1、简要描述如何安装配置apache的一个开源hadoop，只描述即可，无需列出具体步骤，列出具体步骤更好。

1使用root账户登录

2 修改IP

3 修改host主机名

4 配置SSH免密码登录

5 关闭防火墙

6 安装JDK

6 解压hadoop安装包

7 配置hadoop的核心文件 hadoop-env.sh，core-site.xml , mapred-site.xml ， hdfs-site.xml

8 配置hadoop环境变量

9 格式化 hadoop namenode-format

10 启动节点start-all.sh

2、 请列出正常的hadoop集群中hadoop都分别需要启动 哪些进程，他们的作用分别都是什么，请尽量列的详细一些。

--namenode =>HDFS的守护进程，负责维护整个文件系统，存储着整个文件系统的元数据信息，有image+edit log namenode不会持久化存储这些数据，而是在启动时重建这些数据。

--datanode  =>是具体文件系统的工作节点，当我们需要某个数据，namenode告诉我们去哪里找，就直接和那个DataNode对应的服务器的后台进程进行通信，由DataNode进行数据的检索，然后进行具体的读/写操作

--secondarynamenode  =>一个冗余的守护进程，相当于一个namenode的元数据的备份机制，定期的更新，和namenode进行通信，将namenode上的image和edits进行合并，可以作为namenode的备份使用

--resourcemanager =>是yarn平台的守护进程，负责所有资源的分配与调度，client的请求由此负责，监控nodemanager

--nodemanager  => 是单个节点的资源管理，执行来自resourcemanager的具体任务和命令

————————————————

3、启动hadoop报如下错误，该如何解决？

    --1.error  org.apache.hadoop.hdfs.server.namenode.NameNode

        --找不到主类，应该是配置文件的hadoop的安装位置配置错误，对hadoop-env.sh文件进行检查修改

    --2.org.apache.hadoop.hdfs.server.common.inconsistentFSStateException

        --这个是存储目录不存在，或者被删除，对namenode进行格式化，或重新格式化，对tmp.dir进行自己的设置

    --3.Directory /tmp/hadoop-root/dfs/name is in an inconsistent

        --这个和上面一样的，重新设置core-site.xml中hadoop.tmp.dir的值，对namenode进行格式化，

    --4.state storage direction does not exist or is not accessible?

        --之前是默认的tmp目录，每次重启都会清除这个数据，所以找不到整个文件系统的信息，重新设置core-site.xml中hadoop.tmp.dir的值，对namenode进行格式化，

4、请写出以下的shell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节点需要执行的命令

答：（1）hadoop job –list 得到job的id，然后执 行 hadoop job -kill jobId就可以杀死一个指定jobId的job工作了。

（2）hadoopfs -rmr /tmp/aaa

(3) 增加一个新的节点在新的几点上执行

Hadoop daemon.sh start datanode

Hadooop daemon.sh start tasktracker/nodemanager

下线时，要在conf目录下的excludes文件中列出要下线的datanode机器主机名

然后在主节点中执行 hadoop dfsadmin -refreshnodes à下线一个datanode

删除一个节点的时候，只需要在主节点执行

hadoop mradmin -refreshnodes ---à下线一个tasktracker/nodemanager

4、请列出你所知道的hadoop调度器，并简要说明其工作方法

答：Fifo schedular :默认，先进先出的原则

Capacity schedular :计算能力调度器，选择占用最小、优先级高的先执行，依此类推。

Fair schedular:公平调度，所有的 job 具有相同的资源。

5、 当前日志采样格式为

a , b , c , d

b , b , f , e

a , a , c , f

请你用最熟悉的语言编写mapreduce，计算第四列每个元素出现的个数

答：

public classWordCount1 {

public static final String INPUT\_PATH ="hdfs://hadoop0:9000/in";

public static final String OUT\_PATH ="hdfs://hadoop0:9000/out";

public static void main(String[] args)throws Exception {

Configuration conf = newConfiguration();

FileSystem fileSystem =FileSystem.get(conf);

if(fileSystem.exists(newPath(OUT\_PATH))){}

fileSystem.delete(newPath(OUT\_PATH),true);

Job job = newJob(conf,WordCount1.class.getSimpleName());

//1.0读取文件，解析成key,value对

FileInputFormat.setInputPaths(job,newPath(INPUT\_PATH));

//2.0写上自己的逻辑，对输入的可以，value进行处理，转换成新的key,value对进行输出

job.setMapperClass(MyMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(LongWritable.class);

//3.0对输出后的数据进行分区

//4.0对分区后的数据进行排序，分组，相同key的value放到一个集合中

//5.0对分组后的数据进行规约

//6.0对通过网络将map输出的数据拷贝到reduce节点

//7.0 写上自己的reduce函数逻辑，对map输出的数据进行处理

job.setReducerClass(MyReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

FileOutputFormat.setOutputPath(job,new Path(OUT\_PATH));

job.waitForCompletion(true);

}

static class MyMapper extendsMapper<LongWritable, Text, Text, LongWritable>{

@Override

protected void map(LongWritablek1, Text v1,

org.apache.hadoop.mapreduce.Mapper.Contextcontext)

throws IOException,InterruptedException {

String[] split =v1.toString().split("\t");

for(String words :split){

context.write(split[3],1);

}

}

}

static class MyReducer extends Reducer<Text,LongWritable, Text, LongWritable>{

protected void reduce(Text k2,Iterable<LongWritable> v2,

org.apache.hadoop.mapreduce.Reducer.Contextcontext)

throws IOException,InterruptedException {

Long count = 0L;

for(LongWritable time :v2){

count += time.get();

}

context.write(v2, newLongWritable(count));

}

}

}

1. 请简述hadoop怎样实现二级排序（就是对key和value双排序）

  --在MapReduce中本身就会对我们key进行排序，所以我们要对value进行排序，主要思想为将key和部分value拼接成一个组合key（实现WritableComparable接口或者调用 setSortComparatorClass函数），这样reduce获取的结果便是先按key排序，后按value排序的结果，在这个方法中，用户需 要自己实现Paritioner，继承Partitioner<>,以便只按照key进行数据划分。Hadoop显式的支持二次排序，在Configuration类中有个 setGroupingComparatorClass()方法，可用于设置排序group的key值。

————————————————

7、简述hadoop实现jion的几种方法

**(1)、reduce side join**  
reduce side join是一种最简单的join方式，其主要思想如下：  
在map阶段，map函数同时读取两个文件File1和File2，为了区分两种来源的key/value数据对，对每条数据打一个标签（tag）,比如：tag=0表示来自文件File1，tag=2表示来自文件File2。即：map阶段的主要任务是对不同文件中的数据打标签。  
在reduce阶段，reduce函数获取key相同的来自File1和File2文件的value list， 然后对于同一个key，对File1和File2中的数据进行join（笛卡尔乘积）。即：reduce阶段进行实际的连接操作。

**(2)、map side join**  
之所以存在reduce side join，是因为在map阶段不能获取所有需要的join字段，即：同一个key对应的字段可能位于不同map中。Reduce side join是非常低效的，因为shuffle阶段要进行大量的数据传输。  
Map side join是针对以下场景进行的优化：两个待连接表中，有一个表非常大，而另一个表非常小，以至于小表可以直接存放到内存中。这样，我们可以将小表复制多份，让每个map task内存中存在一份（比如存放到hash table中），然后只扫描大表：对于大表中的每一条记录key/value，在hash table中查找是否有相同的key的记录，如果有，则连接后输出即可。  
为了支持文件的复制，Hadoop提供了一个类DistributedCache，使用该类的方法如下：  
（1）用户使用静态方法DistributedCache.addCacheFile()指定要复制的文件，它的参数是文件的URI（如果是HDFS上的文件，可以这样：[hdfs://namenode:9000/home/XXX/file](https://links.jianshu.com/go?to=hdfs://namenode:9000/home/XXX/file" \t "/Users/huangzhenjia/Documents\\x/_blank)，其中9000是自己配置的NameNode端口号）。JobTracker在作业启动之前会获取这个URI列表，并将相应的文件拷贝到各个TaskTracker的本地磁盘上。（2）用户使用DistributedCache.getLocalCacheFiles()方法获取文件目录，并使用标准的文件读写API读取相应的文件。

**(3)、SemiJoin**  
SemiJoin，也叫半连接，是从分布式数据库中借鉴过来的方法。它的产生动机是：对于reduce side join，跨机器的数据传输量非常大，这成了join操作的一个瓶颈，如果能够在map端过滤掉不会参加join操作的数据，则可以大大节省网络IO。  
实现方法很简单：选取一个小表，假设是File1，将其参与join的key抽取出来，保存到文件File3中，File3文件一般很小，可以放到内存中。在map阶段，使用DistributedCache将File3复制到各个TaskTracker上，然后将File2中不在File3中的key对应的记录过滤掉，剩下的reduce阶段的工作与reduce side join相同。

(4)、reduce side join + BloomFilter  
在某些情况下，SemiJoin抽取出来的小表的key集合在内存中仍然存放不下，这时候可以使用BloomFiler以节省空间。  
BloomFilter最常见的作用是：判断某个元素是否在一个集合里面。它最重要的两个方法是：add() 和contains()。最大的特点是不会存在false negative，即：如果contains()返回false，则该元素一定不在集合中，但会存在一定的true negative，即：如果contains()返回true，则该元素可能在集合中。  
因而可将小表中的key保存到BloomFilter中，在map阶段过滤大表，可能有一些不在小表中的记录没有过滤掉（但是在小表中的记录一定不会过滤掉），这没关系，只不过增加了少量的网络IO而已。

8、 请简述mapreduce中的combine和partition的作用

答：combiner是发生在map的最后一个阶段，其原理也是一个小型的reducer，主要作用是减少输出到reduce的数据量，缓解网络传输瓶颈，提高reducer的执行效率。

partition的主要作用将map阶段产生的所有kv对分配给不同的reducer task处理，可以将reduce阶段的处理负载进行分摊

9、 用mapreduce怎么处理数据倾斜问题

本质：让各分区的数据分布均匀

可以根据业务特点，设置合适的partition策略

如果事先根本不知道数据的分布规律，利用随机抽样器抽样后生成partition策略再处理

10、 hadoop框架怎么来优化

答：

可以从很多方面来进行：比如hdfs怎么优化，mapreduce程序怎么优化，yarn的job调度怎么优化，hbase优化，hive优化。。。。。。。

11、 hadoop中常用的数据压缩算法

答：

Lzo

Gzip

Default

Snapyy

如果要对数据进行压缩，最好是将原始数据转为SequenceFile 或者 Parquet File（spark）

12、MapReduce优化经验

答：(1.)设置合理的map和reduce的个数。合理设置blocksize

(2.)避免出现数据倾斜

(3.combine函数

(4.对数据进行压缩

(5.小文件处理优化：事先合并成大文件，combineTextInputformat，在hdfs上用mapreduce将小文件合并成SequenceFile大文件（key:文件名，value：文件内容）

(6.参数优化

13、mapreduce的大致流程

答：主要分为八个步骤

1/对文件进行切片规划

2/启动相应数量的maptask进程

3/调用FileInputFormat中的RecordReader，读一行数据并封装为k1v1

4/调用自定义的map函数，并将k1v1传给map

5/收集map的输出，进行分区和排序

6/reduce task任务启动，并从map端拉取数据

7/reduce task调用自定义的reduce函数进行处理

8/调用outputformat的recordwriter将结果数据输出

14、 谈谈 hadoop1 和 hadoop2 的区别

答：

hadoop1的主要结构是由HDFS和mapreduce组成的，HDFS主要是用来存储数据，mapreduce主要是用来计算的，那么HDFS的数据是由namenode来存储元数据信息，datanode来存储数据的。Jobtracker接收用户的操作请求之后去分配资源执行task任务。

在hadoop2中，首先避免了namenode单点故障的问题，使用两个namenode来组成namenode feduration的机构，两个namenode使用相同的命名空间，一个是standby状态，一个是active状态。用户访问的时候，访问standby状态，并且，使用journalnode来存储数据的原信息，一个namenode负责读取journalnode中的数据，一个namenode负责写入journalnode中的数据，这个平台组成了hadoop的HA就是high availableAbility高可靠。

然后在hadoop2中没有了jobtracker的概念了，统一的使用yarn平台来管理和调度资源，yarn平台是由resourceManager和NodeManager来共同组成的，ResourceManager来接收用户的操作请求之后，去NodeManager上面启动一个主线程负责资源分配的工作，然后分配好了资源之后告知ResourceManager，然后ResourceManager去对应的机器上面执行task任务。

67. 文件大小默认为 64M，改为 128M 有啥影响？

答：更改文件的block块大小，需要根据我们的实际生产中来更改block的大小，如果block定义的太小，大的文件都会被切分成太多的小文件，减慢用户上传效率，如果block定义的太大，那么太多的小文件可能都会存到一个block块中，虽然不浪费硬盘资源，可是还是会增加namenode的管理内存压力。

15、Hadoop中job和tasks之间的区别是什么？

Job是我们对一个完整的mapreduce程序的抽象封装

Task是job运行时，每一个处理阶段的具体实例，如map task，reduce task，maptask和reduce task都会有多个并发运行的实例

16、hadoop中通过拆分任务到多个节点运行来实现并行计算，但某些节点运行较慢会拖慢整个任务的运行，hadoop采用全程机制应对这个情况？

Speculate 推测执行

17、有可能使hadoop任务输出到多个目录中吗？如果可以，怎么做？

自定义outputformat或者用multioutputs工具

18、如何为一个hadoop任务设置mappers的数量？

Split机制

19、如何为一个hadoop任务设置要创建reduder的数量？

可以通过代码设置

具体设置多少个，应该根据硬件配置和业务处理的类型来决定

20、HDFS数据写入实现机制

        --写入HDFS过程：

            1、根namenode通信请求上传文件，namenode检查目标文件是否已存在，父目录是否存在

            2、namenode返回是否可以上传

            3、client会先对文件进行切分，比如一个blok块128m，文件有300m就会被切分成3个块，一个128M、一个128M、一个44M请求第一个 block该传输到哪些datanode服务器上

            4、namenode返回datanode的服务器

            5、client请求一台datanode上传数据（本质上是一个RPC调用，建立pipeline），第一个datanode收到请求会继续调用第二个datanode，然后第二个调用第三个datanode，将整个pipeline建立完成，逐级返回客户端

            6、client开始往A上传第一个block（先从磁盘读取数据放到一个本地内存缓存），以packet为单位（一个packet为64kb），当然在写入的时候datanode会进行数据校验，它并不是通过一个packet进行一次校验而是以chunk为单位进行校验（512byte），第一台datanode收到一个packet就会传给第二台，第二台传给第三台；第一台每传一个packet会放入一个应答队列等待应答

            7、当一个block传输完成之后，client再次请求namenode上传第二个block的服务器。

        --读取文件过程：

            使用HDFS提供的客户端开发库Client，向远程的Namenode发起RPC请求；Namenode会视情况返回文件的部分或全部block列表，对于每个block，Namenode都会返回有该block拷贝的DataNode地址；客户端开发库Client会选取离客户端最接近的DataNode来读取block；如果客户端本身就是DataNode,那么将从本地直接获取数据.读取完当前block的数据后，关闭与当前的DataNode连接，并为读取下一个block寻找最佳的DataNode；当读完列表的block后，且文件读取还没有结束，客户端开发库会继续向Namenode获取下一批的block列表。读取完一个block都会进行 checksum 验证，如果读取 datanode 时出现错误，客户端会通知 Namenode，然后再从下一个拥有该 block 拷贝的 datanode 继续读。

21、hadoop节点的动态上线下线的大概操作

        --节点上线

            1.关闭新增节点的防火墙

            2.在 NameNode节点的hosts文件中加入新增数据节点的hostname

            3.在每个新增数据节点的hosts文件中加入NameNode的hostname

            4.在NameNode节点上增加新增节点的SSH免密码登录的操作

            5.在NameNode节点上的dfs.hosts中追加上新增节点的hostname,

            6.在其他节点上执行刷新操作：hdfs dfsadmin -refreshNodes

            7.在 NameNode 节点上，更改slaves文件，将要上线的数据节点hostname追加

            到slaves文件中

            8.启动DataNode节点

            9.查看NameNode的监控页面看是否有新增加的节点

       --节点下线

            1.修改/conf/hdfs-site.xml文件

            2.确定需要下线的机器，dfs.osts.exclude文件中配置好需要下架的机器，这个是阻

            止下架的机器去连接NameNode

            3.配置完成之后进行配置的刷新操作./bin/hadoop dfsadmin -refreshNodes,这个

            操作的作用是在后台进行block块的移动

            4.当执行三的命令完成之后，需要下架的机器就可以关闭了，可以查看现在集

            群上连接的节点，正在执行 Decommission，会显示：

            Decommission Status : Decommission in progress 执行完毕后，会显示：

            Decommission Status : Decommissioned

            5.机器下线完毕，将他们从 excludes 文件中移除。

# 22、[hadoop常用端口号](https://www.cnblogs.com/guoyu1/p/12218484.html)

dfs.namenode.http-address:50070  
 SecondaryNameNode 辅助名称节点端口号：50090  
 dfs.datanode.address:50010  
 fs.defaultFS:8020 或者 9000  
 yarn.resourcemanager.webapp.address:8088  
 历史服务器 web访问端口：19888

windows中用浏览器访问namenode提供的web端口：    http://hdp-01:50070

在windows上用浏览器访问resourcemanager的web端口：[http://hdp-04:8088](http://hdp-04:8088/)

**23、简答说一下hadoop的map-reduce编程模型**

首先map task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

reduce task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

**24、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

**25、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map task和reduce task，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

**26、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

**27、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

**28、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

29、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