个IP。 146](#_Toc3300214)

[4.LVS和HAProxy相比，它的缺点是什么? 147](#_Toc3300215)

[5.Sqoop用起来感觉怎样? 149](#_Toc3300216)

[6.ZooKeeper的角色以及相应的Zookepper工作原理? 149](#_Toc3300217)

[7.HBase的Insert与Update的区别? 150](#_Toc3300218)

[8.请简述大数据的结果展现方式。 150](#_Toc3300219)

[9.例举身边的大数据。 150](#_Toc3300220)

[10.简述大数据的数据管理方式。 151](#_Toc3300221)

[11.什么是大数据? 151](#_Toc3300222)

[12.海量日志数据，提取出某日访问百度次数最多的那个IP。 151](#_Toc3300223)

[13.搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。 151](#_Toc3300224)

[16.给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url? 153](#_Toc3300225)

[17.在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。 153](#_Toc3300226)

[1、将现有逻辑在Spark上面实现。 155](#_Toc3300227)

[2、数据倾斜怎么处理？ 155](#_Toc3300228)

[3、各完成一个awk和sed的例子，最简单的应用即可，并说明。 156](#_Toc3300229)

[4、简要描述你知道的数据挖掘算法和使用场景 156](#_Toc3300230)

[5、列举你知道的常用的Hadoop管理和监控的命令、比如hdfsdfs-mkdir/usr 157](#_Toc3300231)

[6、评述hadoop运行原理 157](#_Toc3300232)

[7、讲述mapreduce的原理 157](#_Toc3300233)

[8、JVM&垃圾回收机制 158](#_Toc3300234)

[9、找出公共好友： 158](#_Toc3300235)

[10、combiner作用 158](#_Toc3300236)

[11、在mr环节中，那些环节需要优化，如何优化，请详细说明。 158](#_Toc3300237)

[十个海量数据处理方法大总结 162](#_Toc3300238)

[一、Bloomfilter 162](#_Toc3300239)

[二、Hashing 163](#_Toc3300240)

[三、bit-map 164](#_Toc3300241)

[四、堆 164](#_Toc3300242)

[五、双层桶划分—-其实本质上就是【分而治之】的思想，重在“分”的技巧上! 164](#_Toc3300243)

[六、数据库索引 165](#_Toc3300244)

[七、倒排索引(Invertedindex) 165](#_Toc3300245)

[八、外排序 166](#_Toc3300246)

[九、trie树 166](#_Toc3300247)

[十、分布式处理mapreduce 166](#_Toc3300248)

[Spark面试经典系列之数据倾斜： 168](#_Toc3300249)

[数据倾斜之痛 168](#_Toc3300250)

[解决原理和方法总论 168](#_Toc3300251)

[Map端Reduce及问题思考 168](#_Toc3300252)

[采样分而治之解决方案 169](#_Toc3300253)

[对于两个RDD数据量都很大且倾斜的Key特别多如何解决？ 170](#_Toc3300254)

[并行度的深度使用 171](#_Toc3300255)

[解决方案的“银弹”是什么？ 172](#_Toc3300256)

[使用案例机器学习 173](#_Toc3300257)

[1.商品特征未读降维：SVD、PCA 173](#_Toc3300258)

[2.商品挂错页面检查：TF-IDF、SVM、Logistic--Regression 173](#_Toc3300259)

[3.相关推荐算法模型训练：LoginisticRegression、kmeans、SVM 173](#_Toc3300260)

[4.商品爆品预测：LoginisticRegression 173](#_Toc3300261)

[5.关联性分析：FPGrowth 173](#_Toc3300262)

[6.开发了基于Mllib的机器学习平台 173](#_Toc3300263)

[经验分享 173](#_Toc3300264)

[用户常见错误 173](#_Toc3300265)

[Spark平台权限 174](#_Toc3300266)

[升级遇到的问题 175](#_Toc3300267)

[TODO 175](#_Toc3300268)

[BigData面试题总结 176](#_Toc3300269)

[JAVA相关 176](#_Toc3300270)

[1-1）List与set的区别？ 176](#_Toc3300271)

[1-2）数据库的三大范式？ 176](#_Toc3300272)

[1-3）java的io类的图解 176](#_Toc3300273)

[1-4）对象与引用对象的区别 176](#_Toc3300274)

[1-5）谈谈你对反射机制的理解及其用途？ 176](#_Toc3300275)

[1-6）列出至少五种设计模式 176](#_Toc3300276)

[1-7）RPC原理？ 176](#_Toc3300277)

[1-8）ArrayList、Vector、LinkedList的区别及其优缺点？HashMap、HashTable的区别及优缺点？ 177](#_Toc3300278)

[1-9）使用StringBuffer而不是String 177](#_Toc3300279)

[1-10）集合的扩充 177](#_Toc3300280)

[1-11）java的拆包与封包的问题 178](#_Toc3300281)

[1-12）Java中Class.forName和ClassLoader.loadClass的区别 178](#_Toc3300282)

[1-13）hashMap与hashTable的区别 178](#_Toc3300283)

[1-14）怎样实现数组的反转 179](#_Toc3300284)

[1-15）请使用JAVA实现二分查找 179](#_Toc3300285)

[1-16）java中有两个线程怎样等待一个线程执行完毕 180](#_Toc3300286)

[1-17）hashmaphashtablecurrentHashMap的使用区别 180](#_Toc3300287)

[1-18）简单描述一下java的gc机制，常用的JAVA调优的方法，OOM如何产生的，如何处理OOM问题？？？ 180](#_Toc3300288)

[Linux相关 181](#_Toc3300289)

[2-1）关闭不必要的服务 181](#_Toc3300290)

[2-2）关闭IP6 181](#_Toc3300291)

[2-3）调整文件的最大的打开数 181](#_Toc3300292)

[2-4）修改linux内核参数 182](#_Toc3300293)

[2-5）关闭noatime 182](#_Toc3300294)

[2-6)请用shell命令把某一个文件下的所有的文件分发到其他的机器上 182](#_Toc3300295)

[2-7）echo1+1&&echo"1+1"会输出什么 182](#_Toc3300296)

[2-8）在当前的额目录下找出包含祖母a并且文件的额大小大于55K的文件 183](#_Toc3300297)

[2-9）linux用什么命令查看cpu,硬盘，内存的信息？ 183](#_Toc3300298)

[Hadoop相关 183](#_Toc3300299)

[3-1）简单概述hdfs原理，以及各个模块的职责 183](#_Toc3300300)

[3-2）mr的工作原理 184](#_Toc3300301)

[3-3）怎样判断文件时候存在 184](#_Toc3300302)

[3-4）fsimage和edit的区别？ 184](#_Toc3300303)

[3-5）hdfs中的block默认保存几份？ 185](#_Toc3300304)

[3-6）列举几个配置文件优化？ 185](#_Toc3300305)

[3-7)谈谈数据倾斜，如何发生的，并给出优化方案 185](#_Toc3300306)

[3-8）简单概括安装hadoop的步骤 185](#_Toc3300307)

[3-9）简单概述hadoop中的角色的分配以及功能 186](#_Toc3300308)

[3-10）怎样快速的杀死一个job 186](#_Toc3300309)

[3-11)新增一个节点时怎样快速的启动 186](#_Toc3300310)

[3-12)你认为用java,streaming,pipe方式开发map/reduce,各有什么优点 186](#_Toc3300311)

[3-13）简单概述hadoop的join的方法 187](#_Toc3300312)

[3-14）简单概述hadoop的combinet与partition的区别 187](#_Toc3300313)

[3-15)hdfs的数据压缩算法 187](#_Toc3300314)

[3-16）hadoop的调度 187](#_Toc3300315)

[3-17）reduce后输出的数据量有多大？ 187](#_Toc3300316)

[3-18)datanode在什么情况下不会备份？ 187](#_Toc3300317)

[3-19）combine出现在那个过程？ 187](#_Toc3300318)

[3-20)hdfs的体系结构？ 188](#_Toc3300319)

[3-21)hadoopflush的过程？ 188](#_Toc3300320)

[3-22)什么是队列 188](#_Toc3300321)

[3-23）三个datanode，当有一个datanode出现错误会怎样？ 188](#_Toc3300322)

[3-24）mapReduce的执行过程 188](#_Toc3300323)

[3-25）Cloudera提供哪几种安装CDH的方法 188](#_Toc3300324)

[3-26）选择题与判断题 188](#_Toc3300325)

[3-27）hadoop的combinet与partition效果图 189](#_Toc3300326)

[3-28）hadoop的机架感知（或者说是扩普） 189](#_Toc3300327)

[3-29）文件大小默认为64M，改为128M有啥影响？ 189](#_Toc3300328)

[3-30）datanode首次加入cluster的时候，如果log报告不兼容文件版本，那需要namenode执行格式化操作，这样处理的原因是？ 189](#_Toc3300329)

[3-31）什么hadoopstreaming？ 189](#_Toc3300330)

[3-32）MapReduce中排序发生在哪几个阶段？这些排序是否可以避免？为什么？ 190](#_Toc3300331)

[3-33）hadoop的shuffer的概念 190](#_Toc3300332)

[3-34）hadoop的优化 190](#_Toc3300333)

[3-35)3个datanode中有一个个datanode出现错误会怎样？ 192](#_Toc3300334)

[3-36）怎样决定mapreduce的中的map以及reduce的数量 192](#_Toc3300335)

[3-37）两个文件合并的问题 192](#_Toc3300336)

[3-38）怎样决定一个job的map和reduce的数量 193](#_Toc3300337)

[3-39）hadoop的sequencefile的格式，并说明下什么是JAVA的序列化，如何实现JAVA的序列化 193](#_Toc3300338)

[3-40）简单概述一下hadoop1与hadoop2的区别 193](#_Toc3300339)

[3-41）YARN的新特性 193](#_Toc3300340)

[3-42）hadoopjoin的原理 193](#_Toc3300341)

[3-43）hadoop的二次排序 193](#_Toc3300342)

[3-44）hadoop的mapreduce的排序发生在几个阶段？ 194](#_Toc3300343)

[3-45）请描述mapreduce中shuffer阶段的工作流程，如何优化shuffer阶段的？ 194](#_Toc3300344)

[3-46）mapreduce的combiner的作用是什么，什么时候不易使用？？ 194](#_Toc3300345)

[Zookeeper相关 194](#_Toc3300346)

[4-1）写出你对zookeeper的理解 194](#_Toc3300347)

[4-2）zookeeper的搭建过程 195](#_Toc3300348)

[Hive相关 195](#_Toc3300349)

[5-1）hive是怎样保存元数据的 195](#_Toc3300350)

[5-2）外部表与内部表的区别 195](#_Toc3300351)

[5-3）对于hive，你写过哪些UDF函数，作用是什么 195](#_Toc3300352)

[5-4)Hive的sortby和orderby的区别 195](#_Toc3300353)

[5-5）hive保存元数据的方式以及各有什么特点？ 196](#_Toc3300354)

[5-6）在开发中问什么建议使用外部表？ 196](#_Toc3300355)

[5-7）hivepartition分区 196](#_Toc3300356)

[5-8）insertinto和overridewrite区别？ 196](#_Toc3300357)

[Hbase相关 196](#_Toc3300358)

[6-1）Hbase的rowkey怎么创建比较好？列族怎么创建比较好？ 196](#_Toc3300359)

[6-2）Hbase的实现原理 197](#_Toc3300360)

[6-3)hbase过滤器实现原则 197](#_Toc3300361)

[6-4）描述HBase,zookeeper搭建过程 197](#_Toc3300362)

[6-5）hive如何调优？ 198](#_Toc3300363)

[6-6）hive的权限的设置 198](#_Toc3300364)

[6-7)hbase写数据的原理 198](#_Toc3300365)

[6-8）hbase宕机了如何处理？ 198](#_Toc3300366)

[6-9）Hbase中的metastore用来做什么的？ 199](#_Toc3300367)

[6-10)hbase客户端在客户端怎样优化？ 199](#_Toc3300368)

[6-11）hbase是怎样预分区的？ 199](#_Toc3300369)

[6-12）怎样将mysql的数据导入到hbase中？ 199](#_Toc3300370)

[6-13）谈谈HBase集群安装注意事项？ 199](#_Toc3300371)

[6-14)简述HBase的瓶颈 200](#_Toc3300372)

[6-15）Redis,传统数据库,hbase,hive每个之间的区别 200](#_Toc3300373)

[6-16）Hbase的特性,以及你怎么去设计rowkey和columnFamily,怎么去建一个table 200](#_Toc3300374)

[6-17）Hhase与hive的区别 200](#_Toc3300375)

[6-18）描述hbase的scan和get功能以及实现的异同 200](#_Toc3300376)

[6-19）HBasescansetBatch和setCaching的区别 201](#_Toc3300377)

[6-20）hbase中cell的结构 201](#_Toc3300378)

[6-21）hbase中region太多和region太大带来的冲突 201](#_Toc3300379)

[Flume相关 201](#_Toc3300380)

[7-1）flume不采集Nginx日志，通过Logger4j采集日志，优缺点是什么？ 201](#_Toc3300381)

[7-2）flume和kafka采集日志区别，采集日志时中间停了，怎么记录之前的日志。 201](#_Toc3300382)

[Kafka相关 201](#_Toc3300383)

[8-1）kafka中怎样储存数据，哟及结构的，data.....目录下有多少个分区，每个分区的存储格式是什么样的？ 202](#_Toc3300384)

[Spark相关 202](#_Toc3300385)

[9-1）mr和spark区别，怎么理解spark-rdd 202](#_Toc3300386)

[9-2）简单描述spark的wordCount的执行过程 202](#_Toc3300387)

[9-3）按照需求使用spark编写一下程序 203](#_Toc3300388)

[Sqoop相关 209](#_Toc3300389)

[Redis相关 209](#_Toc3300390)

[10-1）redis保存磁盘的时间 209](#_Toc3300391)

[环境配置 210](#_Toc3300392)

[1）你们的集群规模？ 210](#_Toc3300393)

[2)你在项目中遇到了哪些难题，是怎么解决的？ 211](#_Toc3300394)

[设计题 211](#_Toc3300395)

[10、Hadoop性能调优？ 216](#_Toc3300396)

# ****BigData面试题总结****

## ****JAVA相关****

### ****1-1）List与set的区别？****

老掉牙的问题了，还在这里老生常谈：List特点：元素有放入顺序，元素可重复，Set特点：元素无放入顺序，元素不可重复。

### ****1-2）数据库的三大范式？****

原子性、一致性、唯一性

### ****1-3）java的io类的图解****

### ****1-4）对象与引用对象的区别****

对象就是好没有初始化的对象，引用对象即使对这个对象进行了初始化，这个初始化可以使自己的直接new的也可以是直接其他的赋值的，那么背new或者背其他赋值的我们叫做是引用对象，最大的区别于

### ****1-5）谈谈你对反射机制的理解及其用途？****

反射有三种获取的方式，分别是：forName/getClass/直接使用class方式使用反射可以获取类的实例

### ****1-6）列出至少五种设计模式****

设计方式有工厂法，懒加载，观察者模式，静态工厂，迭代器模式，外观模式、、、、

### ****1-7）RPC原理？****

Rpc分为同步调用和一部调用，异步与同步的区别在于是否等待服务器端的返回值。Rpc的组件有RpcServer,RpcClick,RpcProxy,RpcConnection,RpcChannel,RpcProtocol,RpcInvoker等组件，

### ****1-8）ArrayList、Vector、LinkedList的区别及其优缺点？HashMap、HashTable的区别及优缺点？****

ArrayList和Vector是采用数组方式存储数据的,是根据索引来访问元素的，都可以

根据需要自动扩展内部数据长度，以便增加和插入元素，都允许直接序号索引元素，但

是插入数据要涉及到数组元素移动等内存操作，所以索引数据快插入数据慢，他们最大

的区别就是synchronized同步的使用。

LinkedList使用双向链表实现存储，按序号索引数据需要进行向前或向后遍历，但

是插入数据时只需要记录本项的前后项即可，所以插入数度较快！

如果只是查找特定位置的元素或只在集合的末端增加、移除元素，那么使用Vector

或ArrayList都可以。如果是对其它指定位置的插入、删除操作，最好选择LinkedList

HashMap、HashTable的区别及其优缺点：

HashTable中的方法是同步的HashMap的方法在缺省情况下是非同步的因此在多线程环境下需要做额外的同步机制。

HashTable不允许有null值key和value都不允许，而HashMap允许有null值key和value都允许因此HashMap使用containKey（）来判断是否存在某个键。

HashTable使用Enumeration，而HashMap使用iterator。

Hashtable是Dictionary的子类，HashMap是Map接口的一个实现类。

### ****1-9）使用StringBuffer而不是String****

当需要对字符串进行操作时，使用StringBuffer而不是String，String是read-only的，如果对它进行修改，会产生临时对象，而StringBuffer是可修改的，不会产生临时对象。

### ****1-10）集合的扩充****

ArrayListlist=newArrayList(90000);list扩充多少次？？

**public**ArrayList(){

**this**(10);

}

默认的扩充是10由此计算

### ****1-11）java的拆包与封包的问题****

System.**out**.println("5"+2);

52

### ****1-12）Java中Class.forName和ClassLoader.loadClass的区别****

Class.forName("xx.xx")等同于Class.forName("xx.xx",true,CALLClass.class.getClassLoader())，第二个参数(bool)表示装载类的时候是否初始化该类，即调用类的静态块的语句及初始化静态成员变量。

ClassLoaderloader=Thread.currentThread.getContextClassLoader();//也可以用(ClassLoader.getSystemClassLoader())

Classcls=loader.loadClass("xx.xx");//这句话没有执行初始化

forName可以控制是否初始化类，而loadClass加载时是没有初始化的。

### ****1-13）hashMap与hashTable的区别****

HashMapHashtable

父类AbstractMapDictiionary

是否同步否是

k，v可否null是否

Hashtable和HashMap采用的hash/rehash[算法](http://lib.csdn.net/base/datastructure)都大概一样，所以性能不会有很大的差异。

### ****1-14）怎样实现数组的反转****

ArrayListarrayList=newArrayList();

arrayList.add("A");

arrayList.add("B");

对数组进行反转

Collections.reverse(arrayList);

### ****1-15）请使用JAVA实现二分查找****

一般的面试者都是些向看看你的思路，所以一般答题时只需要把思路写出来即可。

具体的实现如下：

二分查找就是折半查找，要想折半就必须把原来的数据进行排序，才能方便的查找：

实现代码如下：

publicstaticintbinarySearch(int[]srcArray,intdes){

intlow=0;

inthigh=srcArray.length-1;

while(low<=high){

intmiddle=(low+high)/2;

if(des==srcArray[middle]){

returnmiddle;

}elseif(des<srcArray[middle]){

high=middle-1;

}else{

low=middle+1;

}

}

return-1;

}

### ****1-16）java中有两个线程怎样等待一个线程执行完毕****

可以使用join关键字

### ****1-17）hashmaphashtablecurrentHashMap的使用区别****

hashmaphashtable的醉的的区别在于hashtable是线程安全的，而hashmap不是线程安全的，currentHashMap也是线程安全的。

ConcurrentHashMap是使用了锁分段技术技术来保证线程安全的。所分段的技术是：讲数据分成一段一段的储存，给每一段的数据添加一把锁，当线程访问一个数据时，其他的数据可以被访问。

### ****1-18）简单描述一下java的gc机制，常用的JAVA调优的方法，OOM如何产生的，如何处理OOM问题？？？****

1、程序在运行时会产生很多的对象的信息，当这些对象的信息没有用时，则会被gc回收

2、调优的方式主要是调节年轻代与老年代的内存的大小

3、OOM是OutOfMemory的缩写(搞得跟多高大上似的)就是线程创建的多了，没有及时的回收过来所产生的，代码如下：

publicclassJavaVMStackOOM{

privatevoiddontStop(){

while(true){

}

}

publicvoidstackLeakByThread(){

while(true){

Threadthread=newThread(newRunnable(){

@Override

publicvoidrun(){

dontStop();

}

});

thread.start();

}

}

publicstaticvoidmain(String[]args){

JavaVMStackOOMoom=newJavaVMStackOOM();

oom.stackLeakByThread();

}

4、既然知道以上的现象，在写代码时应该注意，不要过多的创建线程的数目。

## ****Linux相关****

### ****2-1）关闭不必要的服务****

A、使用ntsysv命令查看开启与关闭的服务

B、停止打印服务

[root@hadoop1/]#/etc/init.d/cupsstop

[root@hadoop1/]#chkconfigcupsoff

### ****2-2）关闭IP6****

[root@hadoop1/]#vim/etc/modprobe.conf

在下面添加一下配置：

aliasnet-pf-10off

aliasipv6off

### ****2-3）调整文件的最大的打开数****

查看当前的文件的数量：[root@hadoop1/]#ulimit-a

修改配置：

[root@hadoop1/]#vi/etc/security/limits.conf在文件最后加上：

\*softnofile65535

\*hardnofile65535

\*softnproc65535

\*hardnproc65535

### ****2-4）修改linux内核参数****

[root@hadoop1/]#vi/etc/sysctl.conf

在文本的最后追加一下内容：

net.core.somaxconn=32768

表示物理内存使用到90%（100-10=90）的时候才使用swap交换区

### ****2-5）关闭noatime****

在最后追加一下内容

/dev/sda2/dataext3noatime,nodiratime00

### ****2-6)请用shell命令把某一个文件下的所有的文件分发到其他的机器上****

Scp-r/user/localhadoop2:/user/local

### ****2-7）echo1+1&&echo"1+1"会输出什么****

[root@hadoop1~]#echo1+1&&echo"1+1"

1+1

1+1

[root@hadoop1~]#echo1+1&&echo"1+1"&&echo"1+"1

1+1

1+1

1+1

### ****2-8）在当前的额目录下找出包含祖母a并且文件的额大小大于55K的文件****

[root@hadoop1test]#find.|grep-ri"a"

a.text:a

后半句没有写出来，有时间在搞

### ****2-9）linux用什么命令查看cpu,硬盘，内存的信息？****

Top命令

## ****Hadoop相关****

### ****3-1）简单概述hdfs原理，以及各个模块的职责****

1、客户端向nameNode发送要上传文件的请求

2、nameNode返回给用户是否能上传数据的状态

3、加入用户端需要上传一个1024M的文件，客户端会通过Rpc请求NameNode，并返回需要上传给那些DataNode(分配机器的距离以及空间的大小等),namonode会选择就近原则分配机器。

4、客户端请求建立block传输管道chnnel上传数据

5、在上传是datanode会与其他的机器建立连接并把数据块传送到其他的机器上

6、dataNode向namenode汇报自己的储存情况以及自己的信息

7、档第一个快上传完后再去执行其他的复制的传送

### ****3-2）mr的工作原理****

1、当执行mr程序是，会执行一个Job

2、客户端的jobClick会请求namenode的jobTracker要执行任务

3、jobClick会去HDFS端复制作业的资源文件

4、客户端的jobClick会向namenode提交作业,让namenode做准备

5、Namenode的jobTracker会去初始化创建的对象

6、Namenode会获取hdfs的划分的分区

7、Namenode去检查TaskTracker的心跳信息，查看存活的机器

8、当执行的datenode执行任务时Datenode会去HDFS获取作业的资源的文件

9、TaskTracker会去执行代码，并登陆JVM的执行渠道

10、JVM或执行MapTask或者ReduceTask

11、执行终结

### ****3-3）怎样判断文件时候存在****

这是[Linux](http://lib.csdn.net/base/linux)上的知识，只需要在IF[-f]括号中加上-f参数即可判断文件是否存在

### ****3-4）fsimage和edit的区别？****

大家都知道namenode与secondarynamenode的关系，当他们要进行数据同步时叫做checkpoint时就用到了fsimage与edit，fsimage是保存最新的元数据的信息，当fsimage数据到一定的大小事会去生成一个新的文件来保存元数据的信息，这个新的文件就是edit，edit会回滚最新的数据。

### ****3-5）hdfs中的block默认保存几份？****

不管是hadoop1.x还是hadoop2.x都是默认的保存三份，可以通过参数dfs.replication就行修改，副本的数目要根据机器的个数来确定。

### ****3-6）列举几个配置文件优化？****

Core-site.xml文件的优化

fs.trash.interval

默认值：0

说明：这个是开启hdfs文件删除自动转移到垃圾箱的选项，值为垃圾箱文件清除时间。一般开启这个会比较好，以防错误删除重要文件。单位是分钟。

dfs.namenode.handler.count

默认值：10

说明：[Hadoop](http://lib.csdn.net/base/hadoop)系统里启动的任务线程数，这里改为40，同样可以尝试该值大小对效率的影响变化进行最合适的值的设定。

mapreduce.tasktracker.http.threads

默认值：40

说明：map和reduce是通过http进行数据传输的，这个是设置传输的并行线程数。

### ****3-7)谈谈数据倾斜，如何发生的，并给出优化方案****

数据的倾斜主要是两个的数据相差的数量不在一个级别上，在只想任务时就造成了数据的倾斜，可以通过分区的方法减少reduce数据倾斜性能的方法，例如;抽样和范围的分区、自定义分区、数据大小倾斜的自定义侧咯

### ****3-8）简单概括安装hadoop的步骤****

1.创建[hadoop](http://lib.csdn.net/base/hadoop)帐户。

2.setup.改IP。

3.安装[Java](http://lib.csdn.net/base/java)，并修改/etc/profile文件，配置java的环境变量。

4.修改Host文件域名。

5.安装SSH，配置无密钥通信。

6.解压hadoop。

7.配置conf文件下hadoop-env.sh、core-site.sh、mapre-site.sh、hdfs-site.sh。

8.配置hadoop的环境变量。

9.Hadoopnamenode-format

10.Start-all.sh

### ****3-9）简单概述hadoop中的角色的分配以及功能****

Namenode:负责管理元数据的信息

SecondName:做namenode冷备份，对于namenode的机器当掉后能快速切换到制定的Secondname上

DateNode:主要做储存数据的。

JobTracker:管理任务，并把任务分配到taskTasker

TaskTracker：执行任务的

### ****3-10）怎样快速的杀死一个job****

1、执行hadoopjob-list拿到job-id

2、Hadoopjobkillhadoop-id

### ****3-11)新增一个节点时怎样快速的启动****

Hadoop-daemon.shstartdatanode

### ****3-12)你认为用java,streaming,pipe方式开发map/reduce,各有什么优点****

开发mapReduce只用过java与[Hive](http://lib.csdn.net/base/hive),不过使用java开发mapreduce显得笨拙，效率也慢，基于java慢的原因于是[hive](http://lib.csdn.net/base/hive)，这样就方便了查询与设计

### ****3-13）简单概述hadoop的join的方法****

Hadoop常用的jion有reducesidejoin,mapsidejoin,SemiJoin不过reducesidejoin与mapsidejoin比较常用，不过都是比较耗时的。

### ****3-14）简单概述hadoop的combinet与partition的区别****

combine和partition都是函数，中间的步骤应该只有shuffle！combine分为map端和reduce端，作用是把同一个key的键值对合并在一起，可以自定义的,partition是分割map每个节点的结果，按照key分别映射给不同的reduce，也是可以自定义的。这里其实可以理解归类。

### ****3-15)hdfs的数据压缩算法****

Hadoop的压缩算法有很多，其中比较常用的就是gzip算法与bzip2算法，都可以可通过CompressionCodec来实现

### ****3-16）hadoop的调度****

Hadoop的调度有三种其中fifo的调度hadoop的默认的，这种方式是按照作业的优先级的高低与到达时间的先后执行的，还有公平调度器：名字见起意就是分配用户的公平获取共享集群呗!容量调度器:让程序都能货到执行的能力，在队列中获得资源。

### ****3-17）reduce后输出的数据量有多大？****

输出的数据量还不是取决于map端给他的数据量，没有数据reduce也没法运算啊!!

### ****3-18)datanode在什么情况下不会备份？****

Hadoop保存的三个副本如果不算备份的话，那就是在正常运行的情况下不会备份，也是就是在设置副本为1的时候不会备份，说白了就是单台机器呗！！还有datanode在强制关闭或者非正常断电不会备份。

### ****3-19）combine出现在那个过程？****

Hadoop的map过程，根据意思就知道结合的意思吗，剩下的你们就懂了。想想wordcound

### ****3-20)hdfs的体系结构？****

HDFS有namenode、secondraynamenode、datanode组成。

namenode负责管理datanode和记录元数据

secondraynamenode负责合并日志

datanode负责存储数据

### ****3-21)hadoopflush的过程？****

Flush就是把数据落到磁盘，把数据保存起来呗!

### ****3-22)什么是队列****

队列的实现是链表，消费的顺序是先进先出。

### ****3-23）三个datanode，当有一个datanode出现错误会怎样？****

第一不会给储存带来影响，因为有其他的副本保存着，不过建议尽快修复，第二会影响运算的效率，机器少了，reduce在保存数据时选择就少了，一个数据的块就大了所以就会慢。

### ****3-24）mapReduce的执行过程****

首先map端会Text接受到来自的数据，text可以把数据进行操作，最后通过context把key与value写入到下一步进行计算，一般的reduce接受的value是个集合可以运算，最后再通过context把数据持久化出来。

### ****3-25）Cloudera提供哪几种安装CDH的方法****

·Clouderamanager

·Tarball

·Yum

·Rpm

### ****3-26）选择题与判断题****

http://blog.csdn[.NET](http://lib.csdn.net/base/dotnet)/jiangheng0535/article/details/16800415

### ****3-27）hadoop的combinet与partition效果图****

### ****3-28）hadoop的机架感知（或者说是扩普）****

看图说话

数据块会优先储存在离namenode进的机器或者说成离namenode机架近的机器上，正好是验证了那句话不走网络就不走网络，不用磁盘就不用磁盘。

### ****3-29）文件大小默认为64M，改为128M有啥影响？****

这样减少了namenode的处理能力，数据的元数据保存在namenode上，如果在网络不好的情况下会增到datanode的储存速度。可以根据自己的网络来设置大小。

### ****3-30）datanode首次加入cluster的时候，如果log报告不兼容文件版本，那需要namenode执行格式化操作，这样处理的原因是？****

这样处理是不合理的，因为那么namenode格式化操作，是对文件系统进行格式

化，namenode格式化时清空dfs/name下空两个目录下的所有文件，之后，会在目

录dfs.name.dir下创建文件。

文本不兼容，有可能时namenode与datanode的数据里的namespaceID、

clusterID不一致，找到两个ID位置，修改为一样即可解决。

### ****3-31）什么hadoopstreaming？****

提示：指的是用其它语言处理

### ****3-32）MapReduce中排序发生在哪几个阶段？这些排序是否可以避免？为什么？****

一个MapReduce作业由Map阶段和Reduce阶段两部分组成，这两阶段会对数

据排序，从这个意义上说，MapReduce框架本质就是一个DistributedSort。在Map

阶段，在Map阶段，MapTask会在本地磁盘输出一个按照key排序（采用的是快速

排序）的文件（中间可能产生多个文件，但最终会合并成一个），在Reduce阶段，每

个ReduceTask会对收到的数据排序，这样，数据便按照Key分成了若干组，之后以

组为单位交给reduce（）处理。很多人的误解在Map阶段，如果不使用Combiner

便不会排序，这是错误的，不管你用不用Combiner，MapTask均会对产生的数据排

序（如果没有ReduceTask，则不会排序，实际上Map阶段的排序就是为了减轻Reduce

端排序负载）。由于这些排序是MapReduce自动完成的，用户无法控制，因此，在

hadoop1.x中无法避免，也不可以关闭，但hadoop2.x是可以关闭的。

### ****3-33）hadoop的shuffer的概念****

Shuffer是一个过程，实在map端到reduce在调reduce数据之前都叫shuffer,主要是分区与排序，也就是内部的缓存分分区以及分发（是reduce来拉数据的）和传输

### ****3-34）hadoop的优化****

1、优化的思路可以从配置文件和系统以及代码的设计思路来优化

2、配置文件的优化：调节适当的参数，在调参数时要进行[测试](http://lib.csdn.net/base/softwaretest)

3、代码的优化：combiner的个数尽量与reduce的个数相同，数据的类型保持一致，可以减少拆包与封包的进度

4、系统的优化：可以设置[linux](http://lib.csdn.net/base/linux)系统打开最大的文件数预计网络的带宽MTU的配置

5、为job添加一个Combiner，可以大大的减少shuffer阶段的maoTask拷贝过来给远程的reducetask的数据量，一般而言combiner与reduce相同。

6、在开发中尽量使用stringBuffer而不是string，string的模式是read-only的，如果对它进行修改，会产生临时的对象，二stringBuffer是可修改的，不会产生临时对象。

7、修改一下配置：

一下是修改mapred-site.xml文件

修改最大槽位数

槽位数是在各个tasktracker上的mapred-site.xml上设置的，默认都是2

<property>

<name>mapred.tasktracker.map.tasks.maximum</name>

task的最大数

<value>2</value>

</property>

<property>

<name>mapred.tasktracker.reduce.tasks.maximum</name>

ducetask的最大数

<value>2</value>

</property>

调整心跳间隔

集群规模小于300时，心跳间隔为300毫秒

mapreduce.jobtracker.heartbeat.interval.min心跳时间

北京市昌平区建材城西路金燕龙办公楼一层电话：400-618-9090

mapred.heartbeats.in.second集群每增加多少节点，时间增加下面的值

mapreduce.jobtracker.heartbeat.scaling.factor集群每增加上面的个数，心跳增多少

启动带外心跳

mapreduce.tasktracker.outofband.heartbeat默认是false

配置多块磁盘

mapreduce.local.dir

配置RPChander数目

mapred.job.tracker.handler.count默认是10，可以改成50，根据机器的能力

配置HTTP线程数目

tasktracker.http.threads默认是40，可以改成100根据机器的能力

选择合适的压缩方式

以snappy为例：

<property>

<name>mapred.compress.map.output</name>

<value>true</value>

</property>

<property>

<name>mapred.map.output.compression.codec</name>

<value>org.apache.hadoop.io.compress.SnappyCodec</value>

</property>

### ****3-35)3个datanode中有一个个datanode出现错误会怎样？****

这个datanode的数据会在其他的datanode上重新做备份。

### ****3-36）怎样决定mapreduce的中的map以及reduce的数量****

在mapreduce中map是有块的大小来决定的，reduce的数量可以按照用户的业务来配置。

### ****3-37）两个文件合并的问题****

[**给定a、b两个文件，各存放50亿个url，每个url各占用64字节，内存限制是4G，如何找出a、b文件共同的url？**](http://www.cnblogs.com/zemliu/archive/2012/08/16/2641036.html)

主要的思想是把文件分开进行计算，在对每个文件进行对比，得出相同的URL,因为以上说是含有相同的URL所以不用考虑数据倾斜的问题。详细的解题思路为：

可以估计每个文件的大小为5G\*64=300G，远大于4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

遍历文件a，对每个url求取hash(url)%1000，然后根据所得值将url分别存储到1000个小文件（设为a0,a1,...a999）当中。这样每个小文件的大小约为300M。遍历文件b，采取和a相同的方法将url分别存储到1000个小文件(b0,b1....b999)中。这样处理后，所有可能相同的url都在对应的小文件(a0vsb0,a1vsb1....a999vsb999)当中，不对应的小文件（比如a0vsb99）不可能有相同的url。然后我们只要求出1000对小文件中相同的url即可。

比如对于a0vsb0，我们可以遍历a0，将其中的url存储到hash\_map当中。然后遍历b0，如果url在hash\_map中，则说明此url在a和b中同时存在，保存到文件中即可。

如果分成的小文件不均匀，导致有些小文件太大（比如大于2G），可以考虑将这些太大的小文件再按类似的方法分成小小文件即可

### ****3-38）怎样决定一个job的map和reduce的数量****

map的数量通常是由hadoop集群的DFS块大小确定的，也就是输入文件的总块数，reduce端是复制map端的数据，相对于map端的任务，reduce节点资源是相对于比较缺少的，同时运行的速度会变慢，争取的任务的个数应该是0.95过着1.75。

### ****3-39）hadoop的sequencefile的格式，并说明下什么是JAVA的序列化，如何实现JAVA的序列化****

1、hadoop的序列化(sequencefile)是一二进制的形式来保存的

2、Java的序列化是讲对象的内容进行流化

3、实现序列化需要实现Serializable接口便可以了

### ****3-40）简单概述一下hadoop1与hadoop2的区别****

Hadoop2与hadoop1最大的区别在于HDFS的[架构](http://lib.csdn.net/base/architecture)与mapreduce的很大的区别，而且速度上有很大的提升，hadoop2最主要的两个变化是：namenode可以集群的部署了，hadoop2中的mapreduce中的jobTracker中的资源调度器与生命周期管理拆分成两个独立的组件，并命名为YARN

### ****3-41）YARN的新特性****

YARN是hadoop2.x之后才出的，主要是hadoop的HA(也就是集群)，磁盘的容错，资源调度器

### ****3-42）hadoopjoin的原理****

实现两个表的join首先在map端需要把表标示一下，把其中的一个表打标签，到reduce端再进行笛卡尔积的运算，就是reduce进行的实际的链接操作。

### ****3-43）hadoop的二次排序****

Hadoop默认的是HashPartitioner排序，当map端一个文件非常大另外一个文件非常小时就会产生资源的分配不均匀，既可以使用setPartitionerClass来设置分区，即形成了二次分区。

### ****3-44）hadoop的mapreduce的排序发生在几个阶段？****

发生在两个阶段即使map与reduce阶段

### ****3-45）请描述mapreduce中shuffer阶段的工作流程，如何优化shuffer阶段的？****

Mapreduce的shuffer是出在maptask到reducetask的这段过程中，首先会进入到copy过程，会通过http方式请求maptask所在的taskTracker获取maptask的输出的文件，因此当maptask结束，这些文件就会落到磁盘中，merge实在map端的动作，只是在map拷贝过来的数值，会放到内存缓冲区中，给shuffer使用，reduce阶段，不断的merge后最终会把文件放到磁盘中。

### ****3-46）mapreduce的combiner的作用是什么，什么时候不易使用？？****

Mapreduce中的Combiner就是为了避免map任务和reduce任务之间的数据传输而设置的，Hadoop允许用户针对maptask的输出指定一个合并函数。即为了减少传输到Reduce中的数据量。它主要是为了削减Mapper的输出从而减少网络带宽和Reducer之上的负载。

在数据量较少时不宜使用。

3-47）

## ****Zookeeper相关****

### ****4-1）写出你对zookeeper的理解****

随着[大数据](http://lib.csdn.net/base/hadoop)的快速发展，多机器的协调工作，避免主要机器单点故障的问题，于是就引入管理机器的一个软件，他就是zookeeper来协助机器正常的运行。

Zookeeper有两个角色分别是leader与follower，其中leader是主节点，其他的是副节点，在安装配置上一定要注意配置奇数个的机器上，便于zookeeper快速切换选举其他的机器。

在其他的软件执行任务时在zookeeper注册时会在zookeeper下生成相对应的目录，以便zookeeper去管理机器。

### ****4-2）zookeeper的搭建过程****

主要是配置文件zoo.cfg配置dataDir的路径一句dataLogDir的路径以及myid的配置以及server的配置，心跳端口与选举端口

## ****Hive相关****

### ****5-1）hive是怎样保存元数据的****

保存元数据的方式有：内存[数据库](http://lib.csdn.net/base/mysql)rerdy，本地[MySQL](http://lib.csdn.net/base/mysql)数据库，远程[mysql](http://lib.csdn.net/base/mysql)数据库，但是本地的mysql数据用的比较多，因为本地读写速度都比较快

### ****5-2）外部表与内部表的区别****

先来说下Hive中内部表与外部表的区别：

Hive创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。在删除表的时候，内部表的元数据和数据会被一起删除，而外部表只删除元数据，不删除数据。这样外部表相对来说更加安全些，数据组织也更加灵活，方便共享源数据。

### ****5-3）对于hive，你写过哪些UDF函数，作用是什么****

UDF:userdefinedfunction的缩写，编写hiveudf的两种方式extendsUDF重写evaluate第二种extendsGenericUDF重写initialize、getDisplayString、evaluate方法

### ****5-4)Hive的sortby和orderby的区别****

orderby会对输入做全局排序，因此只有一个reducer（多个reducer无法保证全局有序）只有一个reducer，会导致当输入规模较大时，需要较长的计算时间。

sortby不是全局排序，其在数据进入reducer前完成排序.

因此，如果用sortby进行排序，并且设置mapred.reduce.tasks>1，则sortby只保证每个reducer的输出有序，不保证全局有序。

### ****5-5）hive保存元数据的方式以及各有什么特点？****

1、Hive有内存数据库derby数据库，特点是保存数据小，不稳定

2、mysql数据库，储存方式可以自己设定，持久化好，一般企业开发都用mysql做支撑

### ****5-6）在开发中问什么建议使用外部表？****

1、外部表不会加载到hive中只会有一个引用加入到元数据中

2、在删除时不会删除表，只会删除元数据，所以不必担心数据的

### ****5-7）hivepartition分区****

分区表，动态分区

### ****5-8）insertinto和overridewrite区别？****

insertinto：将某一张表中的数据写到另一张表中

overridewrite：覆盖之前的内容。

## ****Hbase相关****

### ****6-1）Hbase的rowkey怎么创建比较好？列族怎么创建比较好？****

Rowkey是一个二进制码流，Rowkey的长度被很多开发者建议说设计在10~100个字节，不过建议是越短越好，不要超过16个字节。在查找时有索引会加快速度。

Rowkey散列原则、Rowkey唯一原则、针对事务数据Rowkey设计、针对统计数据的Rowkey设计、针对通用数据的Rowkey设计、支持多条件查询的RowKey设计。

总结设计列族：

1、一般不建议设计多个列族

2、数据块的缓存的设计

3、激进缓存设计

4、布隆过滤器的设计(可以提高随机读取的速度)

5、生产日期的设计

6、列族压缩

7、单元时间版本

### ****6-2）Hbase的实现原理****

[Hbase](http://lib.csdn.net/base/hbase)的实现原理是rpcProtocol

### ****6-3)hbase过滤器实现原则****

感觉这个问题有问题，过滤器多的是啦，说的是哪一个不知道!!!!

[hbase](http://lib.csdn.net/base/hbase)的过滤器有：RowFilter、PrefixFilter、KeyOnlyFilter、RandomRowFilter、InclusiveStopFilter、FirstKeyOnlyFilter、ColumnPrefixFilter、ValueFilter、ColumnCountGetFilter、SingleColumnValueFilter、SingleColumnValueExcludeFilter、WhileMatchFilter、FilterList

你看这么多过滤波器呢，谁知道你问的那个啊！！

比较常用的过滤器有：RowFilter一看就知道是行的过滤器，来过滤行的信息。PrefixFilter前缀的过滤器，就是把前缀作为参数来查找数据呗！剩下的不解释了看过滤器的直接意思就OK了很简单。

### ****6-4）描述HBase,zookeeper搭建过程****

Zookeeper的问题楼上爬爬有步骤，hbase主要的配置文件有hbase.env.sh主要配置的是JDK的路径以及是否使用外部的ZK，hbase-site.xml主要配置的是与HDFS的链接的路径以及zk的信息，修改regionservers的链接其他机器的配置。

### ****6-5）hive如何调优？****

在优化时要注意数据的问题，尽量减少数据倾斜的问题，减少job的数量，同事对小的文件进行成大的文件，如果优化的设计那就更好了，因为hive的运算就是mapReduce所以调节mapreduce的参数也会使性能提高，如调节task的数目。

### ****6-6）hive的权限的设置****

Hive的权限需要在hive-site.xml文件中设置才会起作用，配置默认的是false，需要把hive.security.authorization.enabled设置为true，并对不同的用户设置不同的权限，例如select,drop等的操作。

### ****6-7)hbase写数据的原理****

1.首先，Client通过访问ZK来请求目标数据的地址。

2.ZK中保存了-ROOT-表的地址，所以ZK通过访问-ROOT-表来请求数据地址。

3.同样，-ROOT-表中保存的是.META.的信息，通过访问.META.表来获取具体的RS。

4..META.表查询到具体RS信息后返回具体RS地址给Client。

5.Client端获取到目标地址后，然后直接向该地址发送数据请求。

### ****6-8）hbase宕机了如何处理？****

HBase的RegionServer宕机超过一定时间后，HMaster会将其所管理的region重新分布到其他活动的RegionServer上，由于数据和日志都持久在HDFS中，

该操作不会导致数据丢失。所以数据的一致性和安全性是有保障的。

但是重新分配的region需要根据日志恢复原RegionServer中的内存MemoryStore表，这会导致宕机的region在这段时间内无法对外提供服务。

而一旦重分布，宕机的节点重新启动后就相当于一个新的RegionServer加入集群，为了平衡，需要再次将某些region分布到该server。

因此，RegionServer的内存表memstore如何在节点间做到更高的可用，是HBase的一个较大的挑战。

### ****6-9）Hbase中的metastore用来做什么的？****

Hbase的metastore是用来保存数据的，其中保存数据的方式有有三种第一种于第二种是本地储存，第二种是远程储存这一种企业用的比较多

### ****6-10)hbase客户端在客户端怎样优化？****

Hbase使用JAVA来运算的，索引Java的优化也适用于hbase，在使用过滤器事记得开启bloomfilter可以是性能提高3-4倍，设置HBASE\_HEAPSIZE设置大一些

### ****6-11）hbase是怎样预分区的？****

如何去进行预分区，可以采用下面三步：

1.取样，先随机生成一定数量的rowkey,将取样数据按升序排序放到一个集合里

2.根据预分区的region个数，对整个集合平均分割，即是相关的splitKeys.

3.HBaseAdmin.createTable(HTableDescriptortableDescriptor,byte[][]splitkeys)可以指定预分区的splitKey，即是指定region间的rowkey临界值

### ****6-12）怎样将mysql的数据导入到hbase中？****

不能使用sqoop，速度太慢了，提示如下：

A、一种可以加快批量写入速度的方法是通过预先创建一些空的regions，这样当

数据写入HBase时，会按照region分区情况，在集群内做数据的负载均衡。

B、hbase里面有这样一个hfileoutputformat类，他的实现可以将数据转换成hfile

格式，通过new一个这个类，进行相关配置,这样会在hdfs下面产生一个文件，这个

时候利用hbase提供的jruby的loadtable.rb脚本就可以进行批量导入。

### ****6-13）谈谈HBase集群安装注意事项？****

需要注意的地方是ZooKeeper的配置。这与hbase-env.sh文件相关，文件中

HBASE\_MANAGES\_ZK环境变量用来设置是使用hbase默认自带的Zookeeper还

是使用独立的ZooKeeper。HBASE\_MANAGES\_ZK=false时使用独立的，为true时

使用默认自带的。

某个节点的HRegionServer启动失败，这是由于这3个节点的系统时间不一致相

差超过集群的检查时间30s。

### ****6-14)简述HBase的瓶颈****

Hbase主要的瓶颈就是传输问题，在操作时大部分的操作都是需要对磁盘操作的

### ****6-15）Redis,传统数据库,hbase,hive每个之间的区别****

[Redis](http://lib.csdn.net/base/redis)是基于内存的数据库，注重实用内存的计算，hbase是列式数据库，无法创建主键，地从是基于HDFS的，每一行可以保存很多的列，hive是数据的仓库，是为了减轻mapreduce而设计的，不是数据库是用来与红薯做交互的。

### ****6-16）Hbase的特性,以及你怎么去设计rowkey和columnFamily,怎么去建一个table****

因为hbase是列式数据库，列非表schema的一部分，所以只需要考虑rowkey和columnFamily即可，rowkey有为的相关性，最好数据库添加一个前缀，文件越小，查询速度越快，再设计列是有一个列簇，但是列簇不宜过多。

### ****6-17）Hhase与hive的区别****

ApacheHBase是一种Key/Value系统，它运行在HDFS之上。和Hive不一样，Hbase的能够在它的数据库上实时运行，而不是运行MapReduce任务。Hive被分区为表格，表格又被进一步分割为列簇。列簇必须使用schema定义，列簇将某一类型列集合起来（列不要求schema定义）。例如，“message”列簇可能包含：“to”,”from”“date”,“subject”,和”body”.每一个key/value对在Hbase中被定义为一个cell，每一个key由row-key，列簇、列和时间戳。在Hbase中，行是key/value映射的集合，这个映射通过row-key来唯一标识。Hbase利用Hadoop的基础设施，可以利用通用的设备进行水平的扩展。

### ****6-18）描述hbase的scan和get功能以及实现的异同****

HBase的查询实现只提供两种方式：1、按指定RowKey获取唯一一条记录，get方法（org.apache.hadoop.hbase.client.Get）2、按指定的条件获取一批记录，scan方法（org.apache.hadoop.hbase.client.Scan）实现条件查询功能使用的就是scan方式

### ****6-19）****[HBasescansetBatch和setCaching的区别](http://blog.csdn.net/caoli98033/article/details/44650497)

can可以通过setCaching与setBatch方法提高速度（以空间换时间），

setCaching设置的值为每次rpc的请求记录数，默认是1；cache大可以优化性能，但是太大了会花费很长的时间进行一次传输。

setBatch设置每次取的columnsize；有些row特别大，所以需要分开传给client，就是一次传一个row的几个column。

### ****6-20）hbase中cell的结构****

cell中的数据是没有类型的，全部是字节码形式存贮。

### ****6-21）hbase中region太多和region太大带来的冲突****

Hbase的region会自动split，当region太时，regio太大时分布会不均衡，同时对于大批量的代入数据建议如下：

1、还是必须让业务方对rowkey进行预分片，对业务数据rowkey进行md5或者其他的hash策略，让数据尽量随机分布而不是顺序写入。

2、随时观察region的大小，是否出现大region的情况。

## ****Flume相关****

### ****7-1）flume不采集Nginx日志，通过Logger4j采集日志，优缺点是什么？****

在nginx采集日志时无法获取session的信息，然而logger4j则可以获取session的信息，logger4j的方式比较稳定，不会宕机。缺点：不够灵活，logger4j的方式和项目结合过滤紧密，二flume的方式就比较灵活，便于插拔式比较好，不会影响项目的性能。

### ****7-2）flume和kafka采集日志区别，采集日志时中间停了，怎么记录之前的日志。****

Flume采集日志是通过流的方式直接将日志收集到存储层，而kafka试讲日志缓存在kafka

集群，待后期可以采集到存储层。Flume采集中间停了，可以采用文件的方式记录之前的日志，而kafka是采用offset(偏移量)的方式记录之前的日志。

## ****Kafka相关****

### ****8-1）kafka中怎样储存数据，哟及结构的，data.....目录下有多少个分区，每个分区的存储格式是什么样的？****

1、topic是按照“主题名-分区”存储的

2、分区个数由配置文件决定

3、每个分区下最重要的两个文件是0000000000.log和000000.index，0000000.log

以默认1G大小回滚。

## ****Spark相关****

### ****9-1）mr和spark区别，怎么理解spark-rdd****

Mr是文件方式的分布式计算框架，是将中间结果和最终结果记录在文件中，map和reduce的数据分发也是在文件中。

[Spark](http://lib.csdn.net/base/spark)是内存迭代式的计算框架，计算的中间结果可以缓存内存，也可以缓存硬盘，但是不是每一步计算都需要缓存的。

Spark-rdd是一个数据的分区记录集合，是利用内存来计算的，spark之所以快是因为有内存的模式

### ****9-2）简单描述spark的wordCount的执行过程****

[Scala](http://lib.csdn.net/base/scala)>sc.textFile("/usr/local/words.txt")

res0:org.apache.spark.rdd.RDD[String]=/usr/local/words.txtMapPartitionsRDD[1]attextFileat<console>:22

[scala](http://lib.csdn.net/base/scala)>sc.textFile("/usr/local/words.txt").flatMap(\_.split(""))

res2:org.apache.spark.rdd.RDD[String]=MapPartitionsRDD[4]atflatMapat<console>:22

scala>sc.textFile("/usr/local/words.txt").flatMap(\_.split("")).map((\_,1))

res3:org.apache.spark.rdd.RDD[(String,Int)]=MapPartitionsRDD[8]atmapat<console>:22

scala>sc.textFile("/usr/local/words.txt").flatMap(\_.split("")).map((\_,1)).reduceByKey(\_+\_)

res5:org.apache.spark.rdd.RDD[(String,Int)]=ShuffledRDD[17]atreduceByKeyat<console>:22

scala>sc.textFile("/usr/local/words.txt").flatMap(\_.split("")).map((\_,1)).reduceByKey(\_+\_).collect

res6:Array[(String,Int)]=Array((dageda,1),(xiaoli,1),(hellow,4),(xisdsd,1),(xiaozhang,1))

### ****9-3）按照需求使用spark编写一下程序****

A、当前文件a.text的格式为，请统计每个单词出现的个数

A,b,c,d

B,b,f,e

A,a,c,f

sc.textFile(“/user/local/a.text”).flatMap(\_.split(“,”)).map((\_,1)).ReduceByKey(\_+\_).collect()

或：

**package**cn.bigdata

**import**org.apache.spark.SparkConf

**import**org.apache.spark.SparkContext

**import**org.apache.spark.rdd.RDD

**object**Demo{

/\*

a,b,c,d

b,b,f,e

a,a,c,f

c,c,a,d

\*计算第四列每个元素出现的个数

\*/

**def**main(args:Array[String]):Unit={

**val**conf:SparkConf=**new**SparkConf().setAppName("demo").setMaster("local")

**val**sc:SparkContext=**new**SparkContext(conf)

**val**data:RDD[String]=sc.textFile("f://demo.txt")

//数据切分

**val**fourthData:RDD[(String,Int)]=data.map{x=>

**val**arr:Array[String]=x.split(",")

**val**fourth:String=arr(3)

(fourth,1)

}

**val**result:RDD[(String,Int)]=fourthData.reduceByKey(\_+\_);

println(result.collect().toBuffer)

}

}

B、HDFS中有两个文件a.text与b.text,文件的格式为(ip,username),如：a.text,b.text

a.text

127.0.0.1xiaozhang

127.0.0.1xiaoli

127.0.0.2wangwu

127.0.0.3lisi

B.text

127.0.0.4lixiaolu

127.0.0.5lisi

每个文件至少有1000万行，请用程序完成一下工作，

1）每个文件的个子的IP

2)出现在b.text而没有出现在a.text的IP

3)每个user出现的次数以及每个user对应的IP的个数

**代码如下：**

1）各个文件的ip数

packagecn.bigdata

importjava.util.concurrent.Executors

importorg.apache.hadoop.conf.Configuration

importorg.apache.hadoop.fs.FileSystem

importorg.apache.hadoop.fs.LocatedFileStatus

importorg.apache.hadoop.fs.Path

importorg.apache.hadoop.fs.RemoteIterator

importorg.apache.spark.SparkConf

importorg.apache.spark.SparkContext

importorg.apache.spark.rdd.RDD

importorg.apache.spark.rdd.RDD.rddToPairRDDFunctions

//各个文件的ip数

objectDemo2{

valcachedThreadPool=Executors.newCachedThreadPool()

defmain(args:Array[String]):Unit={

valconf:SparkConf=newSparkConf().setAppName("demo2").setMaster("local")

valsc:SparkContext=newSparkContext(conf)

valhdpConf:Configuration=newConfiguration

valfs:FileSystem=FileSystem.get(hdpConf)

vallistFiles:RemoteIterator[LocatedFileStatus]=fs.listFiles(newPath("f://txt/2/"),true)

while(listFiles.hasNext){

valfileStatus=listFiles.next

valpathName=fileStatus.getPath.getName

cachedThreadPool.execute(newRunnable(){

overridedefrun():Unit={

println("======================="+pathName)

analyseData(pathName,sc)

}

})

}

}

defanalyseData(pathName:String,sc:SparkContext):Unit={

valdata:RDD[String]=sc.textFile("f://txt/2/"+pathName)

valdataArr:RDD[Array[String]]=data.map(\_.split(""))

valipAndOne:RDD[(String,Int)]=dataArr.map(x=>{

valip=x(0)

(ip,1)

})

valcounts:RDD[(String,Int)]=ipAndOne.reduceByKey(\_+\_)

valsortedSort:RDD[(String,Int)]=counts.sortBy(\_.\_2,false)

sortedSort.saveAsTextFile("f://txt/3/"+pathName)

}

}

2）出现在b.txt而没有出现在a.txt的ip

packagecn.bigdata

importjava.util.concurrent.Executors

importorg.apache.spark.SparkConf

importorg.apache.spark.SparkContext

importorg.apache.spark.rdd.RDD

/\*

\*出现在b.txt而没有出现在a.txt的ip

\*/

objectDemo3{

valcachedThreadPool=Executors.newCachedThreadPool()

defmain(args:Array[String]):Unit={

valconf=newSparkConf().setAppName("Demo3").setMaster("local")

valsc=newSparkContext(conf)

valdata\_a=sc.textFile("f://txt/2/a.txt")

valdata\_b=sc.textFile("f://txt/2/b.txt")

valsplitArr\_a=data\_a.map(\_.split(""))

valip\_a:RDD[String]=splitArr\_a.map(x=>x(0))

valsplitArr\_b=data\_b.map(\_.split(""))

valip\_b:RDD[String]=splitArr\_b.map(x=>x(0))

valsubRdd:RDD[String]=ip\_b.subtract(ip\_a)

subRdd.saveAsTextFile("f://txt/4/")

}

}

3）

packagecn.bigdata

importorg.apache.spark.SparkConf

importorg.apache.spark.SparkContext

importorg.apache.spark.rdd.RDD

importscala.collection.mutable.Set

/\*

\*每个user出现的次数以及每个user对应的ip数

\*/

objectDemo4{

defmain(args:Array[String]):Unit={

valconf=newSparkConf().setAppName("Demo4").setMaster("local")

valsc=newSparkContext(conf)

valdata:RDD[String]=sc.textFile("f://txt/5/")

vallines=data.map(\_.split(""))

valuserIpOne=lines.map(x=>{

valip=x(0)

valuser=x(1)

(user,(ip,1))

})

valuserListIpCount:RDD[(String,(Set[String],Int))]=userIpOne.combineByKey(

x=>(Set(x.\_1),x.\_2),

(a:(Set[String],Int),b:(String,Int))=>{

(a.\_1+b.\_1,a.\_2+b.\_2)

},

(m:(Set[String],Int),n:(Set[String],Int))=>{

(m.\_1++n.\_1,m.\_2+n.\_2)

})

valresult:RDD[String]=userListIpCount.map(x=>{

x.\_1+":userCount:"+x.\_2.\_2+",ipCount:"+x.\_2.\_1.size

})

println(result.collect().toBuffer)

}

}

## ****Sqoop相关****

### 10-1）sqoop在导入到MySql数据库是怎样保证数据重复，如果重复了该怎么办？？

在导入时在语句的后面加上一下命令作为节点：

--incrementalappend\

--check-columnid\

--last-value1208

## ****Redis相关****

### ****10-1）redis保存磁盘的时间****

#Note:youcandisablesavingatallcommentingallthe"save"lines.

#

#Itisalsopossibletoremoveallthepreviouslyconfiguredsave

#pointsbyaddingasavedirectivewithasingleemptystringargument

#likeinthefollowingexample:

#

#save""

save9001

save30010

save6010000

## ****环境配置****

### ****1）你们的集群规模？****

这个得看个人在公司的规模，下面介绍一下我们公司的一些配置：

联想Systemx3750服务器，价格3.5万，内存容量32G，产品类型机架式，硬盘接口SSD,CPU频率2.6GH,CPU数量2颗，三级缓存15MB,cpu核心6核，cpu线程数12线程,最大内存支持1.5T,网络是千兆网卡,可插拔时硬盘接口12个卡槽,配置1T的容量

详细：http://detail.zol.com.cn/server/index1101243.shtml

名字软件运行管理

Hadoop1JDK,hadoopnamenode

Hadoop2JDK,hadoopnamenode

Hadoop3JDK,hadoopsecondaryNamenode

Hadoop4JDK,hadoopsecondaryNamenode

Hadoop5JDK,hadoopdatanode

Hadoop6JDK,hadoopdatanode

Hadoop7JDK,hadoopdatanode

Hadoop8JDK,hadoopdatanode

Hadoop9JDK,hadoopdatanode

Hadoop10JDK,zookeeper,tomcat,mvn,kafkaleader

Hadoop11JDK,zookeeper,tomcat,mvn,kafkafollower

Hadoop12JDK,zookeeper,tomcat,mvn,kafkafollower

Hadoop13JDK,hive,mysql,svn,logstarhhive,mysql,svn

Hadoop14JDK,hbase,mysql备份datanode

Hadoop15JDK,nginx,Log日志手机datanode

数据就是每天访问的Log日志不是很大，有的时候大有的时候小的可怜

### ****2)你在项目中遇到了哪些难题，是怎么解决的？****

1、在执行任务时发现副本的个数不对，经过一番的查找发现是超时的原因，修改了配置文件hdfs-site.xml：中修改了超时时间。

2、由于当时在分配各个目录空间大小时，没有很好的分配导致有的目录的空间浪费，于是整体商量后把储存的空间调大了一些。

## ****设计题****

1-1）采集nginx产生的日志，日志的格式为useriptimeurlhtmlId每天产生的文件的数据量上亿条，请设计方案把数据保存到HDFS上，并提供一下实时查询的功能（响应时间小于3s）

A、某个用户某天访问某个URL的次数

B、某个URL某天被访问的总次数

实时思路是：使用Logstash+Kafka+Spark-streaming+[redis](http://lib.csdn.net/base/redis)+报表展示平台

离线的思路是：Logstash+Kafka+Elasticsearch+Spark-streaming+关系型数据库

A、B、数据在进入到Spark-streaming中进行过滤，把符合要求的数据保存到Redis中

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

面试回来之后把其中比较重要的问题记了下来写了个总结：

（答案在后面）

1、简答说一下hadoop的map-reduce编程模型

2、hadoop的TextInputFormat作用是什么，如何自定义实现

3、hadoop和spark的都是并行计算，那么他们有什么相同和区别

4、为什么要用flume导入hdfs，hdfs的构架是怎样的

5、map-reduce程序运行的时候会有什么比较常见的问题

6、简单说一下hadoop和spark的shuffle过程

以下是自己的理解，如果有不对的地方希望各位大侠可以帮我指出来~：

1、简答说一下hadoop的map-reduce编程模型

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合

使用的是hadoop内置的数据类型，比如longwritable、text等

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

2、hadoop的TextInputFormat作用是什么，如何自定义实现

InputFormat会在map操作之前对数据进行两方面的预处理

1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次

2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法

在createRecordReader中可以自定义分隔符

3、hadoop和spark的都是并行计算，那么他们有什么相同和区别

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

4、为什么要用flume导入hdfs，hdfs的构架是怎样的

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死

5、map-reduce程序运行的时候会有什么比较常见的问题

比如说作业中大部分都完成了，但是总有几个reduce一直在运行

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作

6、简单说一下hadoop和spark的shuffle过程

hadoop：map端保存分片数据，通过网络收集到reduce端

spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor

减少shuffle可以提高性能

部分答案不是十分准确欢迎补充:-)

——-补充更新———

1、Hive中存放是什么？

表。

存的是和hdfs的映射关系，hive是逻辑上的数据仓库，实际操作的都是hdfs上的文件，HQL就是用sql语法来写的mr程序。

2、Hive与关系型数据库的关系？

没有关系，hive是数据仓库，不能和数据库一样进行实时的CURD操作。

是一次写入多次读取的操作，可以看成是ETL工具。

3、Flume工作机制是什么？

核心概念是agent，里面包括source、chanel和sink三个组件。

source运行在日志收集节点进行日志采集，之后临时存储在chanel中，sink负责将chanel中的数据发送到目的地。

只有成功发送之后chanel中的数据才会被删除。

首先书写flume配置文件，定义agent、source、chanel和sink然后将其组装，执行flume-ng命令。

4、Sqoop工作原理是什么？

hadoop生态圈上的数据传输工具。

可以将关系型数据库的数据导入非结构化的hdfs、hive或者bbase中，也可以将hdfs中的数据导出到关系型数据库或者文本文件中。

使用的是mr程序来执行任务，使用jdbc和关系型数据库进行交互。

import原理：通过指定的分隔符进行数据切分，将分片传入各个map中，在map任务中在每行数据进行写入处理没有reduce。

export原理：根据要操作的表名生成一个java类，并读取其元数据信息和分隔符对非结构化的数据进行匹配，多个map作业同时执行写入关系型数据库

5、Hbase行健列族的概念，物理模型，表的设计原则？

行健：是hbase表自带的，每个行健对应一条数据。

列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中的数据可以有很多，通过时间戳来区分。

物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。

rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。

列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

6、SparkStreaming和Storm有何区别？

一个实时毫秒一个准实时亚秒，不过storm的吞吐率比较低。

7、mllib支持的算法？

大体分为四大类，分类、聚类、回归、协同过滤。

8、简答说一下hadoop的map-reduce编程模型？

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合。

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出。

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则。

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出。

在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则。

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量。

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job。

9、Hadoop平台集群配置、环境变量设置？

zookeeper：修改zoo.cfg文件，配置dataDir，和各个zk节点的server地址端口，tickTime心跳时间默认是2000ms，其他超时的时间都是以这个为基础的整数倍，之后再dataDir对应目录下写入myid文件和zoo.cfg中的server相对应。

hadoop：修改

hadoop-env.sh配置java环境变量

core-site.xml配置zk地址，临时目录等

hdfs-site.xml配置nn信息，rpc和http通信地址，nn自动切换、zk连接超时时间等

yarn-site.xml配置resourcemanager地址

mapred-site.xml配置使用yarn

slaves配置节点信息

格式化nn和zk。

hbase：修改

hbase-env.sh配置java环境变量和是否使用自带的zk

hbase-site.xml配置hdfs上数据存放路径，zk地址和通讯超时时间、master节点

regionservers配置各个region节点

zoo.cfg拷贝到conf目录下

spark：

安装Scala

修改spark-env.sh配置环境变量和master和worker节点配置信息

环境变量的设置：直接在/etc/profile中配置安装的路径即可，或者在当前用户的宿主目录下，配置在.bashrc文件中，该文件不用source重新打开shell窗口即可，配置在.bash\_profile的话只对当前用户有效。

### 10、Hadoop性能调优？

调优可以通过系统配置、程序编写和作业调度算法来进行。

hdfs的block.size可以调到128/256（网络很好的情况下，默认为64）

调优的大头：mapred.map.tasks、mapred.reduce.tasks设置mr任务数（默认都是1）

mapred.tasktracker.map.tasks.maximum每台机器上的最大map任务数

mapred.tasktracker.reduce.tasks.maximum每台机器上的最大reduce任务数

mapred.reduce.slowstart.completed.maps配置reduce任务在map任务完成到百分之几的时候开始进入

这个几个参数要看实际节点的情况进行配置，reduce任务是在33%的时候完成copy，要在这之前完成map任务，（map可以提前完成）

mapred.compress.map.output,mapred.output.compress配置压缩项，消耗cpu提升网络和磁盘io

合理利用combiner

注意重用writable对象

11、Hadoop高并发？

首先肯定要保证集群的高可靠性，在高并发的情况下不会挂掉，支撑不住可以通过横向扩展。

datanode挂掉了使用hadoop脚本重新启动。

12、hadoop的TextInputFormat作用是什么，如何自定义实现？

InputFormat会在map操作之前对数据进行两方面的预处理。

1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次。

2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map。

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值。

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法。

在createRecordReader中可以自定义分隔符。

13、hadoop和spark的都是并行计算，那么他们有什么相同和区别？

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束。

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job。

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算。

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系。

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错。

14、为什么要用flume导入hdfs，hdfs的构架是怎样的？

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件。

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死。

15、map-reduce程序运行的时候会有什么比较常见的问题？

比如说作业中大部分都完成了，但是总有几个reduce一直在运行。

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜。

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作。

16、简单说一下hadoop和spark的shuffle过程？

hadoop：map端保存分片数据，通过网络收集到reduce端。

spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor。

减少shuffle可以提高性能。

17、RDD机制？

rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。

所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。

rdd执行过程中会形成dag图，然后形成lineage保证容错性等。

从物理的角度来看rdd存储的是block和node之间的映射。

18、spark有哪些组件？

（1）master：管理集群和节点，不参与计算。

（2）worker：计算节点，进程本身不参与计算，和master汇报。

（3）Driver：运行程序的main方法，创建sparkcontext对象。

（4）sparkcontext：控制整个application的生命周期，包括dagsheduler和taskscheduler等组件。

（5）client：用户提交程序的入口。

19、spark工作机制？

用户在client端提交作业后，会由Driver运行main方法并创建sparkcontext上下文。

执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入taskscheduler。

taskscheduler会将stage划分为taskset分发到各个节点的executor中执行。

20、spark的优化怎么做？

通过spark-env文件、程序中sparkconf和setproperty设置。

（1）计算量大，形成的lineage过大应该给已经缓存了的rdd添加checkpoint，以减少容错带来的开销。

（2）小分区合并，过小的分区造成过多的切换任务开销，使用repartition。

21、kafka工作原理？

producer向broker发送事件，consumer从broker消费事件。

事件由topic区分开，每个consumer都会属于一个group。

相同group中的consumer不能重复消费事件，而同一事件将会发送给每个不同group的consumer。

22、ALS算法原理？

答：对于user-product-rating数据，als会建立一个稀疏的评分矩阵，其目的就是通过一定的规则填满这个稀疏矩阵。

als会对稀疏矩阵进行分解，分为用户-特征值，产品-特征值，一个用户对一个产品的评分可以由这两个矩阵相乘得到。

通过固定一个未知的特征值，计算另外一个特征值，然后交替反复进行最小二乘法，直至差平方和最小，即可得想要的矩阵。

23、kmeans算法原理？

随机初始化中心点范围，计算各个类别的平均值得到新的中心点。

重新计算各个点到中心值的距离划分，再次计算平均值得到新的中心点，直至各个类别数据平均值无变化。

24、canopy算法原理？

根据两个阈值来划分数据，以随机的一个数据点作为canopy中心。

计算其他数据点到其的距离，划入t1、t2中，划入t2的从数据集中删除，划入t1的其他数据点继续计算，直至数据集中无数据。

25、朴素贝叶斯分类算法原理？

对于待分类的数据和分类项，根据待分类数据的各个特征属性，出现在各个分类项中的概率判断该数据是属于哪个类别的。

26、关联规则挖掘算法apriori原理？

一个频繁项集的子集也是频繁项集，针对数据得出每个产品的支持数列表，过滤支持数小于预设值的项，对剩下的项进行全排列，重新计算支持数，再次过滤，重复至全排列结束，可得到频繁项和对应的支持数。

作者：[@小黑](http://www.xiaohei.info/)

以下是自己的理解，如果有不对的地方希望各位大侠可以帮我指出来~：

1、简答说一下hadoop的map-reduce编程模型

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合

使用的是hadoop内置的数据类型，比如longwritable、text等

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

2、hadoop的TextInputFormat作用是什么，如何自定义实现

InputFormat会在map操作之前对数据进行两方面的预处理

1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次

2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法

在createRecordReader中可以自定义分隔符

3、hadoop和spark的都是并行计算，那么他们有什么相同和区别

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

4、为什么要用flume导入hdfs，hdfs的构架是怎样的

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死

5、map-reduce程序运行的时候会有什么比较常见的问题

比如说作业中大部分都完成了，但是总有几个reduce一直在运行

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作

6、简单说一下hadoop和spark的shuffle过程

hadoop：map端保存分片数据，通过网络收集到reduce端

spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor

减少shuffle可以提高性能

部分答案不是十分准确欢迎补充:-)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

单项选择题

1.下面哪个程序负责HDFS数据存储。

a)NameNode

b)Jobtracker

c)Datanode

d)secondaryNameNode

e)tasktracker

2.HDfS中的block默认保存几份？

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？

a)SecondaryNameNode

b)DataNode

c)TaskTracker

d)Jobtracker

4.Hadoop作者

a)MartinFowler

b)KentBeck

c)Dougcutting

5.HDFS默认BlockSize

a)32MB

b)64MB

c)128MB

6.下列哪项通常是集群的最主要瓶颈

a)CPU

b)网络

c)磁盘

d)内存

7.关于SecondaryNameNode哪项是正确的？

a)它是NameNode的热备

b)它对内存没有要求

c)它的目的是帮助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

多选题：

8.下列哪项可以作为集群的管理工具

a)Puppet

b)Pdsh

c)ClouderaManager

d)d)Zookeeper

9.配置机架感知的下面哪项正确

a)如果一个机架出问题，不会影响数据读写

b)写入数据的时候会写到不同机架的DataNode中

c)MapReduce会根据机架获取离自己比较近的网络数据

10.Client端上传文件的时候下列哪项正确

a)数据经过NameNode传递给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

11.下列哪个是Hadoop运行的模式

a)单机版

b)伪分布式

c)分布式

12.Cloudera提供哪几种安装CDH的方法

a)Clouderamanager

b)Tarball

c)Yumd)Rpm

判断题：

13.Ganglia不仅可以进行监控，也可以进行告警。（）

14.BlockSize是不可以修改的。（）

15.Nagios不可以监控Hadoop集群，因为它不提供Hadoop支持。（）

16.如果NameNode意外终止，SecondaryNameNode会接替它使集群继续工作。（）

17.ClouderaCDH是需要付费使用的。（）

18.Hadoop是Java开发的，所以MapReduce只支持Java语言编写。（）

19.Hadoop支持数据的随机读写。（）

20.NameNode负责管理metadata，client端每次读写请求，它都会从磁盘中读取或则会写入metadata信息并反馈client端。（）

21.NameNode本地磁盘保存了Block的位置信息。（）

22.DataNode通过长连接与NameNode保持通信。（）

23.Hadoop自身具有严格的权限管理和安全措施保障集群正常运行。（）

24.Slave节点要存储数据，所以它的磁盘越大越好。（）

25.hadoopdfsadmin–report命令用于检测HDFS损坏块。（）

26.Hadoop默认调度器策略为FIFO（）

27.集群内每个节点都应该配RAID，这样避免单磁盘损坏，影响整个节点运行。（）

28.因为HDFS有多个副本，所以NameNode是不存在单点问题的。（）

29.每个map槽就是一个线程。（）

30.Mapreduce的inputsplit就是一个block。（）

31.NameNode的WebUI端口是50030，它通过jetty启动的Web服务。（）

32.Hadoop环境变量中的HADOOP\_HEAPSIZE用于设置所有Hadoop守护线程的内存。它默认是200GB。（）

33.DataNode首次加入cluster的时候，如果log中报告不兼容文件版本，那需要NameNode执行“Hadoopnamenode-format”操作格式化磁盘。（）

别走开，答案在后面哦！

1.下面哪个程序负责HDFS数据存储。答案Cdatanode

a)NameNode

b)Jobtracker

c)Datanode

d)secondaryNameNode

e)tasktracker

2.HDfS中的block默认保存几份？答案A默认3分

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？答案D

a)SecondaryNameNode

b)DataNode

c)TaskTracker

d)Jobtracker

此题分析：

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secondaryNameNode（运行在单独的物理机器上）和NameNode运行在不同的机器上。

JobTracker和TaskTracker

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：obclient，JobTracker与TaskTracker。

1、JobClient会在用户端通过JobClient类将应用已经配置参数打包成jar文件存储到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。

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.Hadoop作者答案CDougcutting

a)MartinFowler

b)KentBeck

c)Dougcutting

5.HDFS默认BlockSize答案：B

a)32MB

b)64MB

c)128MB

(因为版本更换较快，这里答案只供参考)

6.下列哪项通常是集群的最主要瓶颈：答案：C磁盘

a)CPU

b)网络

c)磁盘IO

d)内存

该题解析：

首先集群的目的是为了节省成本，用廉价的pc机，取代小型机及大型机。小型机和大型机有什么特点？

1.cpu处理能力强

2.内存够大

所以集群的瓶颈不可能是a和d

3.网络是一种稀缺资源，但是并不是瓶颈。

4.由于大数据面临海量数据，读写数据都需要io，然后还要冗余数据，hadoop一般备3份数据，所以IO就会打折扣。

7.关于SecondaryNameNode哪项是正确的？答案C

a)它是NameNode的热备

b)它对内存没有要求

c)它的目的是帮助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。

多选题：

8.下列哪项可以作为集群的管理？答案：ABD

a)Puppet

b)Pdsh

c)ClouderaManager

d)Zookeeper

9.配置机架感知的下面哪项正确：答案ABC

a)如果一个机架出问题，不会影响数据读写

b)写入数据的时候会写到不同机架的DataNode中

c)MapReduce会根据机架获取离自己比较近的网络数据

10.Client端上传文件的时候下列哪项正确？答案B

a)数据经过NameNode传递给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

该题分析：

Client向NameNode发起文件写入的请求。

NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息。

Client将文件划分为多个Block，根据DataNode的地址信息，按顺序写入到每一个DataNode块中。

11.下列哪个是Hadoop运行的模式：答案ABC

a)单机版

b)伪分布式

c)分布式

12.Cloudera提供哪几种安装CDH的方法？答案：ABCD

a)Clouderamanager

b)Tarball

c)Yum

d)Rpm

判断题：

13.Ganglia不仅可以进行监控，也可以进行告警。（正确）

分析：此题的目的是考Ganglia的了解。严格意义上来讲是正确。ganglia作为一款最常用的Linux环境中的监控软件，它擅长的的是从节点中按照用户的需求以较低的代价采集数据。但是ganglia在预警以及发生事件后通知用户上并不擅长。最新的ganglia已经有了部分这方面的功能。但是更擅长做警告的还有Nagios。Nagios，就是一款精于预警、通知的软件。通过将Ganglia和Nagios组合起来，把Ganglia采集的数据作为Nagios的数据源，然后利用Nagios来发送预警通知，可以完美的实现一整套监控管理的系统。

14.BlockSize是不可以修改的。（错误）

分析：它是可以被修改的Hadoop的基础配置文件是hadoop-default.xml，默认建立一个Job的时候会建立Job的Config，Config首先读入hadoop-default.xml的配置，然后再读入hadoop-site.xml的配置（这个文件初始的时候配置为空），hadoop-site.xml中主要配置需要覆盖的hadoop-default.xml的系统级配置。

15.Nagios不可以监控Hadoop集群，因为它不提供Hadoop支持。（错误）

分析：Nagios是集群监控工具，而且是云计算三大利器之一

16.如果NameNode意外终止，SecondaryNameNode会接替它使集群继续工作。（错误）

分析：SecondaryNameNode是帮助恢复，而不是替代，如何恢复，可以查看.

17.ClouderaCDH是需要付费使用的。（错误）

分析：第一套付费产品是ClouderaEnterpris，ClouderaEnterprise在美国加州举行的Hadoop大会(HadoopSummit)上公开，以若干私有管理、监控、运作工具加强Hadoop的功能。收费采取合约订购方式，价格随用的Hadoop叢集大小变动。

18.Hadoop是Java开发的，所以MapReduce只支持Java语言编写。（错误）

分析：rhadoop是用R语言开发的，MapReduce是一个框架，可以理解是一种思想，可以使用其他语言开发。

19.Hadoop支持数据的随机读写。（错）

分析：lucene是支持随机读写的，而hdfs只支持随机读。但是HBase可以来补救。HBase提供随机读写，来解决Hadoop不能处理的问题。HBase自底层设计开始即聚焦于各种可伸缩性问题：表可以很“高”，有数十亿个数据行；也可以很“宽”，有数百万个列；水平分区并在上千个普通商用机节点上自动复制。表的模式是物理存储的直接反映，使系统有可能提高高效的数据结构的序列化、存储和检索。

20.NameNode负责管理metadata，client端每次读写请求，它都会从磁盘中读取或则会写入metadata信息并反馈client端。（错误）

此题分析：

NameNode不需要从磁盘读取metadata，所有数据都在内存中，硬盘上的只是序列化的结果，只有每次namenode启动的时候才会读取。

1）文件写入

Client向NameNode发起文件写入的请求。

NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息。

Client将文件划分为多个Block，根据DataNode的地址信息，按顺序写入到每一个DataNode块中。

2）文件读取

Client向NameNode发起文件读取的请求。

21.NameNode本地磁盘保存了Block的位置信息。（个人认为正确，欢迎提出其它意见）

分析：DataNode是文件存储的基本单元，它将Block存储在本地文件系统中，保存了Block的Meta-data，同时周期性地将所有存在的Block信息发送给NameNode。NameNode返回文件存储的DataNode的信息。

Client读取文件信息。

22.DataNode通过长连接与NameNode保持通信。（）

这个有分歧：具体正在找这方面的有利资料。下面提供资料可参考。

首先明确一下概念：

（1）.长连接

Client方与Server方先建立通讯连接，连接建立后不断开，然后再进行报文发送和接收。这种方式下由于通讯连接一直存在，此种方式常用于点对点通讯。

（2）.短连接

Client方与Server每进行一次报文收发交易时才进行通讯连接，交易完毕后立即断开连接。此种方式常用于一点对多点通讯，比如多个Client连接一个Server.

23.Hadoop自身具有严格的权限管理和安全措施保障集群正常运行。（错误）

分析：hadoop只能阻止好人犯错，但是不能阻止坏人干坏事

24.Slave节点要存储数据，所以它的磁盘越大越好。（错误）

分析：一旦Slave节点宕机，数据恢复是一个难题

25.hadoopdfsadmin–report命令用于检测HDFS损坏块。（错误）

26.Hadoop默认调度器策略为FIFO（正确）

27.集群内每个节点都应该配RAID，这样避免单磁盘损坏，影响整个节点运行。（错误）

分析：首先明白什么是RAID，可以参考百科磁盘阵列。这句话错误的地方在于太绝对，具体情况具体分析。题目不是重点，知识才是最重要的。因为hadoop本身就具有冗余能力，所以如果不是很严格不需要都配备RAID。具体参考第二题。

28.因为HDFS有多个副本，所以NameNode是不存在单点问题的。（错误）

29.每个map槽就是一个线程。（错误）

分析：首先我们知道什么是map槽,map槽->mapslotmapslot只是一个逻辑值(org.apache.hadoop.mapred.TaskTracker.TaskLauncher.numFreeSlots)，而不是对应着一个线程或者进程

30.Mapreduce的inputsplit就是一个block。（错误）

31.NameNode的WebUI端口是50030，它通过jetty启动的Web服务。（错误）

32.Hadoop环境变量中的HADOOP\_HEAPSIZE用于设置所有Hadoop守护线程的内存。它默认是200GB。（错误）

分析：hadoop为各个守护进程（namenode,secondarynamenode,jobtracker,datanode,tasktracker）统一分配的内存在hadoop-env.sh中设置，参数为HADOOP\_HEAPSIZE，默认为1000M。

33.DataNode首次加入cluster的时候，如果log中报告不兼容文件版本，那需要NameNode执行“Hadoopnamenode-format”操作格式化磁盘。（错误）

分析：

首先明白介绍，什么ClusterID

ClusterID。添加了一个新的标识符ClusterID用于标识集群中所有的节点。当格式化一个Namenode，需要提供这个标识符或者自动生成。这个ID可以被用来格式化加入集群的其他Namenode。

二次整理

有的同学问题的重点不是上面分析内容：内容如下：

这个报错是说明DataNode所装的Hadoop版本和其它节点不一致，应该检查DataNode的Hadoop版本

|  |
| --- |
|  |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

　　1、你会Java语言吗？熟悉到什么程度？

　　2、你最喜欢的编程语言是什么？为什么？

　　3、处理过的最大的数据量？你是如何处理他们的？处理的结果如何。

　　2、在处理大数据过程中，如何保证得到期望值？

　　3、如何让一个网络爬虫速度更快、抽取更好的信息以及更好总结数据从而得到一干净的数据库？

　　4、点击流数据应该是实时处理？为什么？哪部分应该实时处理？

　　6、如何把非结构化的数据转换成结构化的数据？这是否真的有必要做这样的转换？把数据存成平面文本文件是否比存成关系数据库更好？

　　7、如何判别mapreduce过程有好的负载均衡？什么是负载均衡？

　　8、Spark和Hive的区别，以及Spark和Hive的数据倾斜调优问题？

　　9、Hive和Hbase的区别？

　　10、MapReduce的思想，以及MapReduce调优问题？

　　11、你所了解的开源网站？

　　12、有两个集群，每个集群有3个节点，使用hive分析相同的数据，sql语句完全一样，一个集群的分析结果比另外一个慢的多，给出造成这种现象的可能原因？

　　13、Hbase的优化？

　　14、集群的版本，以及集群的瓶颈问题？

　　15、CRM项目，怎么跟Spark结合？

　　16、如何创建一个关键字分类？

　　17、海量日志数据，提取出某日访问百度次数最多的那个IP？

　　18、Hadoop和Spark处理数据时，出现内存溢出的处理方法？

　　19、有一个1G大小的一个文件，里面每一是一个词，词的大小不超过16字节，内存大小限制大小1M，返回频率最高的50个词。

　　20、你是如何处理缺少数据的？你是推荐使用什么样的处理技术，或者说你是用什么样的技术处理呢？

　　如果不会Java，面试第一题就答不会，这样难道不尴尬吗？正如加里·金(崇拜/崇拜)说：“这是一场革命，庞大的数据资源使得各个领域开始了量化进程，无论学术界、商界还是政府，所有领域都将开始这种进程。”大数据时代，没有Java基础，如何开启你的大数据时代？

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.简要描述如何安装配置一个apache开源版hadoop，描述即可，列出步骤更好

2.请列出正常工作的hadoop集群中hadoop都需要启动哪些进程，他们的作用分别是什么？

3.启动hadoop报如下错误，该如何解决？

errororg.apache.hadoop.hdfs.server.namenode.NameNode

org.apache.hadoop.hdfs.server.common.inconsistentFSStateExceptio

nDirectory/tmp/hadoop-root/dfs/nameisinaninconsistent

statestoragedirectiondoesnotexistorisnotaccessible?

4.请写出以下执行命令

1）杀死一个job?

2)删除hdfs上的/tmp/aaa目录

3加入一个新的存储节点和删除一个计算节点需要刷新集群状态命令？

5.请列出你所知道的hadoop调度器，并简要说明其工作方法？

6.请列出在你以前工作中所使用过的开发mapreduce的语言？

7.当前日志采样格式为

1. a,b,c,d
2. b,b,f,e
3. a,a,c,f

复制代码

请用你最熟悉的语言编写一个mapreduce，并计算第四列每个元素出现的个数

8.你认为用Java，Streaming,pipe方式开发mapreduce,各有哪些优缺点？

9.hive有哪些方式保存元数据，各有哪些特点？

10.请简述hadoop怎么样实现二级排序？

11.简述hadoop实现join的几种方法？

12.请用Java实现非递归二分查找？

13.请简述mapreduce中，combiner，partition作用？

14.某个目录下有两个文件a.txt和b.txt,文件格式为（ip，username）,

列如：

a.txt

127.0.0.1zhangsan

127.0.0.1wangxiaoer

127.0.0.2lisi

127.0.0.3wangwu

b.txt

127.0.0.4lixiaolu

127.0.0.1lisi

每个文件至少100万行，请使用Linux命令完成如下工作：

1）每个文件各自的ip数

2）出现在b.txt而没有出现在a.txt的ip

3）每个user出现的次数以及每个user对应的ip数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Hadoop集群可以运行的3个模式？

单机（本地）模式

伪分布式模式

全分布式模式

2.单机（本地）模式中的注意点？

在单机模式（standalone）中不会存在守护进程，所有东西都运行在一个JVM上。这里同样没有DFS，使用的是本地文件系统。单机模式适用于开发过程中运行MapReduce程序，这也是最少使用的一个模式。

3.伪分布模式中的注意点？

伪分布式（Pseudo）适用于开发和测试环境，在这个模式中，所有守护进程都在同一台机器上运行。

4.VM是否可以称为Pseudo？

不是，两个事物，同时Pseudo只针对Hadoop。

5.全分布模式又有什么注意点？

全分布模式通常被用于生产环境，这里我们使用N台主机组成一个Hadoop集群，Hadoop守护进程运行在每台主机之上。这里会存在Namenode运行的主机，Datanode运行的主机，以及tasktracker运行的主机。在分布式环境下，主节点和从节点会分开。

6.Hadoop是否遵循UNIX模式？

是的，在UNIX用例下，Hadoop还拥有“conf”目录。

7.Hadoop安装在什么目录下？

Cloudera和Apache使用相同的目录结构，Hadoop被安装在cdusrlibhadoop-0.20。

8.Namenode、Jobtracker和tasktracker的端口号是？

Namenode，70；Jobtracker，30；Tasktracker，60。

9.Hadoop的核心配置是什么？

Hadoop的核心配置通过两个xml文件来完成：1，hadoop-default.xml；2，hadoop-site.xml。这些文件都使用xml格式，因此每个xml中都有一些属性，包括名称和值，但是当下这些文件都已不复存在。

10.那当下又该如何配置？

Hadoop现在拥有3个配置文件：1，core-site.xml；2，hdfs-site.xml；3，mapred-site.xml。这些文件都保存在conf子目录下。

11.RAM的溢出因子是？

溢出因子（Spillfactor）是临时文件中储存文件的大小，也就是Hadoop-temp目录。

12.fs.mapr.working.dir只是单一的目录？

fs.mapr.working.dir只是一个目录。

13.hdfs-site.xml的3个主要属性？

dfs.name.dir决定的是元数据存储的路径以及DFS的存储方式（磁盘或是远端）

dfs.data.dir决定的是数据存储的路径

fs.checkpoint.dir用于第二Namenode

14.如何退出输入模式？

退出输入的方式有：1，按ESC；2，键入q（如果你没有输入任何当下）或者键入wq（如果你已经输入当下），并且按下Enter。

15.当你输入hadoopfsck造成“connectionrefusedjavaexception’”时，系统究竟发生了什么？

这意味着Namenode没有运行在你的VM之上。

16.我们使用Ubuntu及Cloudera，那么我们该去哪里下载Hadoop，或者是默认就与Ubuntu一起安装？

这个属于Hadoop的默认配置，你必须从Cloudera或者Edureka的dropbox下载，然后在你的系统上运行。当然，你也可以自己配置，但是你需要一个Linuxbox，Ubuntu或者是RedHat。在Cloudera网站或者是Edureka的Dropbox中有安装步骤。

17.“jps”命令的用处？

这个命令可以检查Namenode、Datanode、TaskTracker、JobTracker是否正常工作。

18.如何重启Namenode？

点击stop-all.sh，再点击start-all.sh。

键入sudohdfs（Enter），su-hdfs（Enter），etcinit.dha（Enter），及etcinit.dhadoop-0.20-namenodestart（Enter）。

19.Fsck的全名？

全名是：FileSystemCheck。

20.如何检查Namenode是否正常运行？

如果要检查Namenode是否正常工作，使用命令etcinit.dhadoop-0.20-namenodestatus或者就是简单的jps。

21.mapred.job.tracker命令的作用？

可以让你知道哪个节点是JobTracker。

22.etcinit.d命令的作用是？

etcinit.d说明了守护进程（服务）的位置或状态，其实是LINUX特性，和Hadoop关系不大。

23.如何在浏览器中查找Namenode？

如果你确实需要在浏览器中查找Namenode，你不再需要localhost8021，Namenode的端口号是50070。

24.如何从SU转到Cloudera？

从SU转到Cloudera只需要键入exit。

25.启动和关闭命令会用到哪些文件？

Slaves及Masters。

26.Slaves由什么组成？

Slaves由主机的列表组成，每台1行，用于说明数据节点。

27.Masters由什么组成？

Masters同样是主机的列表组成，每台一行，用于说明第二Namenode服务器。

28.hadoop-env.sh是用于做什么的？

hadoop-env.sh提供了Hadoop中.JAVA\_HOME的运行环境。

29.Master文件是否提供了多个入口？

是的你可以拥有多个Master文件接口。

30.Hadoop-env.sh文件当下的位置？

hadoop-env.sh现在位于conf。

31.在Hadoop\_PID\_DIR中，PID代表了什么？

PID代表了“ProcessID”。

32.varhadooppids用于做什么？

varhadooppids用来存储PID。

33.hadoop-metrics.properties文件的作用是？

hadoop-metrics.properties被用做“Reporting”，控制Hadoop报告，初始状态是“nottoreport”。

34.Hadoop需求什么样的网络？

Hadoop核心使用Shell（SSH）来驱动从节点上的服务器进程，并在主节点和从节点之间使用password-lessSSH连接。

35.全分布式环境下为什么需求password-lessSSH？

这主要因为集群中通信过于频繁，JobTracker需要尽可能快的给TaskTracker发布任务。

36.这会导致安全问题吗？

完全不用担心。Hadoop集群是完全隔离的，通常情况下无法从互联网进行操作。与众不同的配置，因此我们完全不需要在意这种级别的安全漏洞，比如说通过互联网侵入等等。Hadoop为机器之间的连接提供了一个相对安全的方式。

37.SSH工作的端口号是？

SSH工作的端口号是NO.22，当然可以通过它来配置，22是默认的端口号。

38.SSH中的注意点还包括？

SSH只是个安全的shell通信，可以把它当做NO.22上的一种协议，只需要配置一个密码就可以安全的访问。

39.为什么SSH本地主机需要密码？

在SSH中使用密码主要是增加安全性，在某些情况下也根本不会设置密码通信。

40.如果在SSH中添加key，是否还需要设置密码？

是的，即使在SSH中添加了key，还是需要设置密码。

41.假如Namenode中没有数据会怎么样？

没有数据的Namenode就不能称之为Namenode，通常情况下，Namenode肯定会有数据。

42.当JobTracker宕掉时，Namenode会发生什么？

当JobTracker失败时，集群仍然可以正常工作，只要Namenode没问题。

43.是客户端还是Namenode决定输入的分片？

这并不是客户端决定的，在配置文件中以及决定分片细则。

44.是否可以自行搭建Hadoop集群？

是的，只要对Hadoop环境足够熟悉，你完全可以这么做。

45.是否可以在Windows上运行Hadoop？

你最好不要这么做，RedHatLinux或者是Ubuntu才是Hadoop的最佳操作系统。在Hadoop安装中，Windows通常不会被使用，因为会出现各种各样的问题。因此，Windows绝对不是Hadoop的推荐系统。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

问题导读

1、当前集群的可用资源不能满足应用程序的需求，怎么解决？

2、内存里堆的东西太多了，有什么好办法吗？

http://www.aboutyun.com/static/image/hrline/4.gif

1、WARNTaskSchedulerImpl:Initialjobhasnotacceptedanyresources;checkyourclusteruitoensurethatworkersareregisteredandhavesufficientmemory

当前的集群的可用资源不能满足应用程序所请求的资源。

资源分2类：cores和ram

Core代表对执行可用的executorslots

Ram代表每个Worker上被需要的空闲内存来运行你的Application。

解决方法：

应用不要请求多余空闲可用资源的

关闭掉已经执行结束的Application

2、Applicationisn’tusingalloftheCores:HowtosettheCoresusedbyaSparkApp

设置每个App所能获得的core

解决方法：

spark-env.sh里设置spark.deploy.defaultCores

或

spark.cores.max

3、SparkExecutorOOM:HowtosetMemoryParametersonSpark

OOM是内存里堆的东西太多了

1、增加job的并行度，即增加job的partition数量，把大数据集切分成更小的数据，可以减少一次性load到内存中的数据量。InputFomart，getSplit来确定。

2、spark.storage.memoryFraction

管理executor中RDD和运行任务时的内存比例，如果shuffle比较小，只需要一点点shufflememory，那么就调大这个比例。默认是0.6。不能比老年代还要大。大了就是浪费。

3、spark.executor.memory如果还是不行，那么就要加Executor的内存了，改完executor内存后，这个需要重启。

4、SharkServer/LongRunningApplicationMetadataCleanup

Spark程序的元数据是会往内存中无限存储的。spark.cleaner.ttl来防止OOM，主要出现在SparkSteaming和SharkServer里。

exportSPARK\_JAVA\_OPTS+="-Dspark.kryoserializer.buffer.mb=10-Dspark.cleaner.ttl=43200"

5、ClassNotFound:ClasspathIssues

问题1、缺少jar，不在classpath里。

问题2、jar包冲突，同一个jar不同版本。

解决1：

将所有依赖jar都打入到一个fatJar包里，然后手动设置依赖到指定每台机器的DIR。

valconf=newSparkConf().setAppName(appName).setJars(Seq(System.getProperty("user.dir")+"/target/scala-2.10/sparktest.jar"))

解决2：

把所需要的依赖jar包都放到defaultclasspath里，分发到各个workernode上。

关于性能优化：

第一个是sort-basedshuffle。这个功能大大的减少了超大规模作业在shuffle方面的内存占用量，使得我们可以用更多的内存去排序。

第二个是新的基于Netty的网络模块取代了原有的NIO网络模块。这个新的模块提高了网络传输的性能，并且脱离JVM的GC自己管理内存，降低了GC频率。

第三个是一个独立于Sparkexecutor的externalshuffleservice。这样子executor在GC的时候其他节点还可以通过这个service来抓取shuffle数据，所以网络传输本身不受到GC的影响。

过去一些的参赛系统软件方面的处理都没有能力达到硬件的瓶颈，甚至对硬件的利用率还不到10%。而这次我们的参赛系统在map期间用满了3GB/s的硬盘带宽，达到了这些虚拟机上八块SSD的瓶颈，在reduce期间网络利用率到了1.1GB/s，接近物理极限。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Spark的四大组件下面哪个不是(D)

A.SparkStreamingBMlib

CGraphxDSparkR

2.下面哪个端口不是spark自带服务的端口(C)

A.8080B.4040C.8090D.18080

3.spark1.4版本的最大变化(B)

AsparksqlRelease版本B引入SparkR

CDataFrameD支持动态资源分配

4.SparkJob默认的调度模式(A)

AFIFOBFAIR

C无D运行时指定

5.哪个不是本地模式运行的个条件(D)

Aspark.localExecution.enabled=trueB显式指定本地运行CfinalStage无父StageDpartition默认值

6.下面哪个不是RDD的特点(C)

A.可分区B可序列化C可修改D可持久化

7.关于广播变量，下面哪个是错误的(D)

A任何函数调用B是只读的C存储在各个节点D存储在磁盘或HDFS

8.关于累加器，下面哪个是错误的(D)

A支持加法B支持数值类型

C可并行D不支持自定义类型

9.Spark支持的分布式部署方式中哪个是错误的(D)

AstandaloneBsparkonmesos

CsparkonYARNDSparkonlocal

10.Stage的Task的数量由什么决定(A)

APartitionBJobCStageDTaskScheduler

11.下面哪个操作是窄依赖(B)

AjoinBfilter

CgroupDsort

12.下面哪个操作肯定是宽依赖(C)

AmapBflatMap

CreduceByKeyDsample

13.spark的master和worker通过什么方式进行通信的？(D)

AhttpBnioCnettyDAkka

14默认的存储级别(A)

AMEMORY\_ONLYBMEMORY\_ONLY\_SER

CMEMORY\_AND\_DISKDMEMORY\_AND\_DISK\_SER

15spark.deploy.recoveryMode不支持那种(D)

A.ZooKeeperB.FileSystem

DNONEDHadoop

16.下列哪个不是RDD的缓存方法(C)

Apersist()BCache()

CMemory()

17.Task运行在下来哪里个选项中Executor上的工作单元(C)

ADriverprogramB.sparkmaster

C.workernodeDClustermanager

18.hive的元数据存储在derby和MySQL中有什么区别(B)

A.没区别B.多会话C.支持网络环境D数据库的区别

19.DataFrame和RDD最大的区别(B)

A.科学统计支持B.多了schema

C.存储方式不一样D.外部数据源支持

20.Master的ElectedLeader事件后做了哪些操作(D)

A.通知driverB.通知worker

C.注册applicationD.直接ALIVE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

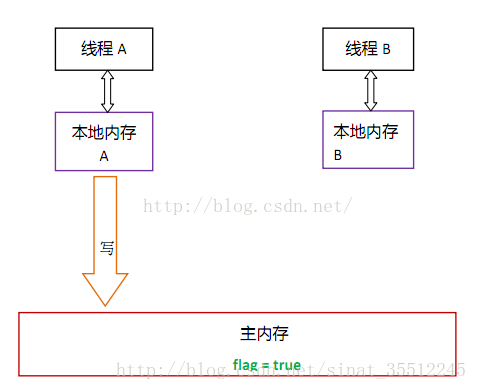
**Volatile的特征：**

A、禁止指令重排（有例外）

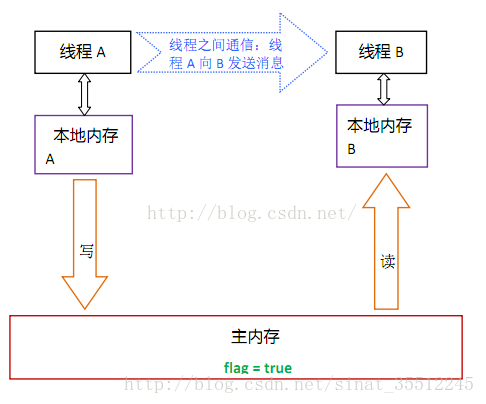
B、可见性

**Volatile的内存语义：**

当写一个volatile变量时，JMM会把线程对应的本地内存中的共享变量值刷新到主内存。



当读一个volatile变量时，JMM会把线程对应的本地内存置为无效，线程接下来将从主内存中读取共享变量。



**Volatile的重排序**

1、当第二个操作为volatile写操做时,不管第一个操作是什么(普通读写或者volatile读写),都不能进行重排序。这个规则确保volatile写之前的所有操作都不会被重排序到volatile之后;

2、当第一个操作为volatile读操作时,不管第二个操作是什么,都不能进行重排序。这个规则确保volatile读之后的所有操作都不会被重排序到volatile之前;

3、当第一个操作是volatile写操作时,第二个操作是volatile读操作,不能进行重排序。

这个规则和前面两个规则一起构成了:两个volatile变量操作不能够进行重排序；

除以上三种情况以外可以进行重排序。

比如：

1、第一个操作是普通变量读/写,第二个是volatile变量的读；

2、第一个操作是volatile变量的写,第二个是普通变量的读/写；

**内存屏障/内存栅栏**

内存屏障（MemoryBarrier，或有时叫做内存栅栏，MemoryFence）是一种CPU指令，用于控制特定条件下的重排序和内存可见性问题。[Java](http://lib.csdn.net/base/java)编译器也会根据内存屏障的规则禁止重排序。（也就是让一个CPU处理单元中的内存状态对其它处理单元可见的一项技术。）

内存屏障可以被分为以下几种类型：

LoadLoad屏障：对于这样的语句Load1;LoadLoad;Load2，在Load2及后续读取操作要读取的数据被访问前，保证Load1要读取的数据被读取完毕。

StoreStore屏障：对于这样的语句Store1;StoreStore;Store2，在Store2及后续写入操作执行前，保证Store1的写入操作对其它处理器可见。

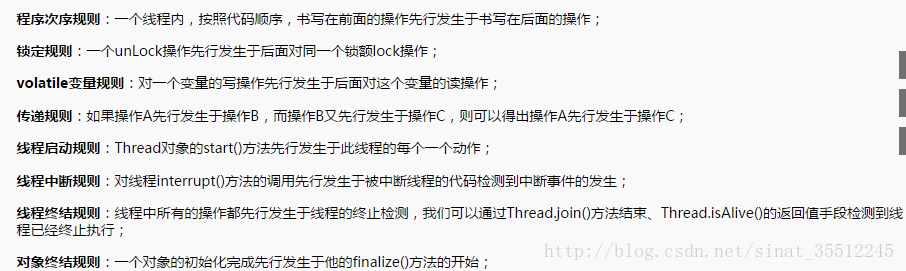
LoadStore屏障：对于这样的语句Load1;LoadStore;Store2，在Store2及后续写入操作被刷出前，保证Load1要读取的数据被读取完毕。

StoreLoad屏障：对于这样的语句Store1;StoreLoad;Load2，在Load2及后续所有读取操作执行前，保证Store1的写入对所有处理器可见。它的开销是四种屏障中最大的。

在大多数处理器的实现中，这个屏障是个万能屏障，兼具其它三种内存屏障的功能。

内存屏障阻碍了CPU采用优化技术来降低内存操作延迟，必须考虑因此带来的性能损失。为了达到最佳性能，最好是把要解决的问题模块化，这样处理器可以按单元执行任务，然后在任务单元的边界放上所有需要的内存屏障。采用这个方法可以让处理器不受限的执行一个任务单元。合理的内存屏障组合还有一个好处是：缓冲区在第一次被刷后开销会减少，因为再填充改缓冲区不需要额外工作了。

**happens-before原则**



**Java是如何实现跨平台的？**

跨平台是怎样实现的呢？这就要谈及Java虚拟机（[Java](http://lib.csdn.net/base/java)VirtualMachine，简称JVM）。

JVM也是一个软件，不同的平台有不同的版本。我们编写的Java源码，编译后会生成一种.class文件，称为字节码文件。Java虚拟机就是负责将字节码文件翻译成特定平台下的机器码然后运行。也就是说，只要在不同平台上安装对应的JVM，就可以运行字节码文件，运行我们编写的Java程序。

而这个过程中，我们编写的Java程序没有做任何改变，仅仅是通过JVM这一”中间层“，就能在不同平台上运行，真正实现了”一次编译，到处运行“的目的。

JVM是一个”桥梁“，是一个”中间件“，是实现跨平台的关键，Java代码首先被编译成字节码文件，再由JVM将字节码文件翻译成机器语言，从而达到运行Java程序的目的。

注意：编译的结果不是生成机器码，而是生成字节码，字节码不能直接运行，必须通过JVM翻译成机器码才能运行。不同平台下编译生成的字节码是一样的，但是由JVM翻译成的机器码却不一样。

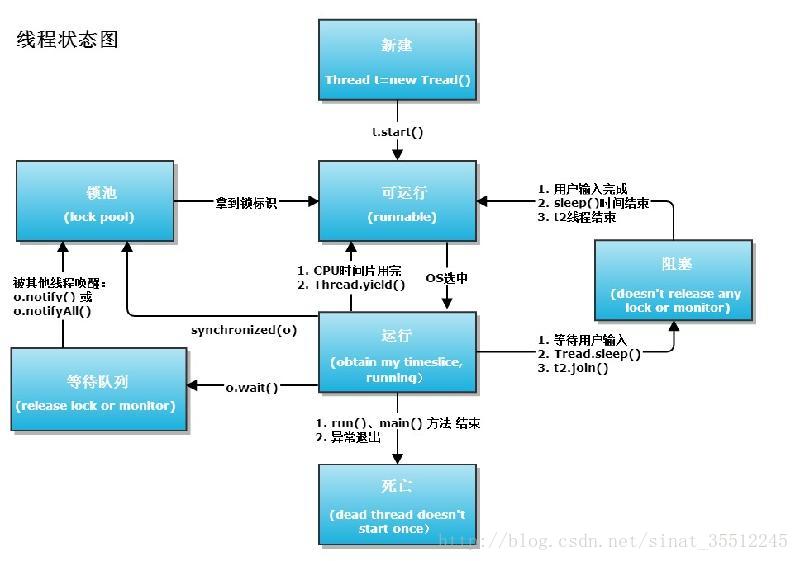
所以，运行Java程序必须有JVM的支持，因为编译的结果不是机器码，必须要经过JVM的再次翻译才能执行。即使你将Java程序打包成可执行文件（例如.exe），仍然需要JVM的支持。

注意：跨平台的是Java程序，不是JVM。JVM是用C/C++开发的，是编译后的机器码，不能跨平台，不同平台下需要安装不同版本的JVM。

**手机扫二维码登录是怎么实现的？**

友情链接：[扫码登录是如何实现的？](http://www.jianshu.com/p/7f072ac61763)

**Java线程有哪些状态，这些状态之间是如何转化的？**



1. 新建(new)：新创建了一个线程对象。
2. 可运行(runnable)：线程对象创建后，其他线程(比如main线程）调用了该对象的start()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获取cpu的使用权。
3. 运行(running)：可运行状态(runnable)的线程获得了cpu时间片（timeslice），执行程序代码。
4. 阻塞(block)：阻塞状态是指线程因为某种原因放弃了cpu使用权，也即让出了cputimeslice，暂时停止运行。直到线程进入可运行(runnable)状态，才有机会再次获得cputimeslice转到运行(running)状态。阻塞的情况分三种：

(一).等待阻塞：运行(running)的线程执行o.wait()方法，JVM会把该线程放入等待队列(waittingqueue)中。

(二).同步阻塞：运行(running)的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池(lockpool)中。

(三).其他阻塞：运行(running)的线程执行Thread.sleep(longms)或t.join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入可运行(runnable)状态。

1. 死亡(dead)：线程run()、main()方法执行结束，或者因异常退出了run()方法，则该线程结束生命周期。死亡的线程不可再次复生。

**List接口、Set接口和Map接口的区别**

友情链接：[List接口、Set接口和Map接口的区别](http://blog.csdn.net/zcg_java/article/details/43232251)

**Cookie和Session的区别？**

友情链接：[Cookies和Session的区别](http://blog.csdn.net/axin66ok/article/details/6175522)

**Java中的equals和hashCode方法详解**

友情链接：[Java提高篇——equals()与hashCode()方法详解](http://www.cnblogs.com/Qian123/p/5703507.html)

**Java中CAS**[**算法**](http://lib.csdn.net/base/datastructure)

友情链接：[乐观的并发策略——基于CAS的自旋](http://www.kancloud.cn/seaboat/java-concurrent/117870)

**TimSort原理**

友情链接：[TimSort原理](http://baike.baidu.com/link?url=4tXLGfO_VjPOOmWTZxhiGg5EyojbWvysoIPtbxqtyEpuwoVOtl43_15ds0XYHTPrQlkucF-yr_H5XztSB9KTB2bXluoWjrCgY22SbikKiTW)

**comparable与comparator的区别？**

友情链接：[Comparable和Comparator的区别](http://www.cnblogs.com/szlbm/p/5504634.html)

**手写单例模式（线程安全）**

友情链接：[快速理解Java中的五种单例模式](http://www.cnblogs.com/hupp/p/4487521.html)

**Java线程间的通信方式？**

友情链接：[Java多线程（七）线程间的通信——wait及notify方法](http://www.cnblogs.com/mengdd/archive/2013/02/20/2917956.html)

友情链接：[Java线程间的通信方式详解](http://www.jb51.net/article/84213.htm)

**Java8的内存分代改进**

友情链接：[Java7、Java8的堆内存有啥变化？](http://blog.csdn.net/chlu113/article/details/51890469)

**对Java内存模型的理解以及其在并发当中的作用？**

友情链接：[对Java内存模型的理解以及其在并发当中的作用？](http://www.cnblogs.com/_popc/p/6096517.html)

**Arrays和Collections对于sort的不同实现原理？**

1、Arrays.sort()

该算法是一个经过调优的快速排序，此算法在很多数据集上提供N\*log(N)的性能，这导致其他快速排序会降低二次型性能。

2、Collections.sort()

该算法是一个经过修改的合并排序算法（其中，如果低子列表中的最高元素效益高子列表中的最低元素，则忽略合并）。此算法可提供保证的N\*log(N)的性能，此实现将指定列表转储到一个数组中，然后再对数组进行排序，在重置数组中相应位置处每个元素的列表上进行迭代。这避免了由于试图原地对链接列表进行排序而产生的n2log(n)性能。

**Java中object常用方法**

1、clone()

2、equals()

3、finalize()

4、getclass()

5、hashcode()

6、notify()

7、notifyAll()

8、toString()

**对于Java中多态的理解**

所谓多态就是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量到底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。

多态的定义：指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式。（发送消息就是函数调用）

Java实现多态有三个必要条件：继承、重写、父类引用指向子类对象。

继承：在多态中必须存在有继承关系的子类和父类。

重写：子类对父类中某些方法进行重新定义，在调用这些方法时就会调用子类的方法。

父类引用指向子类对象：在多态中需要将子类的引用赋给父类对象，只有这样该引用才能够具备技能调用父类的方法和子类的方法。

实现多态的技术称为：动态绑定（dynamicbinding），是指在执行期间判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。

多态的作用：消除类型之间的耦合关系。

**Session机制？**

友情链接：[Session机制详解](http://justsee.iteye.com/blog/1570652)

**Java序列化与反序列化是什么？为什么需要序列化与反序列化？如何实现Java序列化与反序列化？**

友情链接：[Java序列化与反序列化](http://blog.csdn.net/wangloveall/article/details/7992448/)

[**spring**](http://lib.csdn.net/base/javaee)**AOP实现原理？**

友情链接：[SpringAOP实现原理](http://blog.csdn.net/moreevan/article/details/11977115/)

**Servlet工作原理？**

友情链接：[Servlet工作原理解析](http://www.ibm.com/developerworks/cn/java/j-lo-servlet/)

[**java**](http://lib.csdn.net/base/java)**NIO和IO的区别？**

友情链接：[JavaNIO和IO的区别](http://www.jb51.net/article/50621.htm)

**Java中堆内存和栈内存区别？**

友情链接：[Java中堆内存和栈内存详解](http://www.cnblogs.com/whgw/archive/2011/09/29/2194997.html)

**反射讲一讲，主要是概念,都在哪需要反射机制，反射的性能，如何优化？**

反射机制的定义：

是在运行状态中，对于任意的一个类，都能够知道这个类的所有属性和方法，对任意一个对象都能够通过反射机制调用一个类的任意方法，这种动态获取类信息及动态调用类对象方法的功能称为java的反射机制。

反射的作用：

1、动态地创建类的实例，将类绑定到现有的对象中，或从现有的对象中获取类型。

2、应用程序需要在运行时从某个特定的程序集中载入一个特定的类。

**如何保证RESTfulAPI安全性？**

友情链接：[如何设计好的RESTfulAPI之安全性](http://blog.csdn.net/ywk253100/article/details/25654101)

**如何预防**[**MySQL**](http://lib.csdn.net/base/mysql)**注入？**

友情链接：[MySQL及SQL注入与防范方法](http://www.jb51.net/article/87948.htm)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

spark-submit的时候如何引入外部jar包

sparkshuffle的具体过程，你知道几种shuffle方式

spark如何防止内存溢出

cache和pesist的区别

怎么处理数据倾斜

​简要描述Spark分布式集群搭建的步骤

spark使用：

1）当前文件a.text的格式为，请统计每个单词出现的个数、计算第四列每个元素出现的个数

A,b,c,d

B,b,f,e

A,a,c,f

2）在（url,user）的键值对中，如

a.text

127.0.0.1xiaozhang

127.0.0.1xiaoli

127.0.0.2wangwu

127.0.0.3lisi

…..

B.text

127.0.0.4lixiaolu

127.0.0.5lisi

127.0.0.3zhangsan

每个文件至少有1000万行，请用程序完成一下工作，

1）各个文件的ip数

2)出现在b.text而没有出现在a.text的IP

3)每个user出现的次数以及每个user对应的IP的个数4)对应IP数最多的

## SparkCore面试篇01

### 1.Sparkmaster使用zookeeper进行HA的，有哪些元数据保存在Zookeeper？

答：spark通过这个参数spark.deploy.zookeeper.dir指定master元数据在zookeeper中保存的位置，包括Worker，Driver和Application以及Executors。

standby节点要从zk中，获得元数据信息，恢复集群运行状态，才能对外继续提供服务，作业提交资源申请等，在恢复前是不能接受请求的。另外，Master切换需要注意2点

1）在Master切换的过程中，所有的已经在运行的程序皆正常运行！因为SparkApplication在运行前就已经通过ClusterManager获得了计算资源，所以在运行时Job本身的调度和处理和Master是没有任何关系的！

2）在Master的切换过程中唯一的影响是不能提交新的Job：一方面不能够提交新的应用程序给集群，因为只有ActiveMaster才能接受新的程序的提交请求；另外一方面，已经运行的程序中也不能够因为Action操作触发新的Job的提交请求；

### 2.SparkmasterHA主从切换过程不会影响集群已有的作业运行，为什么？

答：因为程序在运行之前，已经申请过资源了，driver和**Executors**通讯，不需要和master进行通讯的。

### 3.SparkonMesos中，什么是的粗粒度分配，什么是细粒度分配，各自的优点和缺点是什么？

答：1）粗粒度：启动时就分配好资源，程序启动，后续具体使用就使用分配好的资源，不需要再分配资源；好处：作业特别多时，资源复用率高，适合粗粒度；不好：容易资源浪费，假如一个job有1000个task，完成了999个，还有一个没完成，那么使用粗粒度，999个资源就会闲置在那里，资源浪费。2）细粒度分配：用资源的时候分配，用完了就立即回收资源，启动会麻烦一点，启动一次分配一次，会比较麻烦。

### 4.如何配置sparkmaster的HA？

1)配置zookeeper

2)修改spark\_env.sh文件,spark的master参数不在指定，添加如下代码到各个master节点

exportSPARK\_DAEMON\_JAVA\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER-Dspark.deploy.zookeeper.url=zk01:2181,zk02:2181,zk03:2181-Dspark.deploy.zookeeper.dir=/spark"

3)将spark\_env.sh分发到各个节点

4)找到一个master节点，执行./start-all.sh，会在这里启动主master,其他的master备节点，启动master命令:./sbin/start-master.sh

5)提交程序的时候指定master的时候要指定三台master，例如

./spark-shell--masterspark://master01:7077,master02:7077,master03:7077

### 5.ApacheSpark有哪些常见的稳定版本，Spark1.6.0的数字分别代表什么意思？

答：常见的大的稳定版本有Spark1.3,Spark1.6,Spark2.0，**Spark1.6.0的数字含义**

1）第一个数字：1

majorversion:代表大版本更新，一般都会有一些api的变化，以及大的优化或是一些结构的改变；

2）第二个数字：6

minorversion:代表小版本更新，一般会新加api，或者是对当前的api就行优化，或者是其他内容的更新，比如说WEBUI的更新等等；

3）第三个数字：0

patchversion，代表修复当前小版本存在的一些bug，基本不会有任何api的改变和功能更新；记得有一个大神曾经说过，如果要切换spark版本的话，最好选patchversion非0的版本，因为一般类似于1.2.0,…1.6.0这样的版本是属于大更新的，有可能会有一些隐藏的bug或是不稳定性存在，所以最好选择1.2.1,…1.6.1这样的版本。

通过版本号的解释说明，可以很容易了解到，spark2.1.1的发布时是针对大版本2.1做的一些bug修改，不会新增功能，也不会新增API，会比2.1.0版本更加稳定。

### 6.driver的功能是什么？

答：1）一个Spark作业运行时包括一个Driver进程，也是作业的主进程，具有main函数，并且有SparkContext的实例，是程序的人口点；2）功能：负责向集群申请资源，向master注册信息，负责了作业的调度，，负责作业的解析、生成Stage并调度Task到Executor上。包括DAGScheduler，TaskScheduler。

### 7.spark的有几种部署模式，每种模式特点？

1）本地模式

Spark不一定非要跑在hadoop集群，可以在本地，起多个线程的方式来指定。将Spark应用以多线程的方式直接运行在本地，一般都是为了方便调试，本地模式分三类

·local：只启动一个executor

·local[k]:启动k个executor

·local[\*]：启动跟cpu数目相同的executor

2)standalone模式

分布式部署集群，自带完整的服务，资源管理和任务监控是Spark自己监控，这个模式也是其他模式的基础，

3)Sparkonyarn模式

分布式部署集群，资源和任务监控交给yarn管理，但是目前仅支持粗粒度资源分配方式，包含cluster和client运行模式，cluster适合生产，driver运行在集群子节点，具有容错功能，client适合调试，dirver运行在客户端

4）SparkOnMesos模式。官方推荐这种模式（当然，原因之一是血缘关系）。正是由于Spark开发之初就考虑到支持Mesos，因此，目前而言，Spark运行在Mesos上会比运行在YARN上更加灵活，更加自然。用户可选择两种调度模式之一运行自己的应用程序：

1)粗粒度模式（Coarse-grainedMode）：每个应用程序的运行环境由一个Dirver和若干个Executor组成，其中，每个Executor占用若干资源，内部可运行多个Task（对应多少个“slot”）。应用程序的各个任务正式运行之前，需要将运行环境中的资源全部申请好，且运行过程中要一直占用这些资源，即使不用，最后程序运行结束后，回收这些资源。

2)细粒度模式（Fine-grainedMode）：鉴于粗粒度模式会造成大量资源浪费，SparkOnMesos还提供了另外一种调度模式：细粒度模式，这种模式类似于现在的云计算，思想是按需分配。

### 8.Spark技术栈有哪些组件，每个组件都有什么功能，适合什么应用场景？

**答：可以画一个这样的技术栈图先，然后分别解释下每个组件的功能和场景**

1）Sparkcore：是其它组件的基础，spark的内核，主要包含：有向循环图、RDD、Lingage、Cache、broadcast等，并封装了底层通讯框架，是Spark的基础。

2）SparkStreaming是一个对实时数据流进行高通量、容错处理的流式处理系统，可以对多种数据源（如Kdfka、Flume、Twitter、Zero和TCP套接字）进行类似Map、Reduce和Join等复杂操作，将流式计算分解成一系列短小的批处理作业。

3）Sparksql：Shark是SparkSQL的前身，SparkSQL的一个重要特点是其能够统一处理关系表和RDD，使得开发人员可以轻松地使用SQL命令进行外部查询，同时进行更复杂的数据分析

4）BlinkDB：是一个用于在海量数据上运行交互式SQL查询的大规模并行查询引擎，它允许用户通过权衡数据精度来提升查询响应时间，其数据的精度被控制在允许的误差范围内。

5）MLBase是Spark生态圈的一部分专注于机器学习，让机器学习的门槛更低，让一些可能并不了解机器学习的用户也能方便地使用MLbase。MLBase分为四部分：MLlib、MLI、MLOptimizer和MLRuntime。

6）GraphX是Spark中用于图和图并行计算

### 9.Spark中Work的主要工作是什么？

答：主要功能：管理当前节点内存，CPU的使用状况，接收master分配过来的资源指令，通过ExecutorRunner启动程序分配任务，worker就类似于包工头，管理分配新进程，做计算的服务，相当于process服务。需要注意的是：

1）worker不会汇报当前信息给master，worker心跳给master主要只有workid，它不会发送资源信息以心跳的方式给mater，master分配的时候就知道work，只有出现故障的时候才会发送资源。

2）worker不会运行代码，具体运行的是Executor是可以运行具体appliaction写的业务逻辑代码，操作代码的节点，它不会运行程序的代码的。

### 10.Spark为什么比mapreduce快？

答：

1）基于内存计算，减少低效的磁盘交互；

2）高效的调度算法，基于DAG；

3)容错机制Linage，精华部分就是DAG和Lineage

### 11.简单说一下hadoop和spark的shuffle相同和差异？

答：1）从high-level的角度来看，两者并没有大的差别。都是将mapper（Spark里是ShuffleMapTask）的输出进行partition，不同的partition送到不同的reducer（Spark里reducer可能是下一个stage里的ShuffleMapTask，也可能是ResultTask）。Reducer以内存作缓冲区，边shuffle边aggregate数据，等到数据aggregate好以后进行reduce()（Spark里可能是后续的一系列操作）。

2）从low-level的角度来看，两者差别不小。HadoopMapReduce是sort-based，进入combine()和reduce()的records必须先sort。这样的好处在于combine/reduce()可以处理大规模的数据，因为其输入数据可以通过外排得到（mapper对每段数据先做排序，reducer的shuffle对排好序的每段数据做归并）。目前的Spark默认选择的是hash-based，通常使用HashMap来对shuffle来的数据进行aggregate，不会对数据进行提前排序。如果用户需要经过排序的数据，那么需要自己调用类似sortByKey()的操作；如果你是Spark1.1的用户，可以将spark.shuffle.manager设置为sort，则会对数据进行排序。在Spark1.2中，sort将作为默认的Shuffle实现。

3）从实现角度来看，两者也有不少差别。HadoopMapReduce将处理流程划分出明显的几个阶段：map(),spill,merge,shuffle,sort,reduce()等。每个阶段各司其职，可以按照过程式的编程思想来逐一实现每个阶段的功能。在Spark中，没有这样功能明确的阶段，只有不同的stage和一系列的transformation()，所以spill,merge,aggregate等操作需要蕴含在transformation()中。

如果我们将map端划分数据、持久化数据的过程称为shufflewrite，而将reducer读入数据、aggregate数据的过程称为shuffleread。那么在Spark中，问题就变为怎么在job的逻辑或者物理执行图中加入shufflewrite和shuffleread的处理逻辑？以及两个处理逻辑应该怎么高效实现？

Shufflewrite由于不要求数据有序，shufflewrite的任务很简单：将数据partition好，并持久化。之所以要持久化，一方面是要减少内存存储空间压力，另一方面也是为了fault-tolerance。

### 12.Mapreduce和Spark的都是并行计算，那么他们有什么相同和区别

答：两者都是用mr模型来进行并行计算:

1)hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束。

2)spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job。这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算。

3)hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系。

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错。

### 13.RDD机制？

答：rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。

所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。

**rdd执行过程中会形成dag图，然后形成lineage保证容错性等**。从物理的角度来看rdd存储的是block和node之间的映射。

### 14、spark有哪些组件？

答：主要有如下组件：

1）master：管理集群和节点，不参与计算。

2）worker：计算节点，进程本身不参与计算，和master汇报。

3）Driver：运行程序的main方法，创建sparkcontext对象。向资源管理器申请资源，进行stage划分和任务分发

4）sparkcontext：控制整个application的生命周期，包括dagsheduler和taskscheduler等组件。

5）client：用户提交程序的入口。

### 15、spark工作机制？

答：用户在client端提交作业后，会由Driver运行main方法并创建sparkcontext上下文。

执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入taskscheduler。taskscheduler会将stage划分为taskset分发到各个节点的executor中执行。

### 16、spark的优化怎么做？

答：spark调优比较复杂，但是大体可以分为三个方面来进行，1）平台层面的调优：防止不必要的jar包分发，提高数据的本地性，选择高效的存储格式如parquet，2）应用程序层面的调优：过滤操作符的优化降低过多小任务，降低单条记录的资源开销，处理数据倾斜，复用RDD进行缓存，作业并行化执行等等，3）JVM层面的调优：设置合适的资源量，设置合理的JVM，启用高效的序列化方法如kyro，增大offhead内存等等

### 17.简要描述Spark分布式集群搭建的步骤

1）准备linux环境，设置集群搭建账号和用户组，设置ssh，关闭防火墙，关闭seLinux，配置host，hostname

2）配置jdk到环境变量

3）搭建hadoop集群，如果要做masterha，需要搭建zookeeper集群

修改hdfs-site.xml,hadoop\_env.sh,yarn-site.xml,slaves等配置文件

4）启动hadoop集群，启动前要格式化namenode

5）配置spark集群，修改spark-env.xml，slaves等配置文件，拷贝hadoop相关配置到sparkconf目录下

6)启动spark集群。

### 18.什么是RDD宽依赖和窄依赖？

RDD和它依赖的parentRDD(s)的关系有两种不同的类型，即窄依赖（narrowdependency）和宽依赖（widedependency）。

1）窄依赖指的是每一个parentRDD的Partition最多被子RDD的一个Partition使用

2）宽依赖指的是多个子RDD的Partition会依赖同一个parentRDD的Partition

### 19.spark-submit的时候如何引入外部jar包

方法一：spark-submit–jars

根据spark官网，在提交任务的时候指定–jars，用逗号分开。这样做的缺点是每次都要指定jar包，如果jar包少的话可以这么做，但是如果多的话会很麻烦。

命令：spark-submit--masteryarn-client--jars\*\*\*.jar,\*\*\*.jar

方法二：extraClassPath

提交时在spark-default中设定参数，将所有需要的jar包考到一个文件里，然后在参数中指定该目录就可以了，较上一个方便很多：

spark.executor.extraClassPath=/home/hadoop/wzq\_workspace/lib/\*spark.driver.extraClassPath=/home/hadoop/wzq\_workspace/lib/\*

需要注意的是,你要在所有可能运行spark任务的机器上保证该目录存在，并且将jar包考到所有机器上。这样做的好处是提交代码的时候不用再写一长串jar了，缺点是要把所有的jar包都拷一遍。

### 20.cache和pesist的区别

答：

1）cache和persist都是用于将一个RDD进行缓存的，这样在之后使用的过程中就不需要重新计算了，可以大大节省程序运行时间；

2）cache只有一个默认的缓存级别MEMORY\_ONLY，cache调用了persist，而persist可以根据情况设置其它的缓存级别；

3）executor执行的时候，默认60%做cache，40%做task操作，persist最根本的函数，最底层的函数

### 参考资料

1.<http://blog.csdn.net/china_demon/article/details/53095480>

2.<http://www.cnblogs.com/shishanyuan/archive/2015/08/04/4700615.html>

3.<http://blog.csdn.net/u012965373/article/details/51427105>

## Sparkcore面试篇02

### 1.cache后面能不能接其他算子,它是不是action操作？

答：cache可以接其他算子，但是接了算子之后，起不到缓存应有的效果，因为会重新触发cache。

cache不是action操作

### 2.reduceByKey是不是action？

答：不是，很多人都会以为是action，reducerdd是action

### 3.数据本地性是在哪个环节确定的？

具体的task运行在那他机器上，dag划分stage的时候确定的

### 4.RDD的弹性表现在哪几点？

1）自动的进行内存和磁盘的存储切换；

2）基于Lingage的高效容错；

3）task如果失败会自动进行特定次数的重试；

4）stage如果失败会自动进行特定次数的重试，而且只会计算失败的分片；

5）checkpoint和persist，数据计算之后持久化缓存

6）数据调度弹性，DAGTASK调度和资源无关

7）数据分片的高度弹性，a.分片很多碎片可以合并成大的，b.par

### 5.常规的容错方式有哪几种类型？

1）.数据检查点,会发生拷贝，浪费资源

2）.记录数据的更新，每次更新都会记录下来，比较复杂且比较消耗性能

### 6.RDD通过Linage（记录数据更新）的方式为何很高效？

1）lazy记录了数据的来源，RDD是不可变的，且是lazy级别的，且rDD之间构成了链条，lazy是弹性的基石。由于RDD不可变，所以每次操作就产生新的rdd，不存在全局修改的问题，控制难度下降，所有有计算链条将复杂计算链条存储下来，计算的时候从后往前回溯900步是上一个stage的结束，要么就checkpoint

2）记录原数据，是每次修改都记录，代价很大如果修改一个集合，代价就很小，官方说rdd是粗粒度的操作，是为了效率，为了简化，每次都是操作数据集合，写或者修改操作，都是基于集合的rdd的写操作是粗粒度的，rdd的读操作既可以是粗粒度的也可以是细粒度，读可以读其中的一条条的记录。

3）简化复杂度，是高效率的一方面，写的粗粒度限制了使用场景如网络爬虫，现实世界中，大多数写是粗粒度的场景

### 7.RDD有哪些缺陷？

1）不支持细粒度的写和更新操作（如网络爬虫），spark写数据是粗粒度的所谓粗粒度，就是批量写入数据，为了提高效率。但是读数据是细粒度的也就是说可以一条条的读

2）不支持增量迭代计算，Flink支持

### 8.说一说Spark程序编写的一般步骤？

**答：**初始化，资源，数据源，并行化，rdd转化，action算子打印输出结果或者也可以存至相应的数据存储介质，具体的可看下图：

### 9.Spark有哪两种算子？

**答：**Transformation（转化）算子和Action（执行）算子。

### 10.Spark提交你的jar包时所用的命令是什么？

**答：**spark-submit。

### 11.Spark有哪些聚合类的算子,我们应该尽量避免什么类型的算子？

**答：**在我们的开发过程中，能避免则尽可能避免使用reduceByKey、join、distinct、repartition等会进行shuffle的算子，尽量使用map类的非shuffle算子。这样的话，没有shuffle操作或者仅有较少shuffle操作的Spark作业，可以大大减少性能开销。

### 12.你所理解的Spark的shuffle过程？

**答：从下面三点去展开**

1）shuffle过程的划分

2）shuffle的中间结果如何存储

3）shuffle的数据如何拉取过来

可以参考这篇博文：

<http://www.cnblogs.com/jxhd1/p/6528540.html>

### 13.你如何从Kafka中获取数据？

**1**)基于Receiver的方式

这种方式使用Receiver来获取数据。Receiver是使用Kafka的高层次ConsumerAPI来实现的。receiver从Kafka中获取的数据都是存储在SparkExecutor的内存中的，然后SparkStreaming启动的job会去处理那些数据。

2)基于Direct的方式

这种新的不基于Receiver的直接方式，是在Spark1.3中引入的，从而能够确保更加健壮的机制。替代掉使用Receiver来接收数据后，这种方式会周期性地查询Kafka，来获得每个topic+partition的最新的offset，从而定义每个batch的offset的范围。当处理数据的job启动时，就会使用Kafka的简单consumerapi来获取Kafka指定offset范围的数据

### 14.对于Spark中的数据倾斜问题你有什么好的方案？

1）前提是定位数据倾斜，是OOM了，还是任务执行缓慢，看日志，看WebUI

2)解决方法，有多个方面

·避免不必要的shuffle，如使用广播小表的方式，将reduce-side-join提升为map-side-join

·分拆发生数据倾斜的记录，分成几个部分进行，然后合并join后的结果

·改变并行度，可能并行度太少了，导致个别task数据压力大

·两阶段聚合，先局部聚合，再全局聚合

·自定义paritioner，分散key的分布，使其更加均匀

详细解决方案参考博文[《Spark数据倾斜优化方法》](http://mp.weixin.qq.com/s?__biz=MzIzNzI1NzY3Nw==&mid=2247484221&idx=1&sn=7e20f08bfb490b91f0920aefb29ca271&chksm=e8ca159fdfbd9c89f610dd230e07f414521b4dd13018994ee9b873421d1e8efcdc535c810225&scene=21#wechat_redirect)

### 15.RDD创建有哪几种方式？

1).使用程序中的集合创建rdd

2).使用本地文件系统创建rdd

3).使用hdfs创建rdd，

4).基于数据库db创建rdd

5).基于Nosql创建rdd，如hbase

6).基于s3创建rdd，

7).基于数据流，如socket创建rdd

如果只回答了前面三种，是不够的，只能说明你的水平还是入门级的，实践过程中有很多种创建方式。

### 16.Spark并行度怎么设置比较合适

答：spark并行度，每个core承载2~4个partition,如，32个core，那么64~128之间的并行度，也就是

设置64~128个partion，并行读和数据规模无关，只和内存使用量和cpu使用时间有关

### 17.Spark中数据的位置是被谁管理的？

答：每个数据分片都对应具体物理位置，数据的位置是被blockManager，无论数据是在磁盘，内存还是tacyan，都是由blockManager管理

### 18.Spark的数据本地性有哪几种？

答：Spark中的数据本地性有三种：

a.PROCESS\_LOCAL是指读取缓存在本地节点的数据

b.NODE\_LOCAL是指读取本地节点硬盘数据

c.ANY是指读取非本地节点数据

通常读取数据PROCESS\_LOCAL>NODE\_LOCAL>ANY，尽量使数据以PROCESS\_LOCAL或NODE\_LOCAL方式读取。其中PROCESS\_LOCAL还和cache有关，如果RDD经常用的话将该RDDcache到内存中，注意，由于cache是lazy的，所以必须通过一个action的触发，才能真正的将该RDDcache到内存中。

### 19.rdd有几种操作类型？

1）transformation，rdd由一种转为另一种rdd

2）action，触发任务执行的算子，每个action对应一个job

3）cronroller，crontroller是控制算子,cache,persist，对性能和效率的有很好的支持

三种类型，不要回答只有2中操作

### 20.Spark如何处理不能被序列化的对象？

将不能序列化的内容封装成object

### 21.collect功能是什么，其底层是怎么实现的？

答：driver通过collect把集群中各个节点的内容收集过来汇总成结果，collect返回结果是Array类型的，collect把各个节点上的数据抓过来，抓过来数据是Array型，collect对Array抓过来的结果进行合并，合并后Array中只有一个元素，是tuple类型（KV类型的）的。

### 22.Spaek程序执行，有时候默认为什么会产生很多task，怎么修改默认task执行个数？

答：1）因为输入数据有很多task，尤其是有很多小文件的时候，有多少个输入block就会有多少个task启动；

2）spark中有partition的概念，每个partition都会对应一个task，task越多，在处理大规模数据的时候，就会越有效率。不过task并不是越多越好，如果平时测试，或者数据量没有那么大，则没有必要task数量太多。

3）参数可以通过spark\_home/conf/spark-default.conf配置文件设置:

spark.sql.shuffle.partitions50spark.default.parallelism10

第一个是针对sparksql的task数量

第二个是非sparksql程序设置生效

### 23.为什么SparkApplication在没有获得足够的资源，job就开始执行了，可能会导致什么什么问题发生?

答：会导致执行该job时候集群资源不足，导致执行job结束也没有分配足够的资源，分配了部分Executor，该job就开始执行task，应该是task的调度线程和Executor资源申请是异步的；如果想等待申请完所有的资源再执行job的：需要将spark.scheduler.maxRegisteredResourcesWaitingTime设置的很大；spark.scheduler.minRegisteredResourcesRatio设置为1，但是应该结合实际考虑

否则很容易出现长时间分配不到资源，job一直不能运行的情况。

### 24.map与flatMap的区别

map：对RDD每个元素转换，文件中的每一行数据返回一个数组对象

flatMap：对RDD每个元素转换，然后再扁平化

将所有的对象合并为一个对象，文件中的所有行数据仅返回一个数组

对象，会抛弃值为null的值

### 25.列举你常用的action？

**collect，reduce,take,count,saveAsTextFile等**

### 26.Spark为什么要持久化，一般什么场景下要进行persist操作？

为什么要进行持久化？

spark所有复杂一点的算法都会有persist身影,spark默认数据放在内存，spark很多内容都是放在内存的，非常适合高速迭代，1000个步骤只有第一个输入数据，中间不产生临时数据，但分布式系统风险很高，所以容易出错，就要容错，rdd出错或者分片可以根据血统算出来，如果没有对父rdd进行persist或者cache的化，就需要重头做。

以下场景会使用persist

1）某个步骤计算非常耗时，需要进行persist持久化

2）计算链条非常长，重新恢复要算很多步骤，很好使，persist

3）checkpoint所在的rdd要持久化persist，

lazy级别，框架发现有checnkpoint，checkpoint时单独触发一个job，需要重算一遍，checkpoint前

要持久化，写个rdd.cache或者rdd.persist，将结果保存起来，再写checkpoint操作，这样执行起来会非常快，不需要重新计算rdd链条了。checkpoint之前一定会进行persist。

4）shuffle之后为什么要persist，shuffle要进性网络传输，风险很大，数据丢失重来，恢复代价很大

5）shuffle之前进行persist，框架默认将数据持久化到磁盘，这个是框架自动做的。

### 27.为什么要进行序列化

序列化可以减少数据的体积，减少存储空间，高效存储和传输数据，不好的是使用的时候要反序列化，非常消耗CPU

### 28.介绍一下join操作优化经验？

答：join其实常见的就分为两类：map-sidejoin和reduce-sidejoin。当大表和小表join时，用map-sidejoin能显著提高效率。将多份数据进行关联是数据处理过程中非常普遍的用法，不过在分布式计算系统中，这个问题往往会变的非常麻烦，因为框架提供的join操作一般会将所有数据根据key发送到所有的reduce分区中去，也就是shuffle的过程。造成大量的网络以及磁盘IO消耗，运行效率极其低下，这个过程一般被称为reduce-side-join。如果其中有张表较小的话，我们则可以自己实现在map端实现数据关联，跳过大量数据进行shuffle的过程，运行时间得到大量缩短，根据不同数据可能会有几倍到数十倍的性能提升。

备注：这个题目面试中非常非常大概率见到，务必搜索相关资料掌握，这里抛砖引玉。

### 29.介绍一下cogrouprdd实现原理，你在什么场景下用过这个rdd？

答：cogroup的函数实现:这个实现根据两个要进行合并的两个RDD操作,生成一个CoGroupedRDD的实例,这个RDD的返回结果是把相同的key中两个RDD分别进行合并操作,最后返回的RDD的value是一个Pair的实例,这个实例包含两个Iterable的值,第一个值表示的是RDD1中相同KEY的值,第二个值表示的是RDD2中相同key的值.由于做cogroup的操作,需要通过partitioner进行重新分区的操作,因此,执行这个流程时,需要执行一次shuffle的操作(如果要进行合并的两个RDD的都已经是shuffle后的rdd,同时他们对应的partitioner相同时,就不需要执行shuffle,)，

场景：表关联查询

### 30下面这段代码输出结果是什么？

--------------------------

defjoinRdd(sc:SparkContext){

valname=Array(Tuple2(1,"spark"),Tuple2(2,"tachyon"),Tuple2(3,"hadoop"))

valscore=Array(

Tuple2(1,100),

Tuple2(2,90),

Tuple2(3,80)

)

valnamerdd=sc.parallelize(name);

valscorerdd=sc.parallelize(score);

valresult=namerdd.join(scorerdd);

result.collect.foreach(println);

}

--------------------------

答案:

(1,(Spark,100))

(2,(tachyon,90))

(3,(hadoop,80))

### 二、参考资料

1.<http://m.blog.csdn.net/article/details?id=50834858>

map-side-join关联优化

## Sparkcore面试篇03

### 1.Spark使用parquet文件存储格式能带来哪些好处？

1)如果说HDFS是大数据时代分布式文件系统首选标准，那么parquet则是整个大数据时代文件存储格式实时首选标准

2)速度更快：从使用sparksql操作普通文件CSV和parquet文件速度对比上看，绝大多数情况会比使用csv等普通文件速度提升10倍左右，在一些普通文件系统无法在spark上成功运行的情况下，使用parquet很多时候可以成功运行

3)parquet的压缩技术非常稳定出色，在sparksql中对压缩技术的处理可能无法正常的完成工作（例如会导致losttask，lostexecutor）但是此时如果使用parquet就可以正常的完成

4)极大的减少磁盘I/o,通常情况下能够减少75%的存储空间，由此可以极大的减少sparksql处理数据的时候的数据输入内容，尤其是在spark1.6x中有个下推过滤器在一些情况下可以极大的减少磁盘的IO和内存的占用，（下推过滤器）

5)spark1.6xparquet方式极大的提升了扫描的吞吐量，极大提高了数据的查找速度spark1.6和spark1.5x相比而言，提升了大约1倍的速度，在spark1.6X中，操作parquet时候cpu也进行了极大的优化，有效的降低了cpu

6)采用parquet可以极大的优化spark的调度和执行。我们测试spark如果用parquet可以有效的减少stage的执行消耗，同时可以优化执行路径

### 2.Executor之间如何共享数据？

答：基于hdfs或者基于tachyon

### 3.Spark累加器有哪些特点？

1）累加器在全局唯一的，只增不减，记录全局集群的唯一状态

2）在executor中修改它，在driver读取

3）executor级别共享的，广播变量是task级别的共享

两个application不可以共享累加器，但是同一个app不同的job可以共享

### 4.如何在一个不确定的数据规模的范围内进行排序？

为了提高效率，要划分划分，划分的范围并且是有序的

要么有序，要么降序？

水塘抽样：目的是从一个集合中选取，集合非常答，适合内存

无法容纳数据的时候使用从N中抽取出K个，N是随机数

### 5.sparkhashParitioner的弊端是什么？

答:HashPartitioner分区的原理很简单，对于给定的key，计算其hashCode，并除于分区的个数取余，如果余数小于0，则用余数+分区的个数，最后返回的值就是这个key所属的分区ID；

弊端是数据不均匀，容易导致数据倾斜，极端情况下某几个分区会拥有rdd的所有数据

### 6.RangePartitioner分区的原理?

答:RangePartitioner分区则尽量保证每个分区中数据量的均匀，而且分区与分区之间是有序的，也就是说一个分区中的元素肯定都是比另一个分区内的元素小或者大；但是分区内的元素是不能保证顺序的。简单的说就是将一定范围内的数映射到某一个分区内。其原理是水塘抽样。可以参考这篇博文

https://www.iteblog.com/archives/1522.html

### 7.介绍parition和block有什么关联关系？

**答：**

**1）**hdfs中的block是分布式存储的最小单元，等分，可设置冗余，这样设计有一部分磁盘空间的浪费，但是整齐的block大小，便于快速找到、读取对应的内容；

2）**Spark**中的partion是弹性分布式数据集RDD的最小单元，RDD是由分布在各个节点上的partion组成的。partion是指的spark在计算过程中，生成的数据在计算空间内最小单元，同一份数据（RDD）的partion大小不一，数量不定，是根据application里的算子和最初读入的数据分块数量决定；

3）block位于存储空间、partion位于计算空间，block的大小是固定的、partion大小是不固定的，是从2个不同的角度去看数据。

### 8.Spark应用程序的执行过程是什么？

1)构建SparkApplication的运行环境（启动SparkContext），SparkContext向资源管理器（可以是Standalone、Mesos或YARN）注册并申请运行Executor资源；

2).资源管理器分配Executor资源并启动StandaloneExecutorBackend，Executor运行情况将随着心跳发送到资源管理器上；

3).SparkContext构建成DAG图，将DAG图分解成Stage，并把Taskset发送给TaskScheduler。Executor向SparkContext申请Task，TaskScheduler将Task发放给Executor运行同时SparkContext将应用程序代码发放给Executor。

4).Task在Executor上运行，运行完毕释放所有资源。

### 9.hbase预分区个数和spark过程中的reduce个数相同么

答：和spark的map个数相同，reduce个数如果没有设置和reduce前的map数相同。

### 10.如何理解Standalone模式下，Spark资源分配是粗粒度的？

答：spark默认情况下资源分配是粗粒度的，也就是说程序在提交时就分配好资源，后面执行的时候使用分配好的资源，除非资源出现了故障才会重新分配。比如Sparkshell启动，已提交，一注册，哪怕没有任务，worker都会分配资源给executor。

### 11.Spark如何自定义partitioner分区器？

**答：1）**spark默认实现了HashPartitioner和RangePartitioner两种分区策略，我们也可以自己扩展分区策略，自定义分区器的时候继承org.apache.spark.Partitioner类，实现类中的三个方法：

defnumPartitions:Int：这个方法需要返回你想要创建分区的个数；

defgetPartition(key:Any):Int：这个函数需要对输入的key做计算，然后返回该key的分区ID，范围一定是0到numPartitions-1；

equals()：这个是Java标准的判断相等的函数，之所以要求用户实现这个函数是因为Spark内部会比较两个RDD的分区是否一样。

2）使用，调用parttionBy方法中传入自定义分区对象

参考：http://blog.csdn.net/high2011/article/details/68491115

### 12.spark中task有几种类型？

答：2种类型：

1）resulttask类型，最后一个task，

2）shuffleMapTask类型，除了最后一个task都是

### 13.union操作是产生宽依赖还是窄依赖？

**答：窄依赖**

### 14.rangePartioner分区器特点？

答：**rangePartioner**尽量保证每个分区中数据量的均匀，而且分区与分区之间是有序的，一个分区中的元素肯定都是比另一个分区内的元素小或者大；但是分区内的元素是不能保证顺序的。简单的说就是将一定范围内的数映射到某一个分区内。RangePartitioner作用：将一定范围内的数映射到某一个分区内，在实现中，分界的**算法**尤为重要。算法对应的函数是rangeBounds

### 15.什么是二次排序，你是如何用spark实现二次排序的？（互联网公司常面）

答：就是考虑2个维度的排序，key相同的情况下如何排序，参考博文：http://blog.csdn.net/sundujing/article/details/51399606

### 16.如何使用Spark解决TopN问题？（互联网公司常面）

答：常见的面试题,参考博文：

<http://www.cnblogs.com/yurunmiao/p/4898672.html>

### 17.如何使用Spark解决分组排序问题？（互联网公司常面）

组织数据形式：

aa11

bb11

cc34

aa22

bb67

cc29

aa36

bb33

cc30

aa42

bb44

cc49

需求：

1、对上述数据按key值进行分组

2、对分组后的值进行排序

3、截取分组后值得top3位以key-value形式返回结果

答案：如下

----------------------

valgroupTopNRdd=sc.textFile("hdfs://db02:8020/user/hadoop/groupsorttop/groupsorttop.data")

groupTopNRdd.map(\_.split("")).map(x=>(x(0),x(1))).groupByKey().map(

x=>{

valxx=x.\_1

valyy=x.\_2

(xx,yy.toList.sorted.reverse.take(3))

}

).collect

---------------------

### 18.窄依赖父RDD的partition和子RDD的parition是不是都是一对一的关系？

答：不一定，除了一对一的窄依赖，还包含一对固定个数的窄依赖（就是对父RDD的依赖的Partition的数量不会随着RDD数量规模的改变而改变），比如join操作的每个partiion仅仅和已知的partition进行join，这个join操作是窄依赖，依赖固定数量的父rdd，因为是确定的partition关系

### 19.Hadoop中，Mapreduce操作的mapper和reducer阶段相当于spark中的哪几个算子？

答：相当于spark中的map算子和reduceByKey算子，当然还是有点区别的,MR会自动进行排序的，spark要看你用的是什么partitioner

### 20.什么是shuffle，以及为什么需要shuffle？

shuffle中文翻译为洗牌，需要shuffle的原因是：某种具有共同特征的数据汇聚到一个计算节点上进行计算

### 21.不需要排序的hashshuffle是否一定比需要排序的sortshuffle速度快？

答：不一定！！当数据规模小，Hashshuffle快于SortedShuffle数据规模大的时候；当数据量大，sortedShuffle会比Hashshuffle快很多，因为数量大的有很多小文件，不均匀，甚至出现数据倾斜，消耗内存大，1.x之前spark使用hash，适合处理中小规模，1.x之后，增加了Sortedshuffle，Spark更能胜任大规模处理了。

### 22.Spark中的HashShufle的有哪些不足？

答：1）shuffle产生海量的小文件在磁盘上，此时会产生大量耗时的、低效的IO操作；

2）.容易导致内存不够用，由于内存需要保存海量的文件操作句柄和临时缓存信息，如果数据处理规模比较大的化，容易出现OOM；

3）容易出现数据倾斜，导致OOM

### 23.conslidate是如何优化Hashshuffle时在map端产生的小文件？

答：1）conslidate为了解决**HashShuffle**同时打开过多文件导致Writerhandler内存使用过大以及产生过多文件导致大量的随机读写带来的低效磁盘IO；

2）conslidate根据CPU的个数来决定每个taskshufflemap端产生多少个文件，假设原来有10个task，100个reduce，每个CPU有10个CPU那么使用hashshuffle会产生10\*100=1000个文件，conslidate产生10\*10=100个文件

备注：conslidate部分减少了文件和文件句柄，并行读很高的情况下（task很多时）还是会很多文件

### 24.Sort-basesdshuffle产生多少个临时文件

答：2\*Map阶段所有的task数量，Mapper阶段中并行的Partition的总数量，其实就是Mapper端task

### 25.Sort-basedshuffle的缺陷?

1)如果mapper中task的数量过大，依旧会产生很多小文件，此时在shuffle传递数据的过程中reducer段，reduce会需要同时大量的记录进行反序列化，导致大量的内存消耗和GC的巨大负担，造成系统缓慢甚至崩溃

2）如果需要在分片内也进行排序，此时需要进行mapper段和reducer段的两次排序

### 26.Sparkshell启动时会启动derby?

**答：**sparkshell启动会启动sparksql，sparksql默认使用derby保存元数据，但是尽量不要用derby，它是单实例，不利于开发。会在本地生成一个文件metastore\_db,如果启动报错，就把那个文件给删了，derby数据库是单实例，不能支持多个用户同时操作，尽量避免使用

### 27.spark.default.parallelism这个参数有什么意义，实际生产中如何设置？

答：1）参数用于设置每个stage的默认task数量。这个参数极为重要，如果不设置可能会直接影响你的Spark作业性能；

2）很多人都不会设置这个参数，会使得集群非常低效，你的cpu，内存再多，如果task始终为1，那也是浪费，spark官网建议task个数为CPU的核数\*executor的个数的2~3倍。

### 28.spark.storage.memoryFraction参数的含义,实际生产中如何调优？

**答：1）**用于设置RDD持久化数据在Executor内存中能占的比例，默认是0.6,，默认Executor60%的内存，可以用来保存持久化的RDD数据。根据你选择的不同的持久化策略，如果内存不够时，可能数据就不会持久化，或者数据会写入磁盘。2）如果持久化操作比较多，可以提高spark.storage.memoryFraction参数，使得更多的持久化数据保存在内存中，提高数据的读取性能，如果shuffle的操作比较多，有很多的数据读写操作到JVM中，那么应该调小一点，节约出更多的内存给JVM，避免过多的JVMgc发生。在webui中观察如果发现gc时间很长，可以设置spark.storage.memoryFraction更小一点。

### 29.spark.shuffle.memoryFraction参数的含义，以及优化经验？

答：1）spark.shuffle.memoryFraction是shuffle调优中重要参数，shuffle从上一个task拉去数据过来，要在Executor进行聚合操作，聚合操作时使用Executor内存的比例由该参数决定，默认是20%

如果聚合时数据超过了该大小，那么就会spill到磁盘，极大降低性能；2）如果Spark作业中的RDD持久化操作较少，shuffle操作较多时，建议降低持久化操作的内存占比，提高shuffle操作的内存占比比例，避免shuffle过程中数据过多时内存不够用，必须溢写到磁盘上，降低了性能。此外，如果发现作业由于频繁的gc导致运行缓慢，意味着task执行用户代码的内存不够用，那么同样建议调低这个参数的值

### 30.介绍一下你对UnifiedMemoryManagement内存管理模型的理解？

答：Spark中的内存使用分为两部分：执行（execution）与存储（storage）。执行内存主要用于shuffles、joins、sorts和aggregations，存储内存则用于缓存或者跨节点的内部数据传输。1.6之前，对于一个Executor,内存都有哪些部分构成：

1）ExecutionMemory。这片内存区域是为了解决shuffles,joins,sortsandaggregations过程中为了避免频繁IO需要的buffer。通过spark.shuffle.memoryFraction(默认0.2)配置。

2）StorageMemory。这片内存区域是为了解决blockcache(就是你显示调用dd.cache,rdd.persist等方法),还有就是broadcasts,以及taskresults的存储。可以通过参数spark.storage.memoryFraction(默认0.6)。设置

3）OtherMemory。给系统预留的，因为程序本身运行也是需要内存的。(默认为0.2).

传统内存管理的不足：

1).Shuffle占用内存0.2\*0.8，内存分配这么少，可能会将数据spill到磁盘，频繁的磁盘IO是很大的负担，Storage内存占用0.6，主要是为了迭代处理。传统的Spark内存分配对操作人的要求非常高。（Shuffle分配内存：ShuffleMemoryManager,TaskMemoryManager,ExecutorMemoryManager）一个Task获得全部的Execution的Memory，其他Task过来就没有内存了，只能等待。

2).默认情况下，Task在线程中可能会占满整个内存，分片数据特别大的情况下就会出现这种情况，其他Task没有内存了，剩下的cores就空闲了，这是巨大的浪费。这也是人为操作的不当造成的。

3).MEMORY\_AND\_DISK\_SER的storage方式，获得RDD的数据是一条条获取，iterator的方式。如果内存不够（spark.storage.unrollFraction），unroll的读取数据过程，就是看内存是否足够，如果足够，就下一条。unroll的space是从Storage的内存空间中获得的。unroll的方式失败，就会直接放磁盘。

4).默认情况下，Task在spill到磁盘之前，会将部分数据存放到内存上，如果获取不到内存，就不会执行。永无止境的等待，消耗CPU和内存。

在此基础上，Spark提出了UnifiedMemoryManager，不再分ExecutionMemory和StorageMemory,实际上还是分的，只不过是ExecutionMemory访问StorageMemory，StorageMemory也可以访问ExecutionMemory，如果内存不够，就会去借。

### 二、参考资料

1.<http://www.tuicool.com/articles/VFnQfaBSpark>如何解决常见的TopN问题

2.https://www.iteblog.com/archives/1522.htmlSpark分区器HashPartitioner和RangePartitioner代码详解

3.http://blog.csdn.net/wisgood/article/details/51436321spark内存管理

4.http://blog.csdn.net/qq403977698/article/details/51086433spark统一内存管理

5.http://blog.csdn.net/high2011/article/details/68491115创建和使用自定义分区

Spark on Yarn面试篇04

### 1.MRV1有哪些不足？

1)可扩展性（对于变化的应付能力）

a)JobTracker内存中保存用户作业的信息

b)JobTracker使用的是粗粒度的锁

2)可靠性和可用性

a)JobTracker失效会多事集群中所有的运行作业，用户需手动重新提交和恢复工作流

3)对不同编程模型的支持

HadoopV1以MapReduce为中心的设计虽然能支持广泛的用例，但是并不适合所有大型计算,如storm，spark

### 2.描述Yarn执行一个任务的过程？

1）客户端client向ResouceManager提交Application，ResouceManager接受Application并根据集群资源状况选取一个node来启动Application的任务调度器driver（ApplicationMaster）

2）ResouceManager找到那个node，命令其该node上的nodeManager来启动一个新的JVM进程运行程序的driver（ApplicationMaster）部分，driver（ApplicationMaster）启动时会首先向ResourceManager注册，说明由自己来负责当前程序的运行

3）driver（ApplicationMaster）开始下载相关jar包等各种资源，基于下载的jar等信息决定向ResourceManager申请具体的资源内容。

4）ResouceManager接受到driver（ApplicationMaster）提出的申请后，会最大化的满足资源分配请求，并发送资源的元数据信息给driver（ApplicationMaster）；

5）driver（ApplicationMaster）收到发过来的资源元数据信息后会根据元数据信息发指令给具体机器上的NodeManager，让其启动具体的container。

6）NodeManager收到driver发来的指令，启动container，container启动后必须向driver（ApplicationMaster）注册。

7）driver（ApplicationMaster）收到container的注册，开始进行任务的调度和计算，直到任务完成。

补充：如果ResourceManager第一次没有能够满足driver（ApplicationMaster）的资源请求，后续发现有空闲的资源，会主动向driver（ApplicationMaster）发送可用资源的元数据信息以提供更多的资源用于当前程序的运行。

### 3.Yarn中的container是由谁负责销毁的，在HadoopMapreduce中container可以复用么？

答：ApplicationMaster负责销毁，在HadoopMapreduce不可以复用，在sparkonyarn程序container可以复用

### 4.提交任务时，如何指定SparkApplication的运行模式？

1）cluster模式：./spark-submit--classxx.xx.xx--masteryarn--deploy-modeclusterxx.jar

2)client模式:./spark-submit--classxx.xx.xx--masteryarn--deploy-modeclientxx.jar

### 5.不启动Spark集群Master和work服务，可不可以运行Spark程序？

答：可以，只要资源管理器第三方管理就可以，如由yarn管理，spark集群不启动也可以使用spark；spark集群启动的是work和master，这个其实就是资源管理框架，yarn中的resourceManager相当于master，NodeManager相当于worker，做计算是Executor，和spark集群的work和manager可以没关系，归根接底还是JVM的运行，只要所在的JVM上安装了spark就可以。

### 6.Spark中的4040端口由什么功能?

答：收集Spark作业运行的信息

### 7.sparkonyarnCluster模式下，ApplicationMaster和driver是在同一个进程么？

答：是,driver位于ApplicationMaster进程中。该进程负责申请资源，还负责监控程序、资源的动态情况。

### 8.如何使用命令查看application运行的日志信息

答：yarnlogs-applicationId<appID>

### 9.SparkonYarn模式有哪些优点？

1)与其他计算框架共享集群资源（eg.Spark框架与MapReduce框架同时运行，如果不用Yarn进行资源分配，MapReduce分到的内存资源会很少，效率低下）；资源按需分配，进而提高集群资源利用等。

2)相较于Spark自带的Standalone模式，Yarn的资源分配更加细致

3)Application部署简化，例如Spark，Storm等多种框架的应用由客户端提交后，由Yarn负责资源的管理和调度，利用Container作为资源隔离的单位，以它为单位去使用内存,cpu等。

4)Yarn通过队列的方式，管理同时运行在Yarn集群中的多个服务，可根据不同类型的应用程序负载情况，调整对应的资源使用量，实现资源弹性管理。

### 10.谈谈你对container的理解？

1）Container作为资源分配和调度的基本单位，其中封装了的资源如内存，CPU，磁盘，网络带宽等。目前yarn仅仅封装内存和CPU

2)Container由ApplicationMaster向ResourceManager申请的，由ResouceManager中的资源调度器异步分配给ApplicationMaster

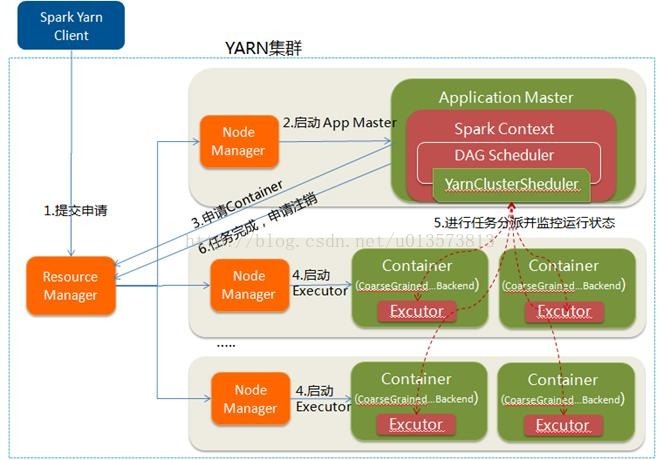
3)Container的运行是由ApplicationMaster向资源所在的NodeManager发起的，Container运行时需提供内部执行的任务命令.

### 11.运行在yarn中Application有几种类型的container？

1）运行ApplicationMaster的Container：这是由ResourceManager（向内部的资源调度器）申请和启动的，用户提交应用程序时，可指定唯一的ApplicationMaster所需的资源；

2）运行各类任务的Container：这是由ApplicationMaster向ResourceManager申请的，并由ApplicationMaster与NodeManager通信以启动之。

### 12.SparkonYarn架构是怎么样的？（要会画哦，这个图）



Yarn提到的AppMaster可以理解为Spark中Standalone模式中的driver。Container中运行着Executor,在Executor中以多线程并行的方式运行Task。运行过程和第二题相似。

### 13.Executor启动时，资源通过哪几个参数指定？

1)num-executors是executor的数量

2)executor-memory是每个executor使用的内存

3)executor-cores是每个executor分配的CPU

### 14.为什么会产生yarn，解决了什么问题，有什么优势?

1)为什么产生yarn，针对MRV1的各种缺陷提出来的资源管理框架

2)解决了什么问题，有什么优势，参考这篇博文：<http://www.aboutyun.com/forum.php?mod=viewthread&tid=6785>

### 15.Mapreduce的执行过程?

阶段1：input/map/partition/sort/spill

阶段2：mapper端merge

阶段3：reducer端merge/reduce/output

详细过程参考这个[http:**//www.cnblogs.com/hipercomer/p/4516581.html**](http://www.cnblogs.com/hipercomer/p/4516581.html)

### 16.一个task的map数量由谁来决定？

一般情况下，在输入源是文件的时候，一个task的map数量由splitSize来决定的，那么splitSize是由以下几个来决定的

goalSize=totalSize/mapred.map.tasks

inSize=max{mapred.min.split.size,minSplitSize}

splitSize=max(minSize,min(goalSize,dfs.block.size))

一个task的reduce数量，由partition决定。

### 17.reduce后输出的数据量有多大？

并不是想知道确切的数据量有多大这个，而是想问你，MR的执行机制，开发完程序，有没有认真评估程序运行效率

1）用于处理redcue任务的资源情况，如果是MRV1的话，分了多少资源给map，多少个reduce,如果是MRV2的话，可以提一下，集群有分了多少内存、CPU给yarn做计算。

2）结合实际应用场景回答，输入数据有多大，大约多少条记录，做了哪些逻辑操作，输出的时候有多少条记录，执行了多久，reduce执行时候的数据有没有倾斜等

3）再提一下，针对mapReduce做了哪几点优化，速度提升了多久，列举1,2个优化点就可以

### 18.你的项目提交到job的时候数据量有多大？

答：1）回答出数据是什么格式，有没有采用什么压缩，采用了压缩的话，压缩比大概是多少；

2）文件大概多大：大概起了多少个map，起了多少个reduce，map阶段读取了多少数据，reduce阶段读取了多少数据，程序大约执行了多久，

3）集群什么规模，集群有多少节点，多少内存，多少CPU核数等。把这些点回答进去，而不是给个数字了事。

### 19.你们提交的job任务大概有多少个？这些job执行完大概用多少时间？

还是考察你开发完程序有没有认真观察过程序的运行，有没有评估程序运行的效率

### 20.你们业务数据量多大？有多少行数据？

这个也是看你们有没有实际的经验,对于没有实战的同学，请把回答的侧重点放在MR的运行机制上面，MR运行效率方面，以及如何优化MR程序（看别人的优化demo，然后在虚拟机上拿demo做一下测试）。

### 22.如何杀死一个正在运行的job

杀死一个job

MRV1：Hadoopjobkilljobid

YARN:yarnapplication-killapplicationId

### 23.列出你所知道的调度器，说明其工作原理

a)Fifoschedular默认的调度器先进先出

b)Capacityschedular计算能力调度器选择占用内存小优先级高的

c)Fairschedular公平调度器所有job占用相同资源

### 24.YarnClient模式下，执行SparkSQL报这个错，Exceptioninthread"Thread-2"java.lang.OutOfMemoryError:PermGenspace，但是在YarnCluster模式下正常运行，可能是什么原因？

1）原因查询过程中调用的是Hive的获取元数据信息、SQL解析，并且使用Cglib等进行序列化反序列化，中间可能产生较多的class文件，导致JVM中的持久代使用较多Cluster模式的持久代默认大小是64M，Client模式的持久代默认大小是32M，而Driver端进行SQL处理时，其持久代的使用可能会达到90M，导致OOM溢出，任务失败。

yarn-cluster模式下出现，yarn-client模式运行时倒是正常的，原来在$SPARK\_HOME/bin/spark-class文件中已经设置了持久代大小：

JAVA\_OPTS="-XX:MaxPermSize=256m$OUR\_JAVA\_OPTS"

2）解决方法:在Spark的conf目录中的spark-defaults.conf里，增加对Driver的JVM配置，因为Driver才负责SQL的解析和元数据获取。配置如下：

spark.driver.extraJavaOptions-XX:PermSize=128M-XX:MaxPermSize=256M

### 25.spark.driver.extraJavaOptions这个参数是什么意思，你们生产环境配了多少？

传递给executors的JVM选项字符串。例如GC设置或者其它日志设置。注意，在这个选项中设置Spark属性或者堆大小是不合法的。Spark属性需要用SparkConf对象或者spark-submit脚本用到的spark-defaults.conf文件设置。堆内存可以通过spark.executor.memory设置

### 26.导致Executor产生FULLgc的原因，可能导致什么问题？

答：可能导致Executor僵死问题，海量数据的shuffle和数据倾斜等都可能导致fullgc。以shuffle为例，伴随着大量的Shuffle写操作，JVM的新生代不断GC，EdenSpace写满了就往SurvivorSpace写，同时超过一定大小的数据会直接写到老生代，当新生代写满了之后，也会把老的数据搞到老生代，如果老生代空间不足了，就触发FULLGC，还是空间不够，那就OOM错误了，此时线程被Blocked，导致整个Executor处理数据的进程被卡住

### 27.Combiner和partition的作用

combine分为map端和reduce端，作用是把同一个key的键值对合并在一起，可以自定义的。combine函数把一个map函数产生的<key,value>对（多个key,value）合并成一个新<key2,value2>.将新的<key2,value2>作为输入到reduce函数中这个value2亦可称之为values，因为有多个。这个合并的目的是为了减少网络传输。partition是分割map每个节点的结果，按照key分别映射给不同的reduce，也是可以自定义的。这里其实可以理解归类。我们对于错综复杂的数据归类。比如在动物园里有牛羊鸡鸭鹅，他们都是混在一起的，但是到了晚上他们就各自牛回牛棚，羊回羊圈，鸡回鸡窝。partition的作用就是把这些数据归类。只不过在写程序的时候，mapreduce使用哈希HashPartitioner帮我们归类了。这个我们也可以自定义。shuffle就是map和reduce之间的过程，包含了两端的combine和partition。Map的结果，会通过partition分发到Reducer上，Reducer做完Reduce操作后，通OutputFormat，进行输出shuffle阶段的主要函数是fetchOutputs(),这个函数的功能就是将map阶段的输出，copy到reduce节点本地

### 28.Spark执行任务时出现java.lang.OutOfMemoryError:GCoverheadlimitexceeded和java.lang.OutOfMemoryError:javaheapspace原因和解决方法？

答：原因：加载了太多资源到内存，本地的性能也不好，gc时间消耗的较多

解决方法：

1）增加参数，-XX:-UseGCOverheadLimit，关闭这个特性，同时增加heap大小，-Xmx1024m

2）下面这个两个参数调大点

exportSPARK\_EXECUTOR\_MEMORY=6000M

exportSPARK\_DRIVER\_MEMORY=7000M

可以参考这个：<http://www.cnblogs.com/hucn/p/3572384.html>

### 29.请列出在你以前工作中所使用过的开发map/reduce的语言

答：java，Scala，Python，shell

### 30.你认为/etc/hosts配置错误，会对集群有什么影响？

答：1）直接导致域名没法解析，主节点与子节点，子节点与子节点没法正常通讯，

2）间接导致配置错误的相关节点删的服务不正常，甚至没法启动，job执行失败等

### 二、参考资料

**1.**http://www.cnblogs.com/hucn/p/3572384.html

2.http:**//www.cnblogs.com/hipercomer/p/4516581.html**

**3.**http://www.aboutyun.com/forum.php?mod=viewthread&tid=6785

SparkCore面试篇05

### 1.scala中private与private[this]修饰符的区别？

### 2.scala中内部类和java中的内部类区别

### 3.Spark中standalone模式特点，有哪些优点和缺点？

### 4.FIFO调度模式的基本原理、优点和缺点？

### 5.FAIR调度模式的优点和缺点？

### 6.CAPCACITY调度模式的优点和缺点？

### 7.列举你了解的序列化方法，并谈谈序列化有什么好处？

### 8.常见的数压缩方式，你们生产集群采用了什么压缩方式，提升了多少效率？

### 9.简要描述Spark写数据的流程？

### 10.Spark中Lineage的基本原理

### 11.使用shell和scala代码实现WordCount？

### 12.请列举你碰到的CPU密集型的应用场景，你有做哪些优化？

### 13.SparkRDD和MR2的区别

### 14.Spark读取hdfs上的文件，然后count有多少行的操作，你可以说说过程吗。那这个count是在内存中，还是磁盘中计算的呢？

### 15.spark和Mapreduce快？为什么快呢？快在哪里呢？

### 16.sparkSql又为什么比hive快呢？

### 17.RDD的数据结构是怎么样的？

### 18.RDD算子里操作一个外部map比如往里面put数据。然后算子外再遍历map。会有什么问题吗。

### 19.hadoop的生态呢。说说你的认识。

### 20.jvm怎么调优的，介绍你的SparkJVM调优经验？

### 21.jvm结构？堆里面几个区？

### 22.怎么用spark做数据清洗

### 23.spark怎么整合hive？

### 24.spark读取数据，是几个Partition呢？

### 25.hbaseregion多大会分区，spark读取hbase数据是如何划分partition的？

### 26.画图，画Spark的工作模式，部署分布架构图

### 27.画图，画图讲解spark工作流程。以及在集群上和各个角色的对应关系。

### 28.java自带有哪几种线程池。

### 29.画图，讲讲shuffle的过程。那你怎么在编程的时候注意避免这些性能问题？

### 30.BlockManager怎么管理硬盘和内存的？

SparkCore面试篇06

### 1.kafka收集数据的原理？

### 2.讲讲列式存储的parquet文件底层格式？

### 3.dataset和dataframe？

### 4scala中trait特征和用法？

### 5.redis和memcache的区别？

### 6.列举Spark中常见的端口，分别有什么功能？

### 7.Sparkmaster如何通过Zookeeper做HA？

### 8.Spark官网中，你常用哪几个模块？

### 9.你有见过哪些原因导致的数据倾斜，怎么解决？

### 10.简要描述宽依赖窄依赖以及各自的特点？

### 11.yarn的原理？

### 12.BlockManager怎么管理硬盘和内存的？

### 13.哪些算子操作涉及到shuffle1

### 14.看过源码？你熟悉哪几个部分的源码？

### 15.集群上nodemanager和ResourceManager的数量关系？

### 16.Spark如何处理结构化数据，Spark如何处理非结构话数据？

### 17.Spark性能优化主要有哪些手段？

### 18.简要描述Spark分布式集群搭建的步骤？

### 19.对于Spark你觉得他对于现有大数据的现状的优势和劣势在哪里？

### 20.对于算法是否进行过自主的研究设计？

### 21.简要描述你了解的一些数据挖掘算法与内容

### 22.什么时候join不发生shuffle？

### 23.sparkshuffle的具体过程，你知道几种shuffle方式

### 24.spark如何防止内存溢出？

### 25.简述hadoop实现join的及各种方式？

### 26rdd转为dataFrame两种方式？

### 27.列举你熟悉的内存系统，各自的优缺点？

### 28.Spark中Master实现HA有哪些方式？

### 29函数式编程特点？

### 30.Sort-basedshuffle的缺陷？

-------------------------

SparkCore面试篇07

### 1、简答说一下hadoop的map-reduce编程模型

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合

使用的是hadoop内置的数据类型，比如longwritable、text等

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

### 2、hadoop的TextInputFormat作用是什么，如何自定义实现

InputFormat会在map操作之前对数据进行两方面的预处理

1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次

2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法

在createRecordReader中可以自定义分隔符

### 3、hadoop和spark的都是并行计算，那么他们有什么相同和区别

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

### 4、为什么要用flume导入hdfs，hdfs的构架是怎样的

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死

### 5、map-reduce程序运行的时候会有什么比较常见的问题

比如说作业中大部分都完成了，但是总有几个reduce一直在运行

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作

### 6、简单说一下hadoop和spark的shuffle过程

hadoop：map端保存分片数据，通过网络收集到reduce端

spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor，减少shuffle可以提高性能

### 7、Hive中存放是什么？

表（数据+元数据）。存的是和hdfs的映射关系，hive是逻辑上的数据仓库，实际操作的都是hdfs上的文件，HQL就是用sql语法来写的mr程序。

### 8、Hive与关系型数据库的关系？

没有关系，hive是数据仓库，不能和数据库一样进行实时的CURD操作。

是一次写入多次读取的操作，可以看成是ETL工具。

### 9、Flume工作机制是什么？

核心概念是agent，里面包括source、chanel和sink三个组件。

source运行在日志收集节点进行日志采集，之后临时存储在chanel中，sink负责将chanel中的数据发送到目的地。

只有成功发送之后chanel中的数据才会被删除。

首先书写flume配置文件，定义agent、source、chanel和sink然后将其组装，执行flume-ng命令。

### 10、Sqoop工作原理是什么？

hadoop生态圈上的数据传输工具。

可以将关系型数据库的数据导入非结构化的hdfs、hive或者bbase中，也可以将hdfs中的数据导出到关系型数据库或者文本文件中。

使用的是mr程序来执行任务，使用jdbc和关系型数据库进行交互。

import原理：通过指定的分隔符进行数据切分，将分片传入各个map中，在map任务中在每行数据进行写入处理没有reduce。

export原理：根据要操作的表名生成一个java类，并读取其元数据信息和分隔符对非结构化的数据进行匹配，多个map作业同时执行写入关系型数据库

### 11、Hbase行健列族的概念，物理模型，表的设计原则？

行健：是hbase表自带的，每个行健对应一条数据。

列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中的数据可以有很多，通过时间戳来区分。

物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。

rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。

列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

### 12、SparkStreaming和Storm有何区别？

一个实时毫秒一个准实时亚秒，不过storm的吞吐率比较低。

### 13、mllib支持的算法？

大体分为四大类，分类、聚类、回归、协同过滤。

### 14、简答说一下hadoop的map-reduce编程模型？

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合。

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出。

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则。

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出。

在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则。

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量。

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job。

### 15、Hadoop平台集群配置、环境变量设置？

zookeeper：修改zoo.cfg文件，配置dataDir，和各个zk节点的server地址端口，tickTime心跳时间默认是2000ms，其他超时的时间都是以这个为基础的整数倍，之后再dataDir对应目录下写入myid文件和zoo.cfg中的server相对应。

hadoop：修改

hadoop-env.sh配置java环境变量

core-site.xml配置zk地址，临时目录等

hdfs-site.xml配置nn信息，rpc和http通信地址，nn自动切换、zk连接超时时间等

yarn-site.xml配置resourcemanager地址

mapred-site.xml配置使用yarn

slaves配置节点信息

格式化nn和zk。

hbase：修改

hbase-env.sh配置java环境变量和是否使用自带的zk

hbase-site.xml配置hdfs上数据存放路径，zk地址和通讯超时时间、master节点

regionservers配置各个region节点

zoo.cfg拷贝到conf目录下

spark：

安装Scala

修改spark-env.sh配置环境变量和master和worker节点配置信息

环境变量的设置：直接在/etc/profile中配置安装的路径即可，或者在当前用户的宿主目录下，配置在.bashrc文件中，该文件不用source重新打开shell窗口即可，配置在.bash\_profile的话只对当前用户有效。

### 16、Hadoop性能调优？

调优可以通过系统配置、程序编写和作业调度算法来进行。

hdfs的block.size可以调到128/256（网络很好的情况下，默认为64）

调优的大头：mapred.map.tasks、mapred.reduce.tasks设置mr任务数（默认都是1）

mapred.tasktracker.map.tasks.maximum每台机器上的最大map任务数

mapred.tasktracker.reduce.tasks.maximum每台机器上的最大reduce任务数

mapred.reduce.slowstart.completed.maps配置reduce任务在map任务完成到百分之几的时候开始进入

这个几个参数要看实际节点的情况进行配置，reduce任务是在33%的时候完成copy，要在这之前完成map任务，（map可以提前完成）

mapred.compress.map.output,mapred.output.compress配置压缩项，消耗cpu提升网络和磁盘io

合理利用combiner

注意重用writable对象

### 17、Hadoop高并发？

首先肯定要保证集群的高可靠性，在高并发的情况下不会挂掉，支撑不住可以通过横向扩展。

datanode挂掉了使用hadoop脚本重新启动。

### 25、kafka工作原理？

producer向broker发送事件，consumer从broker消费事件。

事件由topic区分开，每个consumer都会属于一个group。

相同group中的consumer不能重复消费事件，而同一事件将会发送给每个不同group的consumer。

### 26、ALS算法原理？

答：对于user-product-rating数据，als会建立一个稀疏的评分矩阵，其目的就是通过一定的规则填满这个稀疏矩阵。

als会对稀疏矩阵进行分解，分为用户-特征值，产品-特征值，一个用户对一个产品的评分可以由这两个矩阵相乘得到。

通过固定一个未知的特征值，计算另外一个特征值，然后交替反复进行最小二乘法，直至差平方和最小，即可得想要的矩阵。

### 27、kmeans算法原理？

随机初始化中心点范围，计算各个类别的平均值得到新的中心点。

重新计算各个点到中心值的距离划分，再次计算平均值得到新的中心点，直至各个类别数据平均值无变化。

### 28、canopy算法原理？

根据两个阈值来划分数据，以随机的一个数据点作为canopy中心。

计算其他数据点到其的距离，划入t1、t2中，划入t2的从数据集中删除，划入t1的其他数据点继续计算，直至数据集中无数据。

### 29、朴素贝叶斯分类算法原理？

对于待分类的数据和分类项，根据待分类数据的各个特征属性，出现在各个分类项中的概率判断该数据是属于哪个类别的。

### 30、关联规则挖掘算法apriori原理？

一个频繁项集的子集也是频繁项集，针对数据得出每个产品的支持数列表，过滤支持数小于预设值的项，对剩下的项进行全排列，重新计算支持数，再次过滤，重复至全排列结束，可得到频繁项和对应的支持数。

### 参考资料

1. http://lib.csdn.net/mobile/article/hadoop/6445

Spark错误日志面试集锦-面试篇08

### 1、OperationcategoryREADisnotsupportedinstatestandby是什么原因导致的？org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.ipc.StandbyException):OperationcategoryREADisnotsupportedinstatestandby答：此时请登录Hadoop的管理界面查看运行节点是否处于standby

如登录地址是：http://xx.xx.xx.xx:50070/dfshealth.html#tab-overview

如果是，则不可在处于StandBy机器运行spark计算，因为该台机器为备分机器

### 2、不配置spark.deploy.recoveryMode选项为ZOOKEEPER，会有什么不好的地方

如果不设置spark.deploy.recoveryMode的话，那么集群的所有运行数据在Master重启是都会丢失，可参考BlackHolePersistenceEngine的实现。

### 3、多Master如何配置

因为涉及到多个Master，所以对于应用程序的提交就有了一点变化，因为应用程序需要知道当前的Master的IP地址和端口。这种HA方案处理这种情况很简单，只需要在SparkContext指向一个Master列表就可以了，如spark://host1:port1,host2:port2,host3:port3，应用程序会轮询列表。

### 4、NoSpaceLeftonthedevice（Shuffle临时文件过多）

由于Spark在计算的时候会将中间结果存储到/tmp目录，而目前linux又都支持tmpfs，其实就是将/tmp目录挂载到内存当中。

那么这里就存在一个问题，中间结果过多导致/tmp目录写满而出现如下错误

NoSpaceLeftonthedevice

解决办法

第一种：修改配置文件spark-env.sh,把临时文件引入到一个自定义的目录中去即可

exportSPARK\_LOCAL\_DIRS=/home/utoken/datadir/spark/tmp

第二种：偷懒方式，针对tmp目录不启用tmpfs,直接修改/etc/fstab

### 5、java.lang.OutOfMemory,unabletocreatenewnativethread

**Causedby:java.lang.OutOfMemoryError:unabletocreatenewnativethread**

**atjava.lang.Thread.start0(NativeMethod)**

**atjava.lang.Thread.start(Thread.java:640)**

上面这段错误提示的本质是Linux操作系统无法创建更多进程，导致出错，并不是系统的内存不足。因此要解决这个问题需要修改Linux允许创建更多的进程，就需要修改Linux最大进程数。

[utoken@nn1~]$ulimit-a

临时修改允许打开的最大进程数

[utoken@nn1~]$ulimit-u65535

临时修改允许打开的文件句柄

[utoken@nn1~]$ulimit-n65535

永久修改Linux最大进程数量

[utoken@nn1~]$vim/etc/security/limits.d/90-nproc.conf

* softnproc60000

rootsoftnprocunlimited

永久修改用户打开文件的最大句柄数，该值默认1024，一般都会不够，常见错误就是notopenfile

[utoken@nn1~]$vim/etc/security/limits.conf

bdatasoftnofile65536

bdatahardnofile65536

### 6、Worker节点中的work目录占用许多磁盘空间

目录地址：/home/utoken/software/spark-1.3.0-bin-hadoop2.4/work

这些是Driver上传到worker的文件，需要定时做手工清理，否则会占用许多磁盘空间

### 7、spark-shell提交SparkApplication如何解决依赖库

spark-shell的话，利用–driver-class-path选项来指定所依赖的jar文件，注意的是–driver-class-path后如果需要跟着多个jar文件的话，jar文件之间使用冒号(:)来分割。

### 8、Spark在发布应用的时候，出现连接不上master问题，如下

**15/11/1911:35:50INFOAppClient$ClientEndpoint:Connectingtomasterspark://s1:7077…**

**15/11/1911:35:50WARNReliableDeliverySupervisor:Associationwithremotesystem[akka.tcp://sparkMaster@s1:7077]hasfailed,addressisnowgatedfor[5000]ms.Reason:[Disassociated]**

解决方式

检查所有机器时间是否一致、hosts是否都配置了映射、客户端和服务器端的Scala版本是否一致、Scala版本是否和Spark兼容

检查是否兼容问题请参考官方网站介绍：

### 9、开发spark应用程序（和Flume-NG结合时）发布应用时可能出现org.jboss.netty.channel.ChannelException:Failedtobindto:/192.168.10.156:18800

15/11/2710:33:44ERRORReceiverSupervisorImpl:Stoppedreceiverwitherror:org.jboss.netty.channel.ChannelException:Failedtobindto:/192.168.10.156:18800

15/11/2710:33:44ERRORExecutor:Exceptionintask0.0instage2.0(TID70)

org.jboss.netty.channel.ChannelException:Failedtobindto:/192.168.10.156:18800

atorg.jboss.netty.bootstrap.ServerBootstrap.bind(ServerBootstrap.java:272)

Causedby:java.net.BindException:Cannotassignrequestedaddress

由于spark通过Master发布的时候，会自动选取发送到某一台的worker节点上，所以这里绑定端口的时候，需要选择相应的worker服务器，但是由于我们无法事先了解到，spark发布到哪一台服务器的，所以这里启动报错，是因为在192.168.10.156:18800的机器上面没有启动Driver程序，而是发布到了其他服务器去启动了，所以无法监听到该机器出现问题，所以我们需要设置spark分发包时，发布到所有worker节点机器，或者发布后，我们去寻找发布到了哪一台机器，重新修改绑定IP，重新发布，有一定几率发布成功。详情可见《印象笔记-战5渣系列——SparkStreaming启动问题-推酷》

### 10、spark-shell找不到hadoopso问题解决

[main]WARNorg.apache.hadoop.util.NativeCodeLoader-Unabletoloadnative-hadooplibraryforyourplatform…usingbuiltin-javaclasseswhereapplicable

在Spark的conf目录下，修改spark-env.sh文件，加入LD\_LIBRARY\_PATH环境变量，值为HADOOP的native库路径即可.

**11、ERRORXSDB6:AnotherinstanceofDerbymayhavealreadybootedthedatabase/home/bdata/data/metastore\_db.**

在使用HiveonSpark模式操作hive里面的数据时，报以上错误，原因是因为HIVE采用了derby这个内嵌数据库作为数据库，它不支持多用户同时访问,解决办法就是把derby数据库换成mysql数据库即可

变更方式

### 12、java.lang.IllegalArgumentException:java.net.UnknownHostException:dfscluster

解决办法：

找不到hdfs集群名字dfscluster,这个文件在HADOOP的etc/hadoop下面，有个文件hdfs-site.xml，复制到Spark的conf下，重启即可

如：执行脚本，分发到所有的Spark集群机器中，

[bdata@bdata4hadoop]foriin34,35,36,37,38;doscphdfs−site.xml192.168.10.i:/u01/spark-1.5.1/conf/;done

### 13、Exceptioninthread“main”java.lang.Exception:Whenrunningwithmaster‘yarn-client’eitherHADOOP\_CONF\_DIRorYARN\_CONF\_DIRmustbesetintheenvironment.

问题：在执行yarn集群或者客户端时，报以上错误，

[bdata@bdata4bin]$./spark-sql–masteryarn-client

Exceptioninthread“main”java.lang.Exception:Whenrunningwithmaster‘yarn-client’eitherHADOOP\_CONF\_DIRorYARN\_CONF\_DIRmustbesetintheenvironment.

解决办法

根据提示，配置HADOOP\_CONF\_DIRorYARN\_CONF\_DIR的环境变量即可

exportHADOOP\_HOME=/u01/hadoop-2.6.1

exportHADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

PATH=PATH:HOME/.local/bin:HOME/bin:SQOOP\_HOME/bin:HIVEHOME/bin:HADOOP\_HOME/bin

### 14、Jobabortedduetostagefailure:Task3instage0.0failed4times,mostrecentfailure:Losttask3.3in

[Stage0:>(0+4)/42]2016-01-1511:28:16,512[org.apache.spark.scheduler.TaskSchedulerImpl]-[ERROR]Lostexecutor0on192.168.10.38:remoteRpcclientdisassociated

[Stage0:>(0+4)/42]2016-01-1511:28:23,188[org.apache.spark.scheduler.TaskSchedulerImpl]-[ERROR]Lostexecutor1on192.168.10.38:remoteRpcclientdisassociated

[Stage0:>(0+4)/42]2016-01-1511:28:29,203[org.apache.spark.scheduler.TaskSchedulerImpl]-[ERROR]Lostexecutor2on192.168.10.38:remoteRpcclientdisassociated

[Stage0:>(0+4)/42]2016-01-1511:28:36,319[org.apache.spark.scheduler.TaskSchedulerImpl]-[ERROR]Lostexecutor3on192.168.10.38:remoteRpcclientdisassociated

2016-01-1511:28:36,321[org.apache.spark.scheduler.TaskSetManager]-[ERROR]Task3instage0.0failed4times;abortingjob

Exceptioninthread“main”org.apache.spark.SparkException:Jobabortedduetostagefailure:Task3instage0.0failed4times,mostrecentfailure:Losttask3.3instage0.0(TID14,192.168.10.38):ExecutorLostFailure(executor3lost)

Driverstacktrace:

atorg.apache.spark.scheduler.DAGScheduler.org$apache$spark$scheduler$DAGScheduler$$failJobAndIndependentStages(DAGScheduler.scala:1283)

解决方案

这里遇到的问题主要是因为数据源数据量过大，而机器的内存无法满足需求，导致长时间执行超时断开的情况，数据无法有效进行交互计算，因此有必要增加内存

### 15、长时间等待无反应，并且看到服务器上面的web界面有内存和核心数，但是没有分配，如下图

[Stage0:>(0+0)/42]

或者日志信息显示：

16/01/1514:18:56WARNTaskSchedulerImpl:Initialjobhasnotacceptedanyresources;checkyourclusterUItoensurethatworkersareregisteredandhavesufficientresources

解决方案

出现上面的问题主要原因是因为我们通过参数spark.executor.memory设置的内存过大，已经超过了实际机器拥有的内存，故无法执行，需要等待机器拥有足够的内存后，才能执行任务，可以减少任务执行内存，设置小一些即可

### 16、内存不足或数据倾斜导致ExecutorLost（spark-submit提交）

TaskSetManager:Losttask1.0instage6.0(TID100,192.168.10.37):java.lang.OutOfMemoryError:Javaheapspace

16/01/1514:29:51INFOBlockManagerInfo:Addedbroadcast\_8\_piece0inmemoryon192.168.10.37:57139(size:42.0KB,free:24.2MB)

16/01/1514:29:53INFOBlockManagerInfo:Addedbroadcast\_8\_piece0inmemoryon192.168.10.38:53816(size:42.0KB,free:24.2MB)

16/01/1514:29:55INFOTaskSetManager:Startingtask3.0instage6.0(TID102,192.168.10.37,ANY,2152bytes)

16/01/1514:29:55WARNTaskSetManager:Losttask1.0instage6.0(TID100,192.168.10.37):java.lang.OutOfMemoryError:Javaheapspace

atjava.io.BufferedOutputStream.(BufferedOutputStream.java:76)

atjava.io.BufferedOutputStream.(BufferedOutputStream.java:59)

…….

org.apache.spark.SparkException:Jobabortedduetostagefailure:Task0instage6.0failed4times,mostrecentfailure:Losttask0.3instage6.0(TID142,192.168.10.36):ExecutorLostFailure(executor4lost)

……

WARNTaskSetManager:Losttask4.1instage6.0(TID137,192.168.10.38):java.lang.OutOfMemoryError:GCoverheadlimitexceeded

解决办法：

由于我们在执行Spark任务是，读取所需要的原数据，数据量太大，导致在Worker上面分配的任务执行数据时所需要的内存不够，直接导致内存溢出了，所以我们有必要增加Worker上面的内存来满足程序运行需要。

在SparkStreaming或者其他spark任务中，会遇到在Spark中常见的问题，典型如ExecutorLost相关的问题(shufflefetch失败，Task失败重试等)。这就意味着发生了内存不足或者数据倾斜的问题。这个目前需要考虑如下几个点以获得解决方案：

A、相同资源下，增加partition数可以减少内存问题。原因如下：通过增加partition数，每个task要处理的数据少了，同一时间内，所有正在运行的task要处理的数量少了很多，所有Executor占用的内存也变小了。这可以缓解数据倾斜以及内存不足的压力。

B、关注shuffleread阶段的并行数。例如reduce,group之类的函数，其实他们都有第二个参数，并行度(partition数)，只是大家一般都不设置。不过出了问题再设置一下，也不错。

C、给一个Executor核数设置的太多，也就意味着同一时刻，在该Executor的内存压力会更大，GC也会更频繁。我一般会控制在3个左右。然后通过提高Executor数量来保持资源的总量不变。

### 17、SparkStreaming和kafka整合后读取消息报错：OffsetOutOfRangeException

解决方案：如果和kafka消息中间件结合使用，请检查消息体是否大于默认设置1m，如果大于，则需要设置fetch.message.max.bytes=1m，这里需要把值设置大些

### 18、java.io.IOException:Couldnotlocateexecutablenull\bin\winutils.exeintheHadoopbinaries.（sparksqlonhive任务引发HiveContextNullPointerException）

解决办法

在开发hive和Spark整合的时候，如果是Windows系统，并且没有配置HADOOP\_HOME的环境变量，那么可能找不到winutils.exe这个工具，由于使用hive时，对该命令有依赖，所以不要忽视该错误，否则将无法创建HiveContext，一直报Exceptioninthread“main”java.lang.RuntimeException:java.lang.NullPointerException

因此，解决该办法有两个方式

A、把任务打包成jar，上传到服务器上面，服务器是配置过HADOOP\_HOME环境变量的，并且不需要依赖winutils,所以只需要通过spark-submit方式提交即可，如：

[bdata@bdata4app]$spark-submit–classcom.pride.hive.HiveOnSparkTest–masterspark://bdata4:7077spark-simple-1.0.jar

B、解决winutils.exe命令不可用问题，配置Windows上面HADOOP\_HOME的环境变量，或者在程序最开始的地方设置HADOOP\_HOME的属性配置,这里需要注意，由于最新版本已经没有winutils这些exe命令了，我们需要在其他地方下载该命令放入HADOOP的bin目录下，当然也可以直接配置下载项目的环境变量，变量名一定要是HADOOP\_HOME才行

下载地址：https://github.com/srccodes/hadoop-common-2.2.0-bin/archive/master.zip（记得翻墙哦）

任何项目都生效，需要配置Windows的环境变量，如果只在程序中生效可在程序中配置即可，如

//用于解决Windows下找不到winutils.exe命令

System.setProperty(“hadoop.home.dir”,“E:\Software\hadoop-common-2.2.0-bin”);

### 19、Therootscratchdir:/tmp/hiveonHDFSshouldbewritable.Currentpermissionsare:rwx——

解决办法

1、程序中设置环境变量：System.setProperty(“HADOOP\_USER\_NAME”,“bdata”)

2、修改HDFS的目录权限

Updatethepermissionofyour/tmp/hiveHDFSdirectoryusingthefollowingcommand

hadoopdfs-chmod777/tmp/hive

此问题暂未解决，估计是17点解决winutils有问题，建议最好把任务程序发布到服务器上面解决

### 20、Exceptioninthread“main”org.apache.hadoop.security.AccessControlException:Permissiondenied:user=Administrator,access=WRITE,inode=”/data”:bdata:supergroup:drwxr-xr-x

解决办法

1、在系统的环境变量或javaJVM变量里面添加HADOOP\_USER\_NAME，如程序中添加System.setProperty(“HADOOP\_USER\_NAME”,“bdata”);，这里的值就是以后会运行HADOOP上的Linux的用户名，如果是eclipse，则修改完重启eclipse，不然可能不生效

2、hdfsdfs-chmod777修改相应权限地址

### 21、运行Spark-SQL报错：org.apache.spark.sql.AnalysisException:unresolvedoperator‘Project

解决办法：

在Spark-sql和hive结合时或者单独Spark-sql，运行某些sql语句时，偶尔出现上面错误，那么我们可以检查一下sql的问题，这里遇到的问题是嵌套语句太多，导致spark无法解析，所以需要修改sql或者改用其他方式处理；特别注意该语句可能在hive里面没有错误，spark才会出现的一种错误。

### 22.在$SPARK\_HOME/conf/spark-env.sh中设置这些变量好像也只是在terminal中的shell环境中才有效JAVA\_HOMEisnotsetException:Javagatewayprocessexitedbeforesendingthedriveritsportnumber

但是在命令行中是有的

pipi@pipicmp:~$echo$JAVA\_HOME

/home/pipi/ENV/jdk

解决方法1：在py代码中加入JAVA\_HOME到os中

JAVA\_HOME=/home/pipi/ENV/jdk

os.environ['JAVA\_HOME']=conf.get(SECTION,'JAVA\_HOME')

解决方法2：或者在hadoop中配置好JAVA\_HOME

hadoop中配置JAVA\_HOME

### 23.ValueError:CannotrunmultipleSparkContextsatonce

Welcometo

\_\_\_\_\_\_

/\_\_/\_\_\_\_\_\_\_\_\_\_//\_\_

\_\\/\_\/\_`/\_\_/'\_/

/\_\_/.\_\_/\\_,\_/\_//\_/\\_\version2.0.1

/\_/

UsingPythonversion3.5.2(default,Sep10201608:21:44)

SparkSessionavailableas'spark'.

ValueError:CannotrunmultipleSparkContextsatonce;existingSparkContext(app=pyspark-shell,master=local[\*])createdby<module>at<frozenimportlib.\_bootstrap>:222

原因是：frompyspark.shellimportsqlContext

引入的包中也定义了一个sc=spark.sparkContext导致和本代码中定义重复了。

### 24.spark输出太多warningmessages

调试log时候发现问题解决了

在简略Spark输出设置时[Spark安装和配置]修改过$SPARK\_HOME/conf/log4j.properties.template文件只输出WARN信息，就算改成了ERROR，信息也还是会自动修改成WARN输出出来，不过多了一条提示：

Settingdefaultloglevelto"WARN".Toadjustlogginglevelusesc.setLogLevel(newLevel).

就在这时发现了一个解决方案：

根据提示在代码中加入sc.setLogLevel(**'ERROR'**)就可以解决了！

### 25.org.apache.spark.shuffle.FetchFailedException，一般发生在有大量shuffle操作的时候,task不断的failed,然后又重执行，一直循环下去，非常的耗时

一般遇到这种问题提高executor内存即可,同时增加每个executor的cpu,这样不会减少task并行度。

### 26.Executor&TaskLost因为网络或者gc的原因,worker或executor没有接收到executor或task的心跳反馈WARNTaskSetManager:Losttask1.0instage0.0(TID1,aa.local):ExecutorLostFailure(executorlost)

提高spark.network.timeout的值，根据情况改成300(5min)或更高。

默认为120(120s),配置所有网络传输的延时，如果没有主动设置以下参数，默认覆盖其属性

* spark.core.connection.ack.wait.timeout
* spark.akka.timeout
* spark.storage.blockManagerSlaveTimeoutMs
* spark.shuffle.io.connectionTimeout
* spark.rpc.askTimeoutorspark.rpc.lookupTimeout

### 27.Master挂掉,standby重启也失效，如Master默认使用512M内存，当集群中运行的任务特别多时，就会挂掉，原因是master会读取每个task的eventlog日志去生成sparkui，内存不足自然会OOM，可以在master的运行日志中看到，通过HA启动的master自然也会因为这个原因失败。

1）.增加Master的内存占用，在Master节点spark-env.sh中设置：

exportSPARK\_DAEMON\_MEMORY10g#根据你的实际情况

2）.减少保存在Master内存中的作业信息

spark.ui.retainedJobs500#默认都是1000spark.ui.retainedStages500

### 28.worker挂掉或假死有时候我们还会在webui中看到worker节点消失或处于dead状态，在该节点运行的任务则会报各种lostworker的错误，引发原因和上述大体相同，worker内存中保存了大量的ui信息导致gc时失去和master之间的心跳。

**解决**

1）增加Master的内存占用，在Worker节点spark-env.sh中设置：

exportSPARK\_DAEMON\_MEMORY2g#根据你的实际情况

2）减少保存在Worker内存中的Driver,Executor信息

spark.worker.ui.retainedExecutors200#默认都是1000spark.worker.ui.retainedDrivers200

### 29.报错：ERRORstorage.DiskBlockObjectWriter:Uncaughtexceptionwhilerevertingpartialwritestofile/hadoop/application\_1415632483774\_448143/spark-local-20141127115224-9ca8/04/shuffle\_1\_1562\_27

**java.io.FileNotFoundException:/hadoop/application\_1415632483774\_448143/spark-local-20141127115224-9ca8/04/shuffle\_1\_1562\_27(Nosuchfileordirectory)**

表面上看是因为shuffle没有地方写了，如果后面的stack是localspace的问题，那么清一下磁盘就好了。上面这种问题，是因为一个excutor给分配的内存不够，此时，减少excutor-core的数量，加大excutor-memory的值应该就没有问题。

### 30.报错：ERRORexecutor.CoarseGrainedExecutorBackend:DriverDisassociated[akka.tcp://sparkExecutor@pc-jfqdfx31:48586]->[akka.tcp://sparkDriver@pc-jfqdfx30:41656]disassociated!Shuttingdown.

**15/07/2310:50:56ERRORexecutor.CoarseGrainedExecutorBackend:RECEIVEDSIGNAL15:SIGTERM**

这个错误比较隐晦，从信息上看来不知道是什么问题，但是归根结底还是内存的问题，有两个方法可以解决这个错误，一是，如上面所说，加大excutor-memory的值，减少executor-cores的数量，问题可以解决。二是，加大executor.overhead的值，但是这样其实并没有解决掉根本的问题。所以如果集群的资源是支持的话，就用1的办法吧。

另外，这个错误也出现在partitionBy(newHashPartition(partiton-num))时，如果partiton-num太大或者太小的时候会报这种错误，说白了也是内存的原因，不过这个时候增加内存和overhead没有什么用，得去调整这个partiton-num的值。

## 某公司笔试题

### **1、如何为一个hadoop任务设置mappers的数量**

答案：

使用job.setNumMapTask(intn)手动分割，这是不靠谱的

官方文档：“Note:Thisisonlyahinttotheframework”说明这个方法只是提示作用，不起决定性作用

实际上要用公式计算：

Max（min.split，min（max.split，block））就设置分片的最大最下值computeSplitSize（）设置

参考：http://blog.csdn.net/strongerbit/article/details/7440111

### **2、有可能使hadoop任务输出到多个目录中么？如果可以，怎么做？**

答案：在1.X版本后使用MultipleOutputs.java类实现

源码：

MultipleOutputs.addNamedOutput(conf,"text2",TextOutputFormat.class,Long.class,String.class);

MultipleOutputs.addNamedOutput(conf,"text3",TextOutputFormat.class,Long.class,String.class);

参考：http://my.oschina.net/leejun2005/blog/94706

发音：Multiple['m?lt?pl]--》许多的

### **3、如何为一个hadoop任务设置要创建的reducer的数量**

答案：job.setNumReduceTask(intn)

或者调整hdfs-site.xml中的mapred.tasktracker.reduce.tasks.maximum默认参数值

### **4、在hadoop中定义的主要公用InputFormats中，哪一个是默认值**

(A)TextInputFormat

(B)KeyValueInputFormat

(C)SequenceFileInputFormat

答案：A

### **5、两个类TextInputFormat和KeyValueTextInputFormat的区别？**

答案：

?FileInputFormat的子类：

TextInputFormat（默认类型，键是LongWritable类型，值为Text类型，key为当前行在文件中的偏移量，value为当前行本身）；

?KeyValueTextInputFormat(适合文件自带key，value的情况，只要指定分隔符即可，比较实用，默认是\t分割)；

源码：

StringsepStr=job.get("mapreduce.input.keyvaluelinerecordreader.key.value.separator","\t");

注意：在自定义输入格式时，继承FileInputFormat父类

参考：http://www.cnblogs.com/vichao/archive/2013/06/06/3118100.html

### **6、在一个运行的hadoop任务中，什么是InputSpilt?**

答案：InputSplit是MapReduce对文件进行处理和运算的输入单位，只是一个逻辑概念，每个InputSplit并没有对文件实际的切割，只是记录了要处理的数据的位置（包括文件的path和hosts）和长度（由start和length决定），默认情况下与block一样大。

拓展：需要在定义InputSplit后，展开讲解mapreduce的原理

### **7、Hadoop框架中，文件拆分是怎么被调用的？**

答案：JobTracker，创建一个InputFormat的实例，调用它的getSplits()方法，把输入目录的文件拆分成FileSplist作为Mappertask的输入，生成Mappertask加入Queue。

源码中体现了拆分的数量

longgoalSize=totalSize/(numSplits==0?1:numSplits);

longminSize=Math.max(job.getLong(org.apache.hadoop.mapreduce.lib.input.

FileInputFormat.SPLIT\_MINSIZE,1),minSplitSize);//minSplitSize默认是1

### **8、分别举例什么情况下使用combiner,什么情况下不会使用？**

答案：Combiner适用于对记录汇总的场景（如求和），但是，求平均数的场景就不能使用Combiner了

### **9、Hadoop中job和Tasks之间的区别是什么？**

答案：

job是工作的入口，负责控制、追踪、管理任务，也是一个进程

包含maptask和reducetask

Tasks是map和reduce里面的步骤，主要用于完成任务，也是线程

### **10、Hadoop中通过拆分任务到多个节点运行来实现并行计算，但是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种情况？**

答案：结果查看监控日志，得知产生这种现象的原因是数据倾斜问题

解决：

(1)调整拆分mapper的数量（partition数量）

(2)增加jvm

(3)适当地将reduce的数量变大

### **11、流API中的什么特性带来可以使mapreduce任务可以以不同语言(如perl\ruby\awk等)实现的灵活性？**

答案：用可执行文件作为Mapper和Reducer，接受的都是标准输入，输出的都是标准输出

参考：http://www.web520.cn/archives/9220

### **12、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3个文件的大小分别是：64k65MB127MB

Hadoop框架会把这些文件拆分为多少块？

答案：

64k------->一个block

65MB---->两个文件：64MB是一个block，1MB是一个block

127MB--->两个文件：64MB是一个block,63MB是一个block

### **13、Hadoop中的RecordReader的作用是什么？**

答案：属于split和mapper之间的一个过程

将inputsplit输出的行为一个转换记录，成为key-value的记录形式提供给mapper

### **14、Map阶段结束后，Hadoop框架会处理：Partitioning,shuffle和sort,在这个阶段都会发生了什么？**

答案：

MR一共有四个阶段，splitmapshuffreduce在执行完map之后，可以对map的输出结果进行分区，

分区：这块分片确定到哪个reduce去计算(汇总)

排序：在每个分区中进行排序，默认是按照字典顺序。

Group：在排序之后进行分组

### **15、如果没有定义partitioner,那么数据在被送达reducer前是如何被分区的？**

答案：

Partitioner是在map函数执行context.write()时被调用。

用户可以通过实现自定义的?Partitioner来控制哪个key被分配给哪个?Reducer。

查看源码知道：

如果没有定义partitioner，那么会走默认的分区Hashpartitioner

publicclassHashPartitioner<K,V>extendsPartitioner<K,V>{

/\*\*Use{@linkObject#hashCode()}topartition.\*/

publicintgetPartition(Kkey,Vvalue,intnumReduceTasks){

return(key.hashCode()&Integer.MAX\_VALUE)%numReduceTasks;

}

}

参考：http://blog.csdn.net/gamer\_gyt/article/details/47339755

### **16、什么是Combiner?**

答案：这是一个hadoop优化性能的步骤，它发生在map与reduce之间

目的：解决了数据倾斜的问题，减轻网络压力，实际上时减少了maper的输出

源码信息如下：

publicvoidreduce(Textkey,Iterator<LongWritable>values,

OutputCollector<Text,LongWritable>output,Reporterreporter)

throwsIOException{

LongWritablemaxValue=null;

while(values.hasNext()){

LongWritablevalue=values.next();

if(maxValue==null){

maxValue=value;

}elseif(value.compareTo(maxValue)>0){

maxValue=value;

}

}

output.collect(key,maxValue);

}

在collect实现类中，有这样一段方法

publicsynchronizedvoidcollect(Kkey,Vvalue)

throwsIOException{

outCounter.increment(1);

writer.append(key,value);

if((outCounter.getValue()%progressBar)==0){

progressable.progress();

}

}

下面是说明输出数量达到10000时，开始合并为一个maper

publicstaticfinallongDEFAULT\_COMBINE\_RECORDS\_BEFORE\_PROGRESS=10000;

Mapreduce原理详解：

http://my.oschina[.NET](http://lib.csdn.net/base/dotnet)/itblog/blog/275294

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 公司A：

### 1.讲讲你做的过的项目。项目里有哪些难点重点注意点呢？

### 2.讲讲多线程吧，要是你，你怎么实现一个线程池呢？

### 3.讲一下Mapreduce或者hdfs的原理和机制。map读取数据分片。

### 4.shuffle是什么？怎么调优？

### 5.项目用什么语言写？[Scala](http://lib.csdn.net/base/scala)？[scala](http://lib.csdn.net/base/scala)的特点？和[Java](http://lib.csdn.net/base/java)的区别？

### 6.理论基础怎么样，比如[数据结构](http://lib.csdn.net/base/datastructure)，里面的快速排序，或者，树？讲一讲你了解的树的知识？

### 7.数学怎么样呢？

### 8.讲一下[数据库](http://lib.csdn.net/base/mysql)，SQl，左外连接，原理，实现？

### 9.还了解过数据的什么知识？数据库引擎？

### 10.Hadoop的机架怎么配置的？

### 11.Hbase的设计有什么心得？

### 12.Hbase的操作是用的什么API还是什么工具？

### 13.对调度怎么理解.?用什么工具吗？

### 14.用kettle这种工具还是自己写程序？你们公司是怎么做的？

### 15.你们数据中心开发周期是多长？

### 16.你们hbase里面是存一些什么数据。

## 二面。三个人。

### 1.讲讲你做的项目。

### 2.平时对多线程这方面是怎么处理呢？异步是怎么思考呢？遇到的一些锁啊，是怎么做的呢？比如两个人同时操作一样东西。怎么做的呢？一些并发操作设计到一些变量怎么做的呢？

### 3.你们用的最多是http协议吧？有没有特殊的头呢？讲讲你对tcp/ip的理解？

### 4.有没有用过Zookeeper呢？Zookeeper的适用场景是什么？HA状态维护分布式锁全局配置文件管理操作Zookeeper是用的什么？

### [Spark](http://lib.csdn.net/base/spark)方面：

### 5.spark开发分两个方面？哪两个方面呢？

### 6.比如一个读取hdfs上的文件，然后count有多少行的操作，你可以说说过程吗。那这个count是在内存中，还是磁盘中计算的呢？磁盘中。

### 7.spark和Mapreduce快？为什么快呢？快在哪里呢？1.内存迭代。2.RDD设计。3,算子的设计。

### 8.sparksql又为什么比hive快呢？

### 10.RDD的数据结构是怎么样的？Partition数组。dependence

### 11.hadoop的生态呢。说说你的认识。hdfs底层存储hbase数据库hive数据仓库Zookeeper分布式锁spark大数据分析

## 公司B：

### 1.Spark工作的一个流程。

### 2.[Hbase](http://lib.csdn.net/base/hbase)的PUT的一个过程。

### 3.RDD算子里操作一个外部map比如往里面put数据。然后算子外再遍历map。有什么问题吗。

### 4.shuffle的过程。调优。

### 5.5个partition里面分布有12345678910.用算子求最大值或者和。不能用广播变量和累加器。或者sortbykey.

### 6.大表和小表join.

### 7.知道spark怎么读hbase吗？sparkonhbase.。华为的。

### 8.做过hbase的二级索引吗？

### 9.sortshuffle的优点？

### 10.stage怎么划分的？宽依赖窄依赖是什么？

## 公司W：

### 1.讲讲你做过的项目(一个整体思路)

### 2.问问大概情况。公司里集群规模。hbase数据量。数据规模。

### 3.然后挑选数据工厂开始详细问。问hbase.。加闲聊。

### 4.问二次排序是什么。topn是什么。二次排序要继承什么接口？

### 5.计算的数据怎么来的。

### 6.kakfadirect是什么，。为什么要用这个，有什么优点？。和其他的有什么区别。

### 7.问了shuffle过程。

### 8.怎么调优的，jvm怎么调优的？

### 9.jvm结构？堆里面几个区？

### 10.数据清洗怎么做的？

### 11.怎么用spark做数据清洗

### 12.跟我聊了spark的应用，商场里广告投放，以及黄牛检测

### 13.spark读取数据，是几个Partition呢？hdfs几个block就有几个Partition？

### 14.sparkonyarn的两种模式?client模式？和cluster模式？

### 15.jdbc？mysql的驱动包名字叫什么？

### 16.region多大会分区？

## 公司Q

#### 1.说说Mapreduce？一整个过程的理解。讲一下。

#### 2.hbase存数据用什么rowkey？加时间戳的话，会不会出现时间戳重复的问题，怎么做的呢？

#### 3.Spring的两大模块？AOP，IOC在你们项目中分别是怎么用的呢？

#### 4.你们集群的规模，数据量？

## 公司M

### 1.画图，画Spark的工作模式，部署分布架构图

### 2.画图，画图讲解spark工作流程。以及在集群上和各个角色的对应关系。

### 3.java自带有哪几种线程池。

### 4.数据是怎么收集的。kafka收集数据的原理？

### 5.画图，讲讲shuffle的过程。那你怎么在编程的时候注意避免这些性能问题。

### 6.讲讲列式存储的parquet文件底层格式。

### 7.dataset和dataframe？

### 8.通过什么方式学习spark的？

### 9.有哪些数据倾斜，怎么解决？

### 10.宽依赖窄依赖？

### 11.yarn的原理？

### 12.BlockManager怎么管理硬盘和内存的。

### 13.哪些算子操作涉及到shuffle

### 14.看过源码？你熟悉哪几个部分的源码？

### 15.集群上nodemanager和ResourceManager的数量关系？

### 16.spark怎么整合hive？大概这样。sparkonhive。hive还是hive执行引擎是spark。

## 其他人的：

### 1.Spark如何处理结构化数据，Spark如何处理非结构话数据？

### 2.Spark性能优化主要有哪些手段？

### 3.简要描述Spark分布式集群搭建的步骤

### 4.对于Spark你觉得他对于现有大数据的现状的优势和劣势在哪里？

### 5.对于算法是否进行过自主的研究设计？

### 6.简要描述你了解的一些数据挖掘算法与内容基本我有印象的就这几个问题，聊了2个多小时，脑子都差点被问干了

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url?

　　假如每个url大小为10bytes，那么可以估计每个文件的大小为50G×64=320G，远远大于内存限制的4G，所以不可能将其完全加载到内存中处理，可以采用分治的思想来解决。

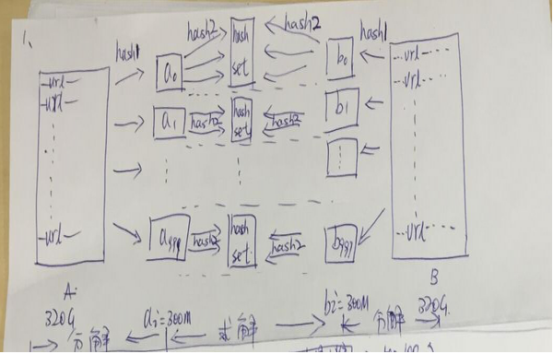
　　Step1：遍历文件a，对每个url求取hash(url)%1000，然后根据所取得的值将url分别存储到1000个小文件(记为a0,a1,...,a999，每个小文件约300M);

　　Step2:遍历文件b，采取和a相同的方式将url分别存储到1000个小文件(记为b0,b1,...,b999);

　　巧妙之处：这样处理后，所有可能相同的url都被保存在对应的小文件(a0vsb0,a1vsb1,...,a999vsb999)中，不对应的小文件不可能有相同的url。然后我们只要求出这个1000对小文件中相同的url即可。

　　Step3：求每对小文件ai和bi中相同的url时，可以把ai的url存储到hash\_set/hash\_map中。然后遍历bi的每个url，看其是否在刚才构建的hash\_set中，如果是，那么就是共同的url，存到文件里面就可以了。

　　草图如下(左边分解A，右边分解B，中间求解相同url)：



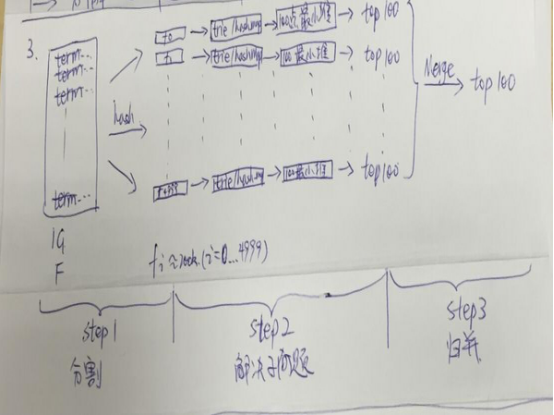
### 2.有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M，要求返回频数最高的100个词。

　　Step1：顺序读文件中，对于每个词x，取hash(x)%5000，然后按照该值存到5000个小文件(记为f0,f1,...,f4999)中，这样每个文件大概是200k左右，如果其中的有的文件超过了1M大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过1M;

　　Step2：对每个小文件，统计每个文件中出现的词以及相应的频率(可以采用trie树/hash\_map等)，并取出出现频率最大的100个词(可以用含100个结点的最小堆)，并把100词及相应的频率存入文件，这样又得到了5000个文件;

　　Step3：把这5000个文件进行归并(类似与归并排序);

　　草图如下(分割大问题，求解小问题，归并)：



### 3.现有海量日志数据保存在一个超级大的文件中，该文件无法直接读入内存，要求从中提取某天出访问百度次数最多的那个IP。

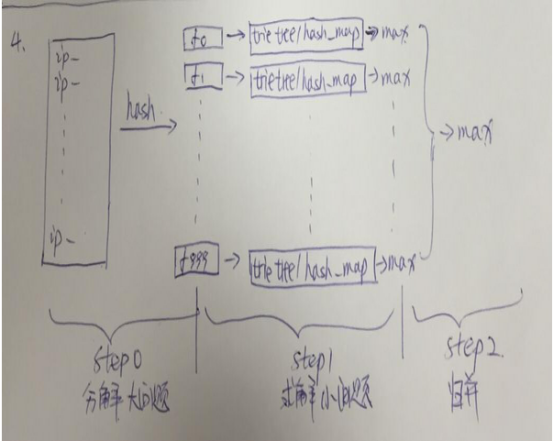
　　Step1：从这一天的日志数据中把访问百度的IP取出来，逐个写入到一个大文件中;

　　Step2：注意到IP是32位的，最多有2^32个IP。同样可以采用映射的方法，比如模1000，把整个大文件映射为1000个小文件;

　　Step3：找出每个小文中出现频率最大的IP(可以采用hash\_map进行频率统计，然后再找出频率最大的几个)及相应的频率;

　　Step4：在这1000个最大的IP中，找出那个频率最大的IP，即为所求。

　　草图如下：



### 4.LVS和HAProxy相比，它的缺点是什么?

　　之前，的确是用LVS进行过MySQL集群的负载均衡，对HAProxy也有过了解，但是将这两者放在眼前进行比较，还真没试着了解过。面试中出现了这么一题，面试官给予的答案是LVS的配置相当繁琐，后来查找了相关资料，对这两种负载均衡方案有了更进一步的了解。LVS的负载均衡性能之强悍已经达到硬件负载均衡的F5的百分之60了，而HAproxy的负载均衡和Nginx负载均衡，均为硬件负载均衡的百分之十左右。由此可见，配置复杂，相应的效果也是显而易见的。在查找资料的过程中，试着将LVS的10种调度算法了解了一下，看似数量挺多的10种算法其实在不同的算法之间，有些只是有着一些细微的差别。在这10种调度算法中，静态调度算法有四种，动态调度算法有6种。

静态调度算法：

　　①RR轮询调度算法

　　这种调度算法不考虑服务器的状态，所以是无状态的，同时也不考虑每个服务器的性能，比如我有1-N台服务器，来N个请求了，第一个请求给第一台，第二个请求给第二台，，，第N个请求给第N台服务器，就酱紫。

　　②加权轮询

　　这种调度算法是考虑到服务器的性能的，你可以根据不同服务器的性能，加上权重进行分配相应的请求。

　　③基于目的地址的hash散列

　　这种调度算法和基于源地址的hash散列异曲同工，都是为了维持一个session，基于目的地址的hash散列，将记住同一请求的目的地址，将这类请求发往同一台目的服务器。简而言之，就是发往这个目的地址的请求都发往同一台服务器。而基于源地址的hash散列，就是来自同一源地址的请求都发往同一台服务器。

　　④基于源地址的hash散列

　　上述已讲，不再赘述。

动态调度

　　①最少连接调度算法

　　这种调度算法会记录响应请求的服务器上所建立的连接数，每接收到一个请求会相应的将该服务器的所建立连接数加1，同时将新来的请求分配到当前连接数最少的那台机器上。

　　②加权最少连接调度算法

　　这种调度算法在最少连接调度算法的基础上考虑到服务器的性能。当然，做这样子的考虑是有其合理性存在的，如果是同一规格的服务器，那么建立的连接数越多，必然越增加其负载，那么仅仅根据最少连接数的调度算法，必然可以实现合理的负载均衡。但如果，服务器的性能不一样呢?比如我有一台服务器，最多只能处理10个连接，现在建立了3个，还有一台服务器最多能处理1000条连接，现在建立了5个，如果单纯地按照上述的最少连接调度算法，妥妥的前者嘛，但前者已经建立了百分之三十的连接了，而后者连百分之一的连接还没有建立，试问，这合理吗?显然不合理。所以加上权重，才算合理。相应的公式也相当简单：active\*256/weight。

　　③最短期望调度算法

　　这种算法，是避免出现上述加权最少连接调度算法中的一种特殊情况，导致即使加上权重，调度器也无差别对待了，举个栗子：

　　假设有三台服务器ABC，其当前所建立的连接数相应地为1,2,3，而权重也是1,2,3。那么如果按照加权最少连接调度算法的话，算出来是这样子的：

　　A:1256/1=256

　　B:2256/2=256

　　C:3256/3=256

　　我们会发现，即便加上权重，A、B、C，经过计算还是一样的，这样子调度器会无差别的在A、B、C中任选一台，将请求发过去。

　　而最短期望将active256/weight的算法改进为(active+1)256/weight

　　那么还是之前的例子：

　　A:(1+1)256/1=2/1256=2256

　　B:(2+1)256/2=3/2256=1.5256

　　C:(3+1)256、3=4/3256≈1.3256

　　显然C

　　④永不排队算法

　　将请求发给当前连接数为0的服务器上。

　　⑤基于局部的最少连接调度算法

　　这种调度算法应用于Cache系统，维持一个请求到一台服务器的映射，其实我们仔细想想哈，之前做的一系列最少连接相关的调度算法。考虑到的是服务器的状态与性能，但是一次请求并不是单向的，就像有一个从未合作过的大牛，他很闲，你让他去解决一个之前碰到过的一个问题，未必有找一个之前已经跟你合作过哪怕现在不怎么闲的臭皮匠效果好哦~，所以基于局部的最少连接调度算法，维持的这种映射的作用是，如果来了一个请求，相对应的映射的那台服务器，没有超载，ok交给老伙伴完事吧，俺放心，如果那台服务器不存在，或者是超载的状态且有其他服务器工作在一半的负载状态，则按最少连接调度算法在集群其余的服务器中找一台将请求分配给它。

　　⑥基于复制的局部最少连接调度算法

　　这种调度算法同样应用于cache系统，但它维持的不是到一台服务器的映射而是到一组服务器的映射，当有新的请求到来，根据最小连接原则，从该映射的服务器组中选择一台服务器，如果它没有超载则交给它去处理这个请求，如果发现它超载，则从服务器组外的集群中，按最少连接原则拉一台机器加入服务器组，并且在服务器组有一段时间未修改后，将最忙的那台服务器从服务器组中剔除。

### 5.Sqoop用起来感觉怎样?

　　说实话，Sqoop在导入数据的速度上确实十分感人，通过进一步了解，发现Sqoop1和Sqoop2在架构上还是有明显不同的，无论是从数据类型上还是从安全权限，密码暴露方面，Sqoop2都有了明显的改进，同时同一些其他的异构数据同步工具比较,如淘宝的DataX或者Kettle相比，Sqoop无论是从导入数据的效率上还是从支持插件的丰富程度上，Sqoop还是相当不错滴!!

### 6.ZooKeeper的角色以及相应的Zookepper工作原理?

　　果然，人的记忆力是有衰减曲线的，当面试官抛出这个问题后，前者角色，我只答出了两种(leader和follower)，后者原理压根就模糊至忘记了。所以恶补了一下，涉及到Zookeeper的角色大概有如下四种：leader、learner(follower)、observer、client。其中leader主要用来决策和调度，follower和observer的区别仅仅在于后者没有写的职能，但都有将client请求提交给leader的职能，而observer的出现是为了应对当投票压力过大这种情形的，client就是用来发起请求的。而Zookeeper所用的分布式一致性算法包括leader的选举其实和-原始部落的获得神器为酋长，或者得玉玺者为皇帝类似，谁id最小，谁为leader，会根据你所配置的相应的文件在相应的节点机下生成id，然后相应的节点会通过getchildren()这个函数获取之前设置的节点下生成的id，谁最小，谁是leader。并且如果万一这个leader挂掉了或者堕落了，则由次小的顶上。而且在配置相应的zookeeper文件的时候回有类似于如下字样的信息：Server.x=AAAA:BBBB:CCCC。其中的x即为你的节点号哈，AAAA对应你所部属zookeeper所在的ip地址，BBBB为接收client请求的端口，CCCC为重新选举leader端口。

### 7.HBase的Insert与Update的区别?

　　这个题目是就着最近的一次项目问的，当时实现的与hbase交互的三个方法分别为insert、delete、update。由于那个项目是对接的一个项目，对接的小伙伴和我协商了下，不将update合并为insert，如果合并的话，按那个项目本身，其实通过insert执行overwrite相当于间接地Update，本质上，或者说在展现上是没什么区别的包括所调用的put。但那仅仅是就着那个项目的程序而言，如果基于HBaseshell层面。将同一rowkey的数据插入HBase，其实虽然展现一条，但是相应的timestamp是不一样的，而且最大的版本数可以通过配置文件进行相应地设置。

### 8.请简述大数据的结果展现方式。

　　1)报表形式

　　基于数据挖掘得出的数据报表，包括数据表格、矩阵、图形和自定义格式的报表等，使用方便、设计灵活。

　　2)图形化展现

　　提供曲线、饼图、堆积图、仪表盘、鱼骨分析图等图形形式宏观展现模型数据的分布情况，从而便于进行决策。

　　3)KPI展现

　　提供表格式绩效一览表并可自定义绩效查看方式，如数据表格或走势图，企业管理者可根据可度量的目标快速评估进度。

　　4)查询展现

　　按数据查询条件和查询内容，以数据表格来汇总查询结果，提供明细查询功能，并可在查询的数据表格基础上进行上钻、下钻、旋转等操作。

### 9.例举身边的大数据。

　　i.QQ，微博等社交软件产生的数据

　　ii.天猫，京东等电子商务产生的数据

　　iii.互联网上的各种数据

### 10.简述大数据的数据管理方式。

　　答：对于图像、视频、URL、地理位置等类型多样的数据，难以用传统的结构化方式描述，因此需要使用由多维表组成的面向列存储的数据管理系统来组织和管理数据。也就是说，将数据按行排序，按列存储，将相同字段的数据作为一个列族来聚合存储。不同的列族对应数据的不同属性，这些属性可以根据需求动态增加，通过这样的分布式实时列式数据库对数据统一进行结构化存储和管理，避免了传统数据存储方式下的关联查询。

### 11.什么是大数据?

　　答：大数据是指无法在容许的时间内用常规软件工具对其内容进行抓取、管理和处理的数据。

### 12.海量日志数据，提取出某日访问百度次数最多的那个IP。

　　首先是这一天，并且是访问百度的日志中的IP取出来，逐个写入到一个大文件中。注意到IP是32位的，最多有个2^32个IP。同样可以采用映射的方法，比如模1000，把整个大文件映射为1000个小文件，再找出每个小文中出现频率最大的IP(可以采用hash\_map进行频率统计，然后再找出频率最大的几个)及相应的频率。然后再在这1000个最大的IP中，找出那个频率最大的IP，即为所求。

　　或者如下阐述(雪域之鹰)：

　　算法思想：分而治之+Hash

　　1)IP地址最多有2^32=4G种取值情况，所以不能完全加载到内存中处理;

　　2)可以考虑采用“分而治之”的思想，按照IP地址的Hash(IP)%1024值，把海量IP日志分别存储到1024个小文件中。这样，每个小文件最多包含4MB个IP地址;

　　3)对于每一个小文件，可以构建一个IP为key，出现次数为value的Hashmap，同时记录当前出现次数最多的那个IP地址;

　　4)可以得到1024个小文件中的出现次数最多的IP，再依据常规的排序算法得到总体上出现次数最多的IP;

### 13.搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。

　　假设目前有一千万个记录(这些查询串的重复度比较高，虽然总数是1千万，但如果除去重复后，不超过3百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门。)，请你统计最热门的10个查询串，要求使用的内存不能超过1G。

　　典型的TopK算法，还是在这篇文章里头有所阐述，详情请参见：十一、从头到尾彻底解析Hash表算法。

　　文中，给出的最终算法是：

　　第一步、先对这批海量数据预处理，在O(N)的时间内用Hash表完成统计(之前写成了排序，特此订正。July、2011.04.27);

　　第二步、借助堆这个数据结构，找出TopK，时间复杂度为N‘logK。

　　即，借助堆结构，我们可以在log量级的时间内查找和调整/移动。因此，维护一个K(该题目中是10)大小的小根堆，然后遍历300万的Query，分别和根元素进行对比所以，我们最终的时间复杂度是：O(N)+N’\*O(logK)，(N为1000万，N’为300万)。ok，更多，详情，请参考原文。

　　或者：采用trie树，关键字域存该查询串出现的次数，没有出现为0。最后用10个元素的最小推来对出现频率进行排序。

　　14.有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。

　　方案：顺序读文件中，对于每个词x，取hash(x)%5000，然后按照该值存到5000个小文件(记为x0,x1,…x4999)中。这样每个文件大概是200k左右。

　　如果其中的有的文件超过了1M大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过1M。

　　对每个小文件，统计每个文件中出现的词以及相应的频率(可以采用trie树/hash\_map等)，并取出出现频率最大的100个词(可以用含100个结点的最小堆)，并把100个词及相应的频率存入文件，这样又得到了5000个文件。下一步就是把这5000个文件进行归并(类似与归并排序)的过程了。

　　15.有10个文件，每个文件1G，每个文件的每一行存放的都是用户的query，每个文件的query都可能重复。要求你按照query的频度排序。

　　还是典型的TOPK算法，解决方案如下：

　　方案1：

　　顺序读取10个文件，按照hash(query)%10的结果将query写入到另外10个文件(记为)中。这样新生成的文件每个的大小大约也1G(假设hash函数是随机的)。

　　找一台内存在2G左右的机器，依次对用hash\_map(query,query\_count)来统计每个query出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的query和对应的query\_cout输出到文件中。这样得到了10个排好序的文件(记为)。

　　对这10个文件进行归并排序(内排序与外排序相结合)。

　　方案2：

　　一般query的总量是有限的，只是重复的次数比较多而已，可能对于所有的query，一次性就可以加入到内存了。这样，我们就可以采用trie树/hash\_map等直接来统计每个query出现的次数，然后按出现次数做快速/堆/归并排序就可以了。

　　方案3：

　　与方案1类似，但在做完hash，分成多个文件后，可以交给多个文件来处理，采用分布式的架构来处理(比如MapReduce)，最后再进行合并。

### 16.给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url?

　　方案1：可以估计每个文件安的大小为5G×64=320G，远远大于内存限制的4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

　　遍历文件a，对每个url求取hash(url)%1000，然后根据所取得的值将url分别存储到1000个小文件(记为a0,a1,…,a999)中。这样每个小文件的大约为300M。

　　遍历文件b，采取和a相同的方式将url分别存储到1000小文件(记为b0,b1,…,b999)。这样处理后，所有可能相同的url都在对应的小文件(a0vsb0,a1vsb1,…,a999vsb999)中，不对应的小文件不可能有相同的url。然后我们只要求出1000对小文件中相同的url即可。

　　求每对小文件中相同的url时，可以把其中一个小文件的url存储到hash\_set中。然后遍历另一个小文件的每个url，看其是否在刚才构建的hash\_set中，如果是，那么就是共同的url，存到文件里面就可以了。

　　方案2：如果允许有一定的错误率，可以使用Bloomfilter，4G内存大概可以表示340亿bit。将其中一个文件中的url使用Bloomfilter映射为这340亿bit，然后挨个读取另外一个文件的url，检查是否与Bloomfilter，如果是，那么该url应该是共同的url(注意会有一定的错误率)。

　　Bloomfilter日后会在本BLOG内详细阐述。

### 17.在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。

　　方案1：采用2-Bitmap(每个数分配2bit，00表示不存在，01表示出现一次，10表示多次，11无意义)进行，共需内存2^32\*2bit=1GB内存，还可以接受。然后扫描这2.5亿个整数，查看Bitmap中相对应位，如果是00变01，01变10，10保持不变。所描完事后，查看bitmap，把对应位是01的整数输出即可。

　　方案2：也可采用与第1题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

　　18.腾讯面试题：给40亿个不重复的unsignedint的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中?

　　与上第6题类似，我的第一反应时快速排序+二分查找。以下是其它更好的方法：

　　方案1：oo，申请512M的内存，一个bit位代表一个unsignedint值。读入40亿个数，设置相应的bit位，读入要查询的数，查看相应bit位是否为1，为1表示存在，为0表示不存在。

　　dizengrong：

　　方案2：这个问题在《编程珠玑》里有很好的描述，大家可以参考下面的思路，探讨一下：

　　又因为2^32为40亿多，所以给定一个数可能在，也可能不在其中;

　　这里我们把40亿个数中的每一个用32位的二进制来表示

　　假设这40亿个数开始放在一个文件中。

　　然后将这40亿个数分成两类:

　　1.最高位为0

　　2.最高位为1

　　并将这两类分别写入到两个文件中，其中一个文件中数的个数<=20亿，而另一个>=20亿(这相当于折半了);

　　与要查找的数的最高位比较并接着进入相应的文件再查找

　　再然后把这个文件为又分成两类:

　　1.次最高位为0

　　2.次最高位为1

　　并将这两类分别写入到两个文件中，其中一个文件中数的个数<=10亿，而另一个>=10亿(这相当于折半了);

　　与要查找的数的次最高位比较并接着进入相应的文件再查找。

　　…….

　　以此类推，就可以找到了,而且时间复杂度为O(logn)，方案2完。

　　附：这里，再简单介绍下，位图方法：

　　使用位图法判断整形数组是否存在重复

　　判断集合中存在重复是常见编程任务之一，当集合中数据量比较大时我们通常希望少进行几次扫描，这时双重循环法就不可取了。

　　位图法比较适合于这种情况，它的做法是按照集合中最大元素max创建一个长度为max+1的新数组，然后再次扫描原数组，遇到几就给新数组的第几位置上1，如遇到5就给新数组的第六个元素置1，这样下次再遇到5想置位时发现新数组的第六个元素已经是1了，这说明这次的数据肯定和以前的数据存在着重复。这种给新数组初始化时置零其后置一的做法类似于位图的处理方法故称位图法。它的运算次数最坏的情况为2N。如果已知数组的最大值即能事先给新数组定长的话效率还能提高一倍。

　　欢迎，有更好的思路，或方法，共同交流。

　　19.怎么在海量数据中找出重复次数最多的一个?

　　方案1：先做hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求(具体参考前面的题)。

　　20.上千万或上亿数据(有重复)，统计其中出现次数最多的钱N个数据。

　　方案1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用hash\_map/搜索二叉树/红黑树等来进行统计次数。然后就是取出前N个出现次数最多的数据了，可以用第2题提到的堆机制完成。

　　21.一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前10个词，请给出思想，给出时间复杂度分析。

　　方案1：这题是考虑时间效率。用trie树统计每个词出现的次数，时间复杂度是O(n\*le)(le表示单词的平准长度)。然后是找出出现最频繁的前10个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是O(n\*lg10)。所以总的时间复杂度，是O(n\*le)与O(n\*lg10)中较大的哪一个。

　　附、100w个数中找出最大的100个数。

　　方案1：在前面的题中，我们已经提到了，用一个含100个元素的最小堆完成。复杂度为O(100w\*lg100)。

　　方案2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分在比100多的时候，采用传统排序算法排序，取前100个。复杂度为O(100w\*100)。

方案3：采用局部淘汰法。选取前100个元素，并排序，记为序列L。然后一次扫描剩余的元素x，与排好序的100个元素中最小的元素比，如果比这个最小的要大，那么把这个最小的元素删除，并把x利用插入排序的思想，插入到序列L中。依次循环，知道扫描了所有的元素。复杂度为O(100w\*100)。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 1、将现有逻辑在[Spark](http://lib.csdn.net/base/spark)上面实现。

### 2、数据倾斜怎么处理？

数据倾斜有很多解决方案，本例子简要介绍一种实现方式，假设表A和表B连接，表A数据倾斜，只有一个key倾斜，首先对A进行采样，统计出最倾斜的key，将A表分隔为A1只有倾斜key，A2不包含倾斜key，然后分别与表B连接。

最后将结果合并，union

### 3、各完成一个awk和sed的例子，最简单的应用即可，并说明。

cat/etc/passwd|awk-F':''{print$1}'读取文件，指定文本分隔符，然后打印第一个域

cattest.sh|sed'2,50d'读取文件的每一行，然后对每一行进行处理后打印在控制台，有很多模式，很多方法。还可以直接修改文件

### 4、简要描述你知道的数据挖掘[算法](http://lib.csdn.net/base/datastructure)和使用场景

（一）基于分类模型的案例

（1）垃圾邮件的判别通常会采用朴素贝叶斯的方法进行判别

（2）医学上的肿瘤判断通过分类模型识别

（二）基于预测模型的案例

（1）红酒品质的判断分类回归树模型进行预测和判断红酒的品质

（2）[搜索引擎](http://lib.csdn.net/base/searchengine)的搜索量和股价波动

（三）基于关联分析的案例：沃尔玛的啤酒尿布

（四）基于聚类分析的案例：零售客户细分

（五）基于异常值分析的案例：支付中的交易欺诈侦测

（六）基于协同过滤的案例：电商猜你喜欢和推荐引擎

（七）基于社会网络分析的案例：电信中的种子客户

（八）基于文本分析的案例

（1）字符识别：扫描王APP

（2）文学著作与统计：红楼梦归属

### 5、列举你知道的常用的[Hadoop](http://lib.csdn.net/base/hadoop)管理和监控的命令、比如hdfsdfs-mkdir/usr

-ls-cat-text-cp-put-chmod-chown

-du-get-copyFromLocal-copyToLocal

-mv-rm-tail-chgrp

### 6、评述[hadoop](http://lib.csdn.net/base/hadoop)运行原理

站在高处，大的方面讲解

1、有hdfs负责数据存放是Hadoop的分布式文件存储系统

1. 将大文件分解为多个Block，每个Block保存多个副本。提供容错机制，副本丢失或者宕机时自动恢复。
2. 默认每个Block保存3个副本，64M为1个Block。
3. 将Block按照key-value映射到内存当中。

2、有mapreduce负责计算，Map（映射）和Reduce（归约）

### 7、讲述mapreduce的原理

ApplicationMaster是一个详细的框架库，它结合从ResourceManager获得的资源和NodeManager协同工作来运行和监控任务。

ResourceManager支持分层级的应用队列，这些队列享有集群一定比例的资源。从某种意义上讲它就是一个纯粹的调度器，

ApplicationMaster的职责有：向调度器索要适当的资源容器，运行任务，跟踪应用程序的状态和监控它们的进程，处理任务的失败原因。

输入分片（inputsplit）

map阶段：

combiner阶段：

### 8、[JVM&垃圾回收机制](http://blog.csdn.net/paul_wei2008/article/details/19340063)

**三个代：**年轻代（YoungGeneration）、年老代（OldGeneration）和持久代（PermanentGeneration）**。**

### 9、找出公共好友：

http://www.cnblogs.com/lucius/p/3483494.html

http://www.aboutyun.com/thread-18826-1-1.html

原理：A有好友B,C,DF有好友DEF

其实A的好友也是他好友的好友

其实F的好友也是他的好友的好友

那么D的共同好友就是AF

### 10、combiner作用

Combiner最基本是实现本地key的聚合，对map输出的key排序、value进行迭代。

Combiner还有本地reduce功能（其本质上就是一个reduce）：

使用Combiner先完成在map端的本地聚合，可以减少网络传输数据量，提高性能。

平均数的归约算法不能多次调用。

### 11、在mr环节中，那些环节需要优化，如何优化，请详细说明。

1、setNumReduceTasks适当的设置reduce的数量，如果数据量比较大，那么可以增加reduce的数量

2、适当的时候使用combine函数，减少网络传输数据量

3、压缩map和reduce的输出数据

4、使用SequenceFile二进制文件。

5、通过application的ui页面观察job的运行参数

6、太多小文件，造成map任务过多的问题，应该可以先合并小文件，或者有一个特定的map作为处理小文件的输入

7、map端效率低原因分析

* 源文件的大小远小于HDFS的块的大小。这意味着任务的开启和停止要耗费更多的时间，就没有足够的时间来读取并处理输入数据。
* 源文件无法分块。这导致需要通过网络IO从其他节点读取文件块。
* 一个节点的本地磁盘或磁盘控制器运行在降级模式中，读取写入性能都很差。这会影响某个节点，而不是全部节点。
* 源文件不来自于HDFS。则可能是Hadoop节点和数据源之间的延迟导致了性能低下。
* Map任务从其他数据节点读取数据。可以从JobTracker的map任务细节信息和任务运行尝试中找到输入块的位置。如果输入块的位置不是任务执行的节点，那就不是本地数据了。

技术33Reduce实例不足或过多

技术34诊断reduce段的数据倾斜的问题

技术35确定reduce任务是否存在整体吞吐量过低

技术36缓慢的洗牌（shuffle）和排序

技术37作业竞争和调度器限制

1.查找硬件的失效

CPU竞争

3内存交换

4磁盘健康

网络

技术46规避使用reduce

技术48使用combine

技术50收集倾斜数据

减小Reduce端数据倾斜的性能损失

抽样和范围分区

方法2：自定义分区

方法3：Combine

方法4：Map端连接和半连接

数据大小倾斜的自定义策略

1.正则表达式（少用）

2.字符串令牌化（TOKENIZATION）Apachecommons中的StringUtils类效率要更好

3.对象重用

4字符串连接

5对象的内存资源消耗

6.4.6优化数据序列化

压缩

二进制文件格式使用二进制文件格式，如Avro和SequenceFile，可以使数据的表达更为紧凑，并提高编组（marshalling）和逆编组的效率，相比文本格式要更为高效。

1. 确保精确的度量，理解如何获得MapReduce和系统的性能指标
2. 使用性能指标来减少潜在的性能问题
3. 通过检查MapRecue/HDFS配置，优化MapReduce洗牌/排序阶段，优化用户JAVA代码，来修复常见的性能问题

5.1小文件

技术24使用Avro存储多个小文件

最简单的方案就是将HDFS中的小文件打包到一个大的文件容器中。这个技术中将本地磁盘中所有的目标文件存储到HDFS中的一个单独的Avro文件。然后在MapReduce中处理Avro文件和其中的小文件。

压缩依赖

CombineFileInputFormat

基于压缩的高效存储

-------------------------------

Hadoop（1）MapReduce性能调优：性能测量（Measuring）

<http://www.aboutyun.com/thread-15514-1-1.html>

Hadoop（2）MapReduce性能调优：理解性能瓶颈，诊断map性能瓶颈

<http://www.aboutyun.com/thread-15517-1-1.html>

Hadoop（3）MapReduce性能调优：诊断reduce性能瓶颈

<http://www.aboutyun.com/thread-15522-1-1.html>

Hadoop（4）MapReduce性能调优：诊断一般性能瓶颈

<http://www.aboutyun.com/thread-15660-1-1.html>

Hadoop（5）MapReduce性能调优：诊断硬件性能瓶颈

<http://www.aboutyun.com/thread-15534-1-1.html>

Hadoop（6）MapReduce性能调优：优化洗牌（shuffle）和排序阶段

<http://www.aboutyun.com/thread-15545-1-1.html>

Hadoop（7）MapReduce性能调优：减小数据倾斜的性能损失

<http://www.aboutyun.com/thread-15544-1-1.html>

Hadoop（8）MapReduce性能调优：优化MapReduce的用户[Java](http://lib.csdn.net/base/java)代码

<http://www.aboutyun.com/thread-15583-1-1.html>

Hadoop（9）MapReduce性能调优：优化数据序列化

<http://www.aboutyun.com/thread-15658-1-1.html>

Hadoop（10）MapReduce文件处理：小文件

<http://www.aboutyun.com/thread-15592-1-1.html>

Hadoop（11）MapReduce文件处理：基于压缩的高效存储（一）

<http://www.aboutyun.com/thread-15626-1-1.html>

Hadoop（12）MapReduce文件处理：基于压缩的高效存储（二）

<http://www.aboutyun.com/thread-15629-1-1.html>

### 1、简答说一下hadoop的map-reduce编程模型

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合

使用的是hadoop内置的数据类型，比如longwritable、text等

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

### 2、hadoop的TextInputFormat作用是什么，如何自定义实现

InputFormat会在map操作之前对数据进行两方面的预处理

1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次

### 2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法

在createRecordReader中可以自定义分隔符

### 3、hadoop和spark的都是并行计算，那么他们有什么相同和区别

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

### 4、为什么要用flume导入hdfs，hdfs的构架是怎样的

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死

### 5、map-reduce程序运行的时候会有什么比较常见的问题

比如说作业中大部分都完成了，但是总有几个reduce一直在运行

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作

### 6、简单说一下hadoop和spark的shuffle过程

hadoop：map端保存分片数据，通过网络收集到reduce端

spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor

减少shuffle可以提高性能

部分答案不是十分准确欢迎补充:-)

——-补充更新———

### 1、Hive中存放是什么？

表。

存的是和hdfs的映射关系，hive是逻辑上的数据仓库，实际操作的都是hdfs上的文件，HQL就是用sql语法来写的mr程序。

2、Hive与关系型数据库的关系？

没有关系，hive是数据仓库，不能和数据库一样进行实时的CURD操作。

是一次写入多次读取的操作，可以看成是ETL工具。

### 3、Flume工作机制是什么？

核心概念是agent，里面包括source、chanel和sink三个组件。

source运行在日志收集节点进行日志采集，之后临时存储在chanel中，sink负责将chanel中的数据发送到目的地。

只有成功发送之后chanel中的数据才会被删除。

首先书写flume配置文件，定义agent、source、chanel和sink然后将其组装，执行flume-ng命令。

### 4、Sqoop工作原理是什么？

hadoop生态圈上的数据传输工具。

可以将关系型数据库的数据导入非结构化的hdfs、hive或者bbase中，也可以将hdfs中的数据导出到关系型数据库或者文本文件中。

使用的是mr程序来执行任务，使用jdbc和关系型数据库进行交互。

import原理：通过指定的分隔符进行数据切分，将分片传入各个map中，在map任务中在每行数据进行写入处理没有reduce。

export原理：根据要操作的表名生成一个java类，并读取其元数据信息和分隔符对非结构化的数据进行匹配，多个map作业同时执行写入关系型数据库

### 5、Hbase行健列族的概念，物理模型，表的设计原则？

行健：是hbase表自带的，每个行健对应一条数据。

列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中的数据可以有很多，通过时间戳来区分。

物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。

rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。

列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

### 6、SparkStreaming和Storm有何区别？

一个实时毫秒一个准实时亚秒，不过storm的吞吐率比较低。

### 7、mllib支持的算法？

大体分为四大类，分类、聚类、回归、协同过滤。

### 8、简答说一下hadoop的map-reduce编程模型？

首先maptask会从本地文件系统读取数据，转换成key-value形式的键值对集合。

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出。

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则。

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出。

在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则。

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量。

reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job。

### 9、Hadoop平台集群配置、环境变量设置？

zookeeper：修改zoo.cfg文件，配置dataDir，和各个zk节点的server地址端口，tickTime心跳时间默认是2000ms，其他超时的时间都是以这个为基础的整数倍，之后再dataDir对应目录下写入myid文件和zoo.cfg中的server相对应。

hadoop：修改

hadoop-env.sh配置java环境变量

core-site.xml配置zk地址，临时目录等

hdfs-site.xml配置nn信息，rpc和http通信地址，nn自动切换、zk连接超时时间等

yarn-site.xml配置resourcemanager地址

mapred-site.xml配置使用yarn

slaves配置节点信息

格式化nn和zk。

hbase：修改

hbase-env.sh配置java环境变量和是否使用自带的zk

hbase-site.xml配置hdfs上数据存放路径，zk地址和通讯超时时间、master节点

regionservers配置各个region节点

zoo.cfg拷贝到conf目录下

spark：

安装Scala

修改spark-env.sh配置环境变量和master和worker节点配置信息

环境变量的设置：直接在/etc/profile中配置安装的路径即可，或者在当前用户的宿主目录下，配置在.bashrc文件中，该文件不用source重新打开shell窗口即可，配置在.bash\_profile的话只对当前用户有效。

### 10、Hadoop性能调优？

调优可以通过系统配置、程序编写和作业调度算法来进行。

hdfs的block.size可以调到128/256（网络很好的情况下，默认为64）

调优的大头：mapred.map.tasks、mapred.reduce.tasks设置mr任务数（默认都是1）

mapred.tasktracker.map.tasks.maximum每台机器上的最大map任务数

mapred.tasktracker.reduce.tasks.maximum每台机器上的最大reduce任务数

mapred.reduce.slowstart.completed.maps配置reduce任务在map任务完成到百分之几的时候开始进入

这个几个参数要看实际节点的情况进行配置，reduce任务是在33%的时候完成copy，要在这之前完成map任务，（map可以提前完成）

mapred.compress.map.output,mapred.output.compress配置压缩项，消耗cpu提升网络和磁盘io

合理利用combiner

注意重用writable对象

### 11、Hadoop高并发？

首先肯定要保证集群的高可靠性，在高并发的情况下不会挂掉，支撑不住可以通过横向扩展。

datanode挂掉了使用hadoop脚本重新启动。

12、hadoop的TextInputFormat作用是什么，如何自定义实现？

InputFormat会在map操作之前对数据进行两方面的预处理。

1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次。

2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map。

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值。

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法。

在createRecordReader中可以自定义分隔符。

### 13、hadoop和spark的都是并行计算，那么他们有什么相同和区别？

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为maptask和reducetask，每个task都是在自己的进程中运行的，当task结束时，进程也会结束。

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job。

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算。

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系。

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错。

### 14、为什么要用flume导入hdfs，hdfs的构架是怎样的？

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件。

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死。

### 15、map-reduce程序运行的时候会有什么比较常见的问题？

比如说作业中大部分都完成了，但是总有几个reduce一直在运行。

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜。

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作。

### 16、简单说一下hadoop和spark的shuffle过程？

hadoop：map端保存分片数据，通过网络收集到reduce端。

spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor。

减少shuffle可以提高性能。

### 17、RDD机制？

rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。

所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。

rdd执行过程中会形成dag图，然后形成lineage保证容错性等。

从物理的角度来看rdd存储的是block和node之间的映射。

### 18、spark有哪些组件？

（1）master：管理集群和节点，不参与计算。

（2）worker：计算节点，进程本身不参与计算，和master汇报。

（3）Driver：运行程序的main方法，创建sparkcontext对象。

（4）sparkcontext：控制整个application的生命周期，包括dagsheduler和taskscheduler等组件。

（5）client：用户提交程序的入口。

### 19、spark工作机制？

用户在client端提交作业后，会由Driver运行main方法并创建sparkcontext上下文。

执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入taskscheduler。

taskscheduler会将stage划分为taskset分发到各个节点的executor中执行。

### 20、spark的优化怎么做？

通过spark-env文件、程序中sparkconf和setproperty设置。

（1）计算量大，形成的lineage过大应该给已经缓存了的rdd添加checkpoint，以减少容错带来的开销。

（2）小分区合并，过小的分区造成过多的切换任务开销，使用repartition。

### 21、kafka工作原理？

producer向broker发送事件，consumer从broker消费事件。

事件由topic区分开，每个consumer都会属于一个group。

相同group中的consumer不能重复消费事件，而同一事件将会发送给每个不同group的consumer。

### 22、ALS算法原理？

答：对于user-product-rating数据，als会建立一个稀疏的评分矩阵，其目的就是通过一定的规则填满这个稀疏矩阵。

als会对稀疏矩阵进行分解，分为用户-特征值，产品-特征值，一个用户对一个产品的评分可以由这两个矩阵相乘得到。

通过固定一个未知的特征值，计算另外一个特征值，然后交替反复进行最小二乘法，直至差平方和最小，即可得想要的矩阵。

### 23、kmeans算法原理？

随机初始化中心点范围，计算各个类别的平均值得到新的中心点。

重新计算各个点到中心值的距离划分，再次计算平均值得到新的中心点，直至各个类别数据平均值无变化。

### 24、canopy算法原理？

根据两个阈值来划分数据，以随机的一个数据点作为canopy中心。

计算其他数据点到其的距离，划入t1、t2中，划入t2的从数据集中删除，划入t1的其他数据点继续计算，直至数据集中无数据。

### 25、朴素贝叶斯分类算法原理？

对于待分类的数据和分类项，根据待分类数据的各个特征属性，出现在各个分类项中的概率判断该数据是属于哪个类别的。

### 26、关联规则挖掘算法apriori原理？

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、你会Java语言吗？熟悉到什么程度？

2、你最喜欢的编程语言是什么？为什么？

3、处理过的最大的数据量？你是如何处理他们的？处理的结果如何。

2、在处理大数据过程中，如何保证得到期望值？

3、如何让一个网络爬虫速度更快、抽取更好的信息以及更好总结数据从而得到一干净的数据库？

4、点击流数据应该是实时处理？为什么？哪部分应该实时处理？

6、如何把非结构化的数据转换成结构化的数据？这是否真的有必要做这样的转换？把数据存成平面文本文件是否比存成关系数据库更好？

7、如何判别mapreduce过程有好的负载均衡？什么是负载均衡？

8、Spark和Hive的区别，以及Spark和Hive的数据倾斜调优问题？

9、Hive和Hbase的区别？

10、MapReduce的思想，以及MapReduce调优问题？

11、你所了解的开源网站？

12、有两个集群，每个集群有3个节点，使用hive分析相同的数据，sql语句完全一样，一个集群的分析结果比另外一个慢的多，给出造成这种现象的可能原因？

13、Hbase的优化？

14、集群的版本，以及集群的瓶颈问题？

15、CRM项目，怎么跟Spark结合？

16、如何创建一个关键字分类？

17、海量日志数据，提取出某日访问百度次数最多的那个IP？

18、Hadoop和Spark处理数据时，出现内存溢出的处理方法？

19、有一个1G大小的一个文件，里面每一是一个词，词的大小不超过16字节，内存大小限制大小1M，返回频率最高的50个词。

20、你是如何处理缺少数据的？你是推荐使用什么样的处理技术，或者说你是用什么样的技术处理呢？

1.简要描述如何安装配置一个apache开源版hadoop，描述即可，列出步骤更好

2.请列出正常工作的hadoop集群中hadoop都需要启动哪些进程，他们的作用分别是什么？

3.启动hadoop报如下错误，该如何解决？

errororg.apache.hadoop.hdfs.server.namenode.NameNode

org.apache.hadoop.hdfs.server.common.inconsistentFSStateExceptio

nDirectory/tmp/hadoop-root/dfs/nameisinaninconsistent

statestoragedirectiondoesnotexistorisnotaccessible?

4.请写出以下执行命令

1）杀死一个job?

2)删除hdfs上的/tmp/aaa目录

3加入一个新的存储节点和删除一个计算节点需要刷新集群状态命令？

5.请列出你所知道的hadoop调度器，并简要说明其工作方法？

6.请列出在你以前工作中所使用过的开发mapreduce的语言？

7.当前日志采样格式为

1. a,b,c,d
2. b,b,f,e
3. a,a,c,f

复制代码

请用你最熟悉的语言编写一个mapreduce，并计算第四列每个元素出现的个数

8.你认为用Java，Streaming,pipe方式开发mapreduce,各有哪些优缺点？

9.hive有哪些方式保存元数据，各有哪些特点？

10.请简述hadoop怎么样实现二级排序？

11.简述hadoop实现join的几种方法？

12.请用Java实现非递归二分查找？

13.请简述mapreduce中，combiner，partition作用？

14.某个目录下有两个文件a.txt和b.txt,文件格式为（ip，username）,

列如：

a.txt

127.0.0.1zhangsan

127.0.0.1wangxiaoer

127.0.0.2lisi

127.0.0.3wangwu

b.txt

127.0.0.4lixiaolu

127.0.0.1lisi

每个文件至少100万行，请使用Linux命令完成如下工作：

1）每个文件各自的ip数

2）出现在b.txt而没有出现在a.txt的ip

3）每个user出现的次数以及每个user对应的ip数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Hadoop集群可以运行的3个模式？

单机（本地）模式

伪分布式模式

全分布式模式

2.单机（本地）模式中的注意点？

在单机模式（standalone）中不会存在守护进程，所有东西都运行在一个JVM上。这里同样没有DFS，使用的是本地文件系统。单机模式适用于开发过程中运行MapReduce程序，这也是最少使用的一个模式。

3.伪分布模式中的注意点？

伪分布式（Pseudo）适用于开发和测试环境，在这个模式中，所有守护进程都在同一台机器上运行。

4.VM是否可以称为Pseudo？

不是，两个事物，同时Pseudo只针对Hadoop。

5.全分布模式又有什么注意点？

全分布模式通常被用于生产环境，这里我们使用N台主机组成一个Hadoop集群，Hadoop守护进程运行在每台主机之上。这里会存在Namenode运行的主机，Datanode运行的主机，以及tasktracker运行的主机。在分布式环境下，主节点和从节点会分开。

6.Hadoop是否遵循UNIX模式？

是的，在UNIX用例下，Hadoop还拥有“conf”目录。

7.Hadoop安装在什么目录下？

Cloudera和Apache使用相同的目录结构，Hadoop被安装在cdusrlibhadoop-0.20。

8.Namenode、Jobtracker和tasktracker的端口号是？

Namenode，70；Jobtracker，30；Tasktracker，60。

9.Hadoop的核心配置是什么？

Hadoop的核心配置通过两个xml文件来完成：1，hadoop-default.xml；2，hadoop-site.xml。这些文件都使用xml格式，因此每个xml中都有一些属性，包括名称和值，但是当下这些文件都已不复存在。

10.那当下又该如何配置？

Hadoop现在拥有3个配置文件：1，core-site.xml；2，hdfs-site.xml；3，mapred-site.xml。这些文件都保存在conf子目录下。

11.RAM的溢出因子是？

溢出因子（Spillfactor）是临时文件中储存文件的大小，也就是Hadoop-temp目录。

12.fs.mapr.working.dir只是单一的目录？

fs.mapr.working.dir只是一个目录。

13.hdfs-site.xml的3个主要属性？

dfs.name.dir决定的是元数据存储的路径以及DFS的存储方式（磁盘或是远端）

dfs.data.dir决定的是数据存储的路径

fs.checkpoint.dir用于第二Namenode

14.如何退出输入模式？

退出输入的方式有：1，按ESC；2，键入q（如果你没有输入任何当下）或者键入wq（如果你已经输入当下），并且按下Enter。

15.当你输入hadoopfsck造成“connectionrefusedjavaexception’”时，系统究竟发生了什么？

这意味着Namenode没有运行在你的VM之上。

16.我们使用Ubuntu及Cloudera，那么我们该去哪里下载Hadoop，或者是默认就与Ubuntu一起安装？

这个属于Hadoop的默认配置，你必须从Cloudera或者Edureka的dropbox下载，然后在你的系统上运行。当然，你也可以自己配置，但是你需要一个Linuxbox，Ubuntu或者是RedHat。在Cloudera网站或者是Edureka的Dropbox中有安装步骤。

17.“jps”命令的用处？

这个命令可以检查Namenode、Datanode、TaskTracker、JobTracker是否正常工作。

18.如何重启Namenode？

点击stop-all.sh，再点击start-all.sh。

键入sudohdfs（Enter），su-hdfs（Enter），etcinit.dha（Enter），及etcinit.dhadoop-0.20-namenodestart（Enter）。

19.Fsck的全名？

全名是：FileSystemCheck。

20.如何检查Namenode是否正常运行？

如果要检查Namenode是否正常工作，使用命令etcinit.dhadoop-0.20-namenodestatus或者就是简单的jps。

21.mapred.job.tracker命令的作用？

可以让你知道哪个节点是JobTracker。

22.etcinit.d命令的作用是？

etcinit.d说明了守护进程（服务）的位置或状态，其实是LINUX特性，和Hadoop关系不大。

23.如何在浏览器中查找Namenode？

如果你确实需要在浏览器中查找Namenode，你不再需要localhost8021，Namenode的端口号是50070。

24.如何从SU转到Cloudera？

从SU转到Cloudera只需要键入exit。

25.启动和关闭命令会用到哪些文件？

Slaves及Masters。

26.Slaves由什么组成？

Slaves由主机的列表组成，每台1行，用于说明数据节点。

27.Masters由什么组成？

Masters同样是主机的列表组成，每台一行，用于说明第二Namenode服务器。

28.hadoop-env.sh是用于做什么的？

hadoop-env.sh提供了Hadoop中.JAVA\_HOME的运行环境。

29.Master文件是否提供了多个入口？

是的你可以拥有多个Master文件接口。

30.Hadoop-env.sh文件当下的位置？

hadoop-env.sh现在位于conf。

31.在Hadoop\_PID\_DIR中，PID代表了什么？

PID代表了“ProcessID”。

32.varhadooppids用于做什么？

varhadooppids用来存储PID。

33.hadoop-metrics.properties文件的作用是？

hadoop-metrics.properties被用做“Reporting”，控制Hadoop报告，初始状态是“nottoreport”。

34.Hadoop需求什么样的网络？

Hadoop核心使用Shell（SSH）来驱动从节点上的服务器进程，并在主节点和从节点之间使用password-lessSSH连接。

35.全分布式环境下为什么需求password-lessSSH？

这主要因为集群中通信过于频繁，JobTracker需要尽可能快的给TaskTracker发布任务。

36.这会导致安全问题吗？

完全不用担心。Hadoop集群是完全隔离的，通常情况下无法从互联网进行操作。与众不同的配置，因此我们完全不需要在意这种级别的安全漏洞，比如说通过互联网侵入等等。Hadoop为机器之间的连接提供了一个相对安全的方式。

37.SSH工作的端口号是？

SSH工作的端口号是NO.22，当然可以通过它来配置，22是默认的端口号。

38.SSH中的注意点还包括？

SSH只是个安全的shell通信，可以把它当做NO.22上的一种协议，只需要配置一个密码就可以安全的访问。

39.为什么SSH本地主机需要密码？

在SSH中使用密码主要是增加安全性，在某些情况下也根本不会设置密码通信。

40.如果在SSH中添加key，是否还需要设置密码？

是的，即使在SSH中添加了key，还是需要设置密码。

41.假如Namenode中没有数据会怎么样？

没有数据的Namenode就不能称之为Namenode，通常情况下，Namenode肯定会有数据。

42.当JobTracker宕掉时，Namenode会发生什么？

当JobTracker失败时，集群仍然可以正常工作，只要Namenode没问题。

43.是客户端还是Namenode决定输入的分片？

这并不是客户端决定的，在配置文件中以及决定分片细则。

44.是否可以自行搭建Hadoop集群？

是的，只要对Hadoop环境足够熟悉，你完全可以这么做。

45.是否可以在Windows上运行Hadoop？

你最好不要这么做，RedHatLinux或者是Ubuntu才是Hadoop的最佳操作系统。在Hadoop安装中，Windows通常不会被使用，因为会出现各种各样的问题。因此，Windows绝对不是Hadoop的推荐系统。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

问题导读

1、当前集群的可用资源不能满足应用程序的需求，怎么解决？

2、内存里堆的东西太多了，有什么好办法吗？

1、WARNTaskSchedulerImpl:Initialjobhasnotacceptedanyresources;checkyourclusteruitoensurethatworkersareregisteredandhavesufficientmemory

当前的集群的可用资源不能满足应用程序所请求的资源。

资源分2类：cores和ram

Core代表对执行可用的executorslots

Ram代表每个Worker上被需要的空闲内存来运行你的Application。

解决方法：

应用不要请求多余空闲可用资源的

关闭掉已经执行结束的Application

2、Applicationisn’tusingalloftheCores:HowtosettheCoresusedbyaSparkApp

设置每个App所能获得的core

解决方法：

spark-env.sh里设置spark.deploy.defaultCores

或

spark.cores.max

3、SparkExecutorOOM:HowtosetMemoryParametersonSpark

OOM是内存里堆的东西太多了

1、增加job的并行度，即增加job的partition数量，把大数据集切分成更小的数据，可以减少一次性load到内存中的数据量。InputFomart，getSplit来确定。

2、spark.storage.memoryFraction

管理executor中RDD和运行任务时的内存比例，如果shuffle比较小，只需要一点点shufflememory，那么就调大这个比例。默认是0.6。不能比老年代还要大。大了就是浪费。

3、spark.executor.memory如果还是不行，那么就要加Executor的内存了，改完executor内存后，这个需要重启。

4、SharkServer/LongRunningApplicationMetadataCleanup

Spark程序的元数据是会往内存中无限存储的。spark.cleaner.ttl来防止OOM，主要出现在SparkSteaming和SharkServer里。

exportSPARK\_JAVA\_OPTS+="-Dspark.kryoserializer.buffer.mb=10-Dspark.cleaner.ttl=43200"

5、ClassNotFound:ClasspathIssues

问题1、缺少jar，不在classpath里。

问题2、jar包冲突，同一个jar不同版本。

解决1：

将所有依赖jar都打入到一个fatJar包里，然后手动设置依赖到指定每台机器的DIR。

valconf=newSparkConf().setAppName(appName).setJars(Seq(System.getProperty("user.dir")+"/target/scala-2.10/sparktest.jar"))

解决2：

把所需要的依赖jar包都放到defaultclasspath里，分发到各个workernode上。

关于性能优化：

第一个是sort-basedshuffle。这个功能大大的减少了超大规模作业在shuffle方面的内存占用量，使得我们可以用更多的内存去排序。

第二个是新的基于Netty的网络模块取代了原有的NIO网络模块。这个新的模块提高了网络传输的性能，并且脱离JVM的GC自己管理内存，降低了GC频率。

第三个是一个独立于Sparkexecutor的externalshuffleservice。这样子executor在GC的时候其他节点还可以通过这个service来抓取shuffle数据，所以网络传输本身不受到GC的影响。

过去一些的参赛系统软件方面的处理都没有能力达到硬件的瓶颈，甚至对硬件的利用率还不到10%。而这次我们的参赛系统在map期间用满了3GB/s的硬盘带宽，达到了这些虚拟机上八块SSD的瓶颈，在reduce期间网络利用率到了1.1GB/s，接近物理极限。

## Spark面试经典系列之数据倾斜：

### 数据倾斜之痛

1、Spark性能真正的杀手

2、数据倾斜之痛

数据倾斜两大直接致命性的后果：

1、OOM，一般OOM都是由于数据倾斜所致

2、速度变慢

数据倾斜基本形态特征：个别Task处理大量数据

数据倾斜的定位：

1、WebUI,可以清晰看见哪些Task运行的数据量大小

2、Log，Log的一个好处是可以清晰的告诉是哪一行出现问题OOM，同时可以清晰的看到在具体哪个Stage出现了数据倾斜（数据倾斜一般会在Shuffle过程中产生的），从而定位具体Shuffle的代码。也有可能发现绝大多数Task非常快，但是个别Task非常慢。

3、代码走读，重点看join、groupByKey、reduceByKey等关键代码

4、对数据特征分布进行分析；

### 解决原理和方法总论

1、Spark数据倾斜解决的原理总论

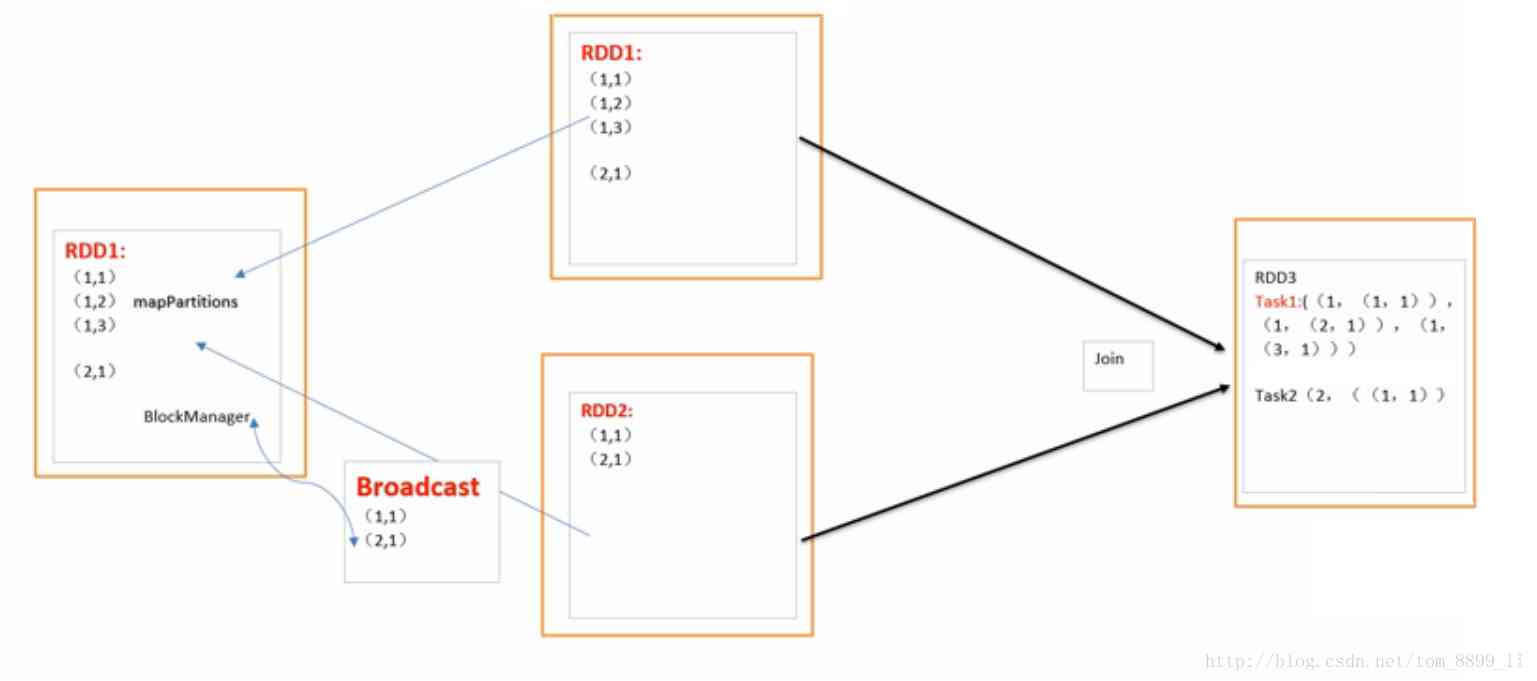
2、Spark数据倾斜解决方法总论

使数据膨胀，tachyon，复用RDD

### Map端Reduce及问题思考

1、Spark数据倾斜解决之Map端Reduce

2、Map端Reduce的问题思考



给小的一段broadcast，然后在大的一端使用mapPartition。

如果数据量太大，有可能引起OOM

### 采样分而治之解决方案

1、采样算法解决数据倾斜的思想

2、采样算法在Spark数据倾斜中的具体操作

某个或某几个Key的Value非常大，从而导致数据倾斜

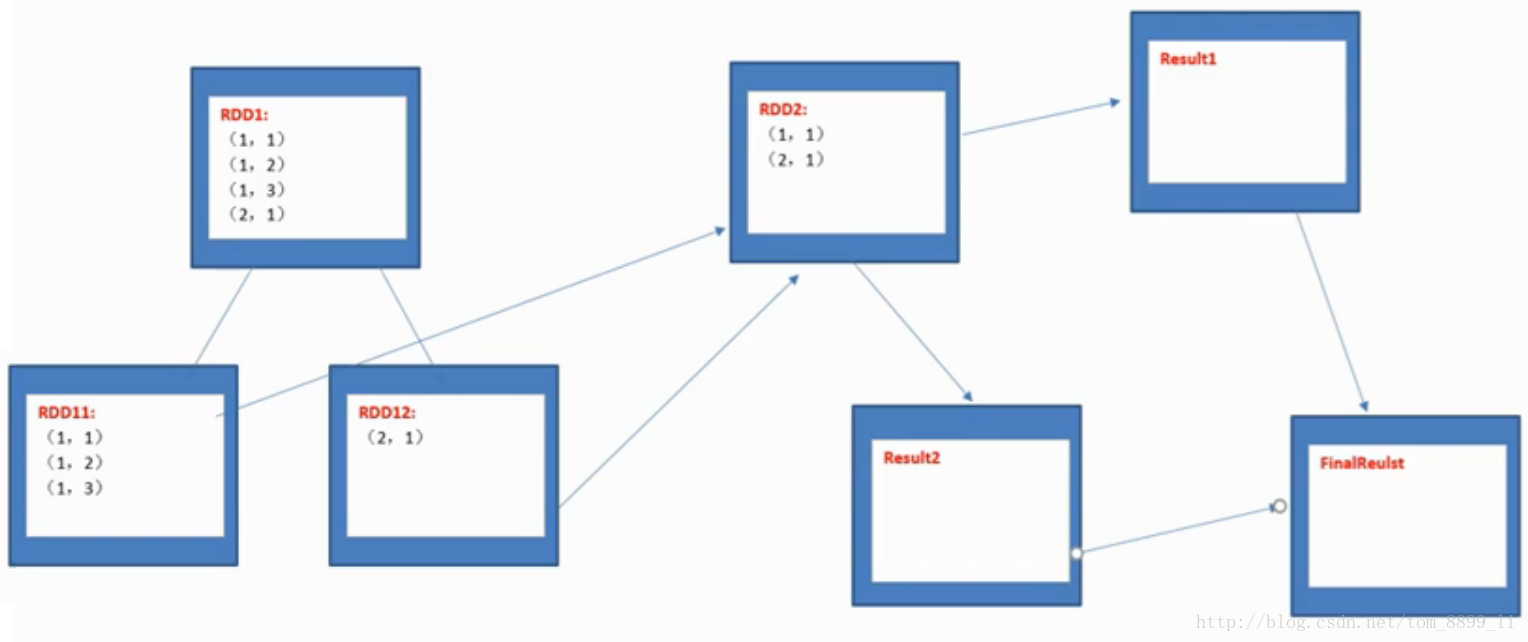
RDD1和RDD2进行join操作，其中我们采用采样的方式发现RDD1中有严重的数据倾斜的Key

第一步:采用SparkRDD中提供的采样接口，我们可以很方便的对全体（例如100亿条数据）进行采样，然后基于采样的数据，我们可以计算出哪个（哪些）Key的Value个数最多；

第二步：把全体数据分成两部分，即把原来RDD1变成RDD11和RDD12，其中RDD11代表导致数据倾斜的Key，RDD12中包含的是不会产生数据倾斜的Key；

第三步：把RDD11和RDD2进行join操作，且把RDD12和RDD2进行join操作，然后把分别join操作后的结果进行Union操作，从而得出和RDD1与RDD2直接进行join操作相同的结果

Spark自己的机制保证的不会产生数据倾斜。



上述流程中：

第一种情况：如果RDD11中的数据量不是很多，可以采用map端的join操作，避免了shuffle和数据倾斜。

第二种情况：如果RDD11中的数据量特别多，此时之所以能够缓解数据倾斜是因为采用了SparkCore天然的并行机制对RDD11中的同样一个Key的数据进行了拆分。从而达到让原本倾斜的Key分散到不同的Task的目的，就缓解了数据倾斜。

思考：在上述过程中如果把倾斜的Key加上随机数，会怎么样？

增加随机数，并行Task数量可能增加，具体是如何操作的？

RDD11中倾斜的Key加上1000以内的随机数，然后和RDD2进行join操作？不行！此时一定需要把RDD11中的Key在RDD2中的相同的Key进行1000以内的随机数，然后再进行join操作，这样做的好处：让倾斜的Key更加不倾斜，在实际生产环境下，会极大的解决在两个进行join的RDD数量都很大且其中一个RDD有一个或者两三个明显倾斜的Key的情况下的数据倾斜问题。

### 对于两个RDD数据量都很大且倾斜的Key特别多如何解决？

1、数据量都很大且倾斜的Key多的情况

2、此种情况下具体操作步骤

两个RDD数据都特别多且倾斜的Key成千上万个，该如何解决数据倾斜的问题？

初步的想法：在倾斜的Key上面加上随机数

该想法的原因：shuffle的时候把key的数据分到不同的task里去

但是现在的倾斜的key非常多，成千上万，所以如果说采样找出倾斜的key的话并不是一个非常好的想法

扩容？

首先，什么是扩容？就是把该RDD中的每一条数据变成5条、10条、20条等，例如RDD中原来是10亿条数据，扩容后可能变成1000亿条数据；

其次，如何做到扩容？flatMap中对要进行扩容的每一条数据都通过0~N-1个不同的前缀变成N条数据（例如变成）

问题：N的值可以随便取吗？需要考虑当前程序能够使用的Core的数量

答案：N的数值一般不能取的太大，通常小于50，否则会对磁盘、内存、网络都会形成极大负担，例如会造成OOM

N这个数值取成10和1000除了OOM等不同以外，是否还有其他影响呢？其实N的数值的大小还会对数据倾斜的解决程度构成直接的影响！N越大，越不容易倾斜，但是也会占用更多的内存、磁盘、网络以及（不必要的）消耗更多的CPU时间

模拟代码:

RDD1joinRDD2

rdd22=RDD2.flatMap{

for(0to9){

0\_item

}

}

rdd11=RDD1.map{

Random(10)

random\_item

}

result=rdd11.join(rdd22)

result.map{

item\_1.split去掉前缀

}

### 并行度的深度使用

1、并行度的初级使用

2、并行度的高级使用

用并行度解决数据倾斜的基本应用：例如reduceByKey

改变并行度之所以能够改善数据倾斜的原因在于，如果某个Task有100个Key且数据量特别大，就极有可能导致OOM或者任务运行特别缓慢，此时如果把并行度变大，则可以分解该Task的数据量，例如把原本该Task的100个Key分解给10个Task，这个就可以减少每个Task的数据量，从而有可能解决OOM和任务慢的问题。

对于reduceByKey而言，你可以传入并行度的参数，也可以自定义Partitioner

增加Executor：改变计算资源，从仅仅数据倾斜的角度来看并不能够直接去解决数据倾斜的问题，但是也有好处，好处是可以同时并发运行更多的Task，结果是可能加快了运行速度。

用并行度解决数据倾斜的高级使用：例如reduceByKey

假设说有倾斜的Key，我们给所有的Key加上一个随机数，然后进行reduceByKey操作；此时同一个Key会有不同的随机数前缀，在进行reduceByKey操作的时候原来的一个非常大的倾斜的Key就分而治之变成若干个更小的Key，不过此时结果和原来不一样，怎么破？进行map操作，目的是把随机数前缀去掉，然后再次进行reduceByKey操作。（当然，如果你很无聊，可以再次做随机数前缀），这样我们就可以把原本倾斜的Key通过分而治之方案分散开来，最后又进行了全局聚合，在这里的本质还是通过改变并行度去解决数据倾斜的问题。

### 解决方案的“银弹”是什么？

1、数据倾斜解决方案总结

2、方案之外的方案

3、数据倾斜解决方案的“银弹”?

逃离Spark技术本身之外如何解决数据倾斜的问题？

之所以会有这样的想法，是因为从结果上来看，数据倾斜的产生来自于数据和数据的处理技术，前面几节课和大家分享都是数据的处理技术层面如何解决数据倾斜，因此，我们现在需要回到数据的层面去解决数据倾斜的问题。

数据本身就是Key-Value的存在方式，所谓的数据倾斜就是说某（几）个Key的Values特别多，所以如果要解决数据倾斜，实质上是解决单一的Key的Values的个数特别多的情况。新的数据倾斜解决方案由此诞生了。

1、把一个大的Key-Values数据分解成为Key-subKey-Values的方式

2、预先和其他的表进行join，将数据倾斜提前到上游的HiveETL

3、可以把大的Key-Values中的Values组拼成为一个字符串，从而形成只有一个元素的Key-Value。

4、加一个中间适配层，当数据进来的时候进行Key的统计和动态排名，基于该排名动态调整Key分布

假如10万个Key都发生了数据倾斜，如何解决呢？此时一般就是加内存和Core

## 使用案例机器学习

### 1.商品特征未读降维：SVD、PCA

### 2.商品挂错页面检查：TF-IDF、SVM、Logistic--Regression

### 3.相关推荐算法模型训练：LoginisticRegression、kmeans、SVM

### 4.商品爆品预测：LoginisticRegression

### 5.关联性分析：FPGrowth

### 6.开发了基于Mllib的机器学习平台

## 经验分享

### 用户常见错误

1.

问题：Collect大量数据到Driver端，导致driveroom；[算法](http://lib.csdn.net/base/datastructure)开发的时候没有注意

解决：driver不能堆积大量数据，尽量不要在driver保存数据

2.

问题：维表数据没用cache内存或者repartition数目太多

解决：将维表数据cache到内存，分区数目不能太多

3.

问题：未对[Spark](http://lib.csdn.net/base/spark)的持久化级别进行选择，需要根据实际的业务需求进行选择

解决：统计RDD的数据量，[大数据](http://lib.csdn.net/base/hadoop)量将Memory\_AND\_DISK作为首选

4.

问题：读写DB没有设置合理的分区数目，并发量太高，影响业务

解决：统计DB的表分区结构，监控DB服务load，压测到位

5.

问题：Spark使用[Hbase](http://lib.csdn.net/base/hbase)scan性能不稳定

解决：Get性能相对稳定，尽量使用Get

6.

问题：Historyserver重启需要回放180G日志，需要4个小时，新完成的app在Historyserver无法立即看到

解决：改为多线程会放SPARK-13988

7.

问题经常回出现classnotfound，但是class文件再包里面存在

解决办法打印classloadder分析，建议不要轻易修改源码classloader

8.

PCA算法只能支持小于14Wfeature特性

解决办法使用SVD进行降维

9.

问题FPGrowth不支持KryoSerializer

解决办法1.6.2之前使用[Java](http://lib.csdn.net/base/java)Serializer

10.

Spark在使用JDBC接口建立DataFrame时，需通过执行SQL来获取该JDBC数据源的Schema，导致创建大量的DataFrame的时候非常耗时

解决办法：Schema相同的table可以不用重复获取schema

地址：<https://github.com/ouyangshourui/SparkJDBCSchema/wiki>

4000个DataFrame的初始化时间从原先的25分钟缩短为10分钟以内

### Spark平台权限

1.4.0Standalonecluster模式不支持多用户

相关组件读写权限问题无法解决，比如读取[Hive](http://lib.csdn.net/base/hive)、[hbase](http://lib.csdn.net/base/hbase)、HDFS数据的权限问题

解决办法：修改SparkContextsparkuser和systemusername

代码地址：<https://github.com/ouyangshourui/ouyangshourui/StandalongClusterAuthorization/wiki>

SparkSql[hive](http://lib.csdn.net/base/hive)元数据密码加密，javax.jdo.option.ConnectionPassword暴露给用户比较危险

解决办法：修改HiveContext.[Scala](http://lib.csdn.net/base/scala)文件中的metadataHive变量，选择自定义的解密算法解密

代码地址：<https://github.com/ouyangshourui/HivePasswordEncryptionDecryption/wiki>

Spark1.5.2Sql放大了Hive读权限，任何用户都可以读取别的用户的Hive表数据

临时解决办法，生成HiveTableScanoperator时调用driver已有的HiveClient权限接口检查当前用户的读权限

<https://github.com/ouyangshourui/HiveReadpermission/wiki>

### 升级遇到的问题

1.4.0Standalone升级到1.5.2onYarn

用户代码使用system.exit(-1)RMwebUI却显示正常，建议直接hrowexception

* 自定义封装[MySQL](http://lib.csdn.net/base/mysql)PostgreSqlJDBC没有考虑driverJDBCDialect的实现导致数据无法返回。
* 每个Executor都与Hive简历connection去获取HiveConf没有broadcastHiveConf（SPARK-10679）

多版本SparkDynamicPersourceAllocation无法共存，DRA需重启YarnNodeManger，耦合性太强（没有解决）

### TODO

SparkStreaming全面落地，吸收ApacheBean思想

SparkSql替代大部分Hive任务

SPark现有任务优化加速

完善[机器学习](http://lib.csdn.net/base/machinelearning)平台覆盖大部分电商和金融领域机器学习算法库

全面拥抱Spark2.0参与社区

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

前K个user