**学习心得**

18183023 谭智元 6.24

**今天学习的内容：**

1.成功运行WordCount示例：

在经过昨天的报错之后，仔细排查mapred-site.xml文件，仔细比对之后发现了疑似非法字符，删除后所有节点均能正常运行。

在运行WordCount的示例时，出现了了一直卡在running job的问题。在咨询了小组内同成员之后，将其遇到相同问题的解决方案直接应用，修改mapred-site.xml的部分信息，终于能够成功运行得到结果。

至此，Hadoop的单机安装配置大功告成！！！

2. Hadoop分布式文件系统：

1）概述：

HDFS，是Hadoop Distributed File System的简称，是Hadoop抽象文件系统的一种实现。Hadoop抽象文件系统可以与本地系统、Amazon S3等集成，甚至可以通过Web协议（webhsfs）来操作。HDFS的文件分布在集群机器上，同时提供副本进行容错及可靠性保证。例如客户端写入读取文件的直接操作都是分布在集群各个机器上的，没有单点性能压力

2）HDFS设计原则：

设计目标：存储非常大的文件（GB、TB、PB）、采用流式的数据访问方式（一次写入、多次读取）、运行于商业硬件上（Hadoop不需要特别贵的机器，可运行于普通商用机器）

不适合的应用类型：低延时的数据访问、大量小文件、多方读写，需要任意的文件修改

3）HDFS核心概念：

i)Blocks：

物理磁盘中有块的概念，磁盘的物理Block是磁盘操作最小的单元，读写操作均以Block为最小单元，一般为512 Byte。文件系统在物理Block之上抽象了另一层概念，文件系统Block物理磁盘Block的整数倍。通常为几KB。Hadoop提供的df、fsck这类运维工具都是在文件系统的Block级别上进行操作。

HDFS的Block块比一般单机文件系统大得多，默认为128M。HDFS的文件被拆分成block-sized的chunk，chunk作为独立单元存储。比Block小的文件不会占用整个Block，只会占据实际大小。

ii)NameNode：

Namenode存放文件系统树及所有文件、目录的元数据。元数据持久化为2种形式：namespcae image-和edit log。

在HDFS中，Namenode可能成为集群的单点故障，Namenode不可用时，整个文件系统是不可用的。HDFS针对单点故障提供了2种解决机制：备份持久化元数据（将文件系统的元数据同时写到多个文件系统）、Secondary Namenode（节点定期合并主Namenode的namespace image和edit log，通过创建检查点checkpoint来合并。其也可在主NameNode崩溃后替代原节点）

iii)DataNode：

数据节点负责存储和提取Block，读写请求可能来自namenode，也可能直接来自客户端。数据节点周期性向Namenode汇报自己节点上所存储的Block相关信息。数据节点也会定期向NameNode节点发送“心跳”信息，以确保当前数据节点可用且存活

iv) Block Caching：

DataNode通常直接从磁盘读取数据，但是频繁使用的Block可以在内存中缓存。默认情况下，一个Block只有一个数据节点会缓存

v) HDFS Federation：

HDFS Federation提供了一种横向扩展NameNode的方式。在Federation模式中，每个NameNode管理命名空间的一部分，例如一个NameNode管理/user目录下的文件， 另一个NameNode管理/share目录下的文件

vi) Hadoop的HA方案：

采用HA的HDFS集群配置两个NameNode，分别处于Active和Standby状态。当Active NameNode故障之后，Standby接过责任继续提供服务，用户没有明显的中断感觉。一般耗时在几十秒到数分钟。

HA涉及到的主要实现逻辑：主备需共享edit log存储、DataNode需要同时往主备发送Block Report、客户端需要配置failover模式（对用户透明）、Standby替代Secondary NameNode

4）命令行接口：

i)一些常用命令如下：

hadoop fs -copyFromLocal // 从本地复制文件到HDFS

hadoop fs mkdir // 创建目录

hadoop fs -ls // 列出文件列表

ii)Hadoop中的文件和目录的权限如下：

读权限（r）：用于读取文件或者列出目录中的内容

写权限（w）：对于文件，就是文件的写权限。目录的写权限指在该目录下创建或者删除文件（目录）的权限。

执行权限（x）：文件没有所谓的执行权限，被忽略。对于目录，执行权限用于访问器目录下的内容。

iii)Hadoop中的用户和组：

每个文件或目录都有owner，group，mode三个属性，owner指文件的所有者，group为权限组。mode由所有者权限、文件所属的组中组员的权限、非所有者非组员的权限组成。

（参考：https://blog.csdn.net/bingduanlbd/article/details/51914550）

3.了解 Hadoop I/O系统：

1）序列化与反序列化：

序列化和反序列化在分布式数据处理里主要出现在进程间通行和永久存储两个应用领域。在Hadoop 系统中，系统中多个节点上的进程间通信是通过远程过程调用（romote procedure call,R PC)实现的，RPC协议将消息序列转化为二进制流后发送到远程节点，远程节点接着将二进制流反序列化为消息，所以RPC对于序列化有以下要求：

紧凑（紧凑的格式可以提高传输效率，充分利用网络带宽）；

快速（减少序列化和反序列化的开销，提高整个分布式系统的性能）；

可扩展（在方法调用的过程中增加新的参数，或者新的服务器系统要能够接受老客户端的旧的格式的消息）；

互操作（可以支持不同语言C++、python、java等写的客户端与服务器交互）

2）Hadoop的序列化格式——Writabale：

Hadoop 定义了自己的序列化格式Writable，它是通过Java语言实现的，格式紧凑速度快。Writable是Hadoop的核心，很多MapReduce程序都会为键和值使用它。

几个常用的Writable类：

i)Text类：Text类是针对UTF-8序列（UTF-8是UNICODE的一种变长字节字符编码，又称万国码）的Writable类。Text的索引是按编码后字节序列中的位置来实现的，因此进行位置索引时必须先将Text 对象转化为java.nio.BytesBuffer对象，然后利用缓冲区对Text对象反复调用bytesToCodePoint()静态方法，以获取下一代码的位置。在一些情况下为了方便处理，需要将Text类转化为String类（ToString）来使用字符串操作丰富的API

ii)ByteWritable类，二进制数据数组的封装，它的序列化格式为一个用于指定后面数据字节数的整数域（4字节），后跟字节本身。

iii)NullWritabe，是Writable的一个特殊类型，它的序列化长度为0，类似于null。

iv)ObjectWritable类，是对java基本类型（String,enum，Writable，null或这些类型组成的数组）的一个通用封装。

v)Writable集合类，共有4个集合类，其中ArrayWritable是对Writable的数组的实现，TwoDArrayWritable 是对Writable的二维数组的实现，MapWritable是对Map的实现，SortedMapWritable是对SortedMap的实现。

vi)定制的Writable类，我们可以根据自己的需求构造自己的Writable类，可以根据需要控制二进制表示和排序顺序，由于Writable是MapReduce数据路径的核心，所以调整二进制表示能对性能残生显著效果。

3）Hadoop的序列化框架——Avro：

Avro是一个数据序列化系统，设计用于支持大 批量数据交换的应用。它的主要特点有：支持二进制序列化方式，可以便捷，快速地处理大量数据；动态语言友好，Avro提供的机制使动态语言可以方便地处理 Avro数据。

Avro依赖模式(Schema)来实现数据结构定义。可以把模式理解为Java的类，它定义每个实例的结构，可以包含哪些属性。可以根据类来产生任意多个实例对象。对实例序列化操作时必须需要知道它的基本结构，也就需要参考类的信息。这里，根据模式产生的Avro对象类似于类的实例对象。Avro更适合于脚本语言的发挥。

Avro支持两种序列化编码方式：二进制编码和JSON编码。使用二进制编码会高效序列化，并且序列化后得到的结果会比较小；而JSON一般用于调试系统或是基于WEB的应用。

4）SequenceFile和MapFile：

Hadoop的HDFS和MapReduce子框架主要是针对大数据文件来设计的，在小文件的处理上不但效率低下，而且十分消耗内存资源(每一个小文件占 用一个Block,每一个block的元数据都存储在namenode的内存里)。解决办法通常是选择一个容器，将这些小文件组织起来统一存储。HDFS 提供了两种类型的容器，分别是SequenceFile和MapFile。

5）数据完整性：

检测数据是否损坏的常用措施是：在数据第一次引入系统时计算校验文件和在数据通过一个不可靠的通道进行传输时再一次生成的计算校验文件比较，这样就能发现数据是否损坏。如果计算所得的新校验和原来的校验不匹配，那么表明数据已经损坏。

注意：该技术并不能修复数据，它只能检测出数据错误。（校验和数据也可能损坏，但是由于校验和文件小，所以损坏的可能性小）

Hadoop的LocalFileSystem执行客户端的校验和验证。当在你写入一个filename的文件时，文件系统客户端会明确地在包含每一个文件校验和的同一个　　　　目录内新建一个名为.filename.crc的隐藏文件。通俗来说，当你把文件上传到HDFS集群中时，若要对文件在本地进行CRC校验就使用LocalFileSystem，不想对文件进行校验时就是用RawFileSystem。

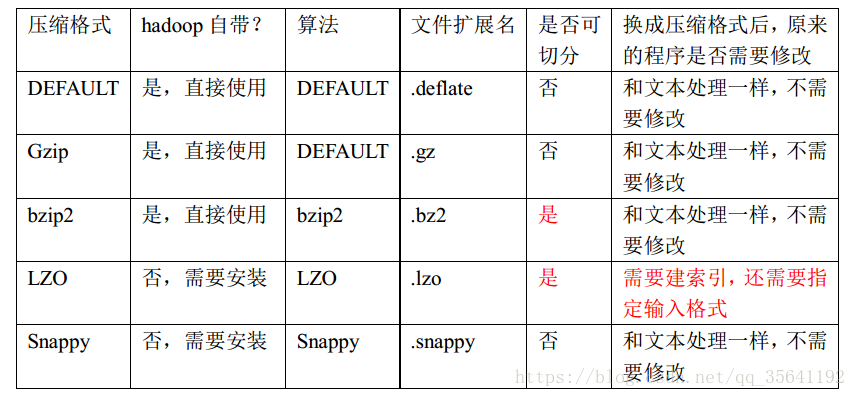
6）数据压缩：

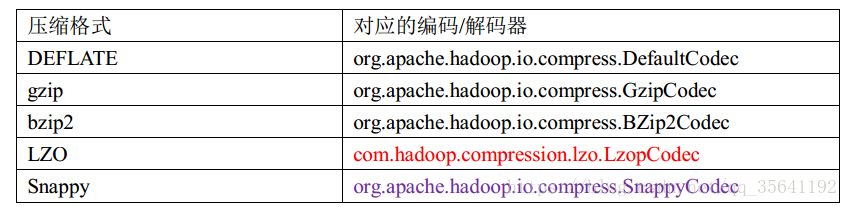
在处理大量数据的时候，文件压缩有两大好处：减少存储文件所需要的磁盘空间和加速数据在网络和磁盘上的传输。

因此，对于任何大容量的分布式存储系统而言，文件压缩是必须的。

比较常见压缩格式及其特点如下。其中，gzip对时间/空间的平衡做的最好；bzip2的压缩效率比gzip高，但比gzip慢；LZO的速度最快，但压缩效果比较差。

在Hadoop中，codec是压缩-解压算法的一种实现，一个对CompressionCodec接口的实现代表一个codec，例如GzipCodec包装了gzip的压缩和解压缩算法。





（参考：<https://blog.csdn.net/qq_35641192/article/details/80490557>）

（参考：http://my.oschina.net/u/872123/blog/120001）

4）了解MapReduce：

Hadoop MapReduce是一个**软件框架**，基于该框架能够容易地编写应用程序，这些应用程序能够运行在由上千个商用机器组成的**大集群**上，并以一种**可靠的**，具有**容错能力**的方式**并行地处理**上TB级别的**海量数据集**。

MapReduce的思想就是“**分而治之**”：

i)Mapper负责“分”，即把复杂的任务分解为若干个“简单的任务”来处理。“简单的任务”包含三层含义：

一是数据或计算的规模相对原任务要大大缩小；二是就近计算原则，即任务会分配到存放着所需数据的节点上进行计算；三是这些小任务可以并行计算，彼此间几乎没有依赖关系。

ii)Reducer负责对map阶段的结果进行汇总。至于需要多少个Reducer，用户可以根据具体问题，通过在mapred-site.xml配置文件里设置参数mapred.reduce.tasks的值，缺省值为1。

MapReduce的整个工作过程包含如下4个独立的实体：

实体一：客户端，用来提交MapReduce作业。

实体二：JobTracker，用来协调作业的运行。

实体三：TaskTracker，用来处理作业划分后的任务。

实体四：HDFS，用来在其它实体间共享作业文件。

一个MapReduce作业通常会把输入的数据集切分为若干独立的数据块，由Map任务以完全并行的方式去处理它们。框架会对Map的输出先进行排序，然后把结果输入给Reduce任务。通常作业的输入和输出都会被存储在文件系统中，整个框架负责任务的调度和监控，以及重新执行已经关闭的任务。

**MapReduce框架的组成**包含以下两个部分：

i)JobTracker：

JobTracker负责调度构成一个作业的所有任务，这些任务分布在不同的TaskTracker上。可以将其理解为公司的项目经理，项目经理接受项目需求，并划分具体的任务给下面的开发工程师。

ii)TaskTracker：

TaskTracker负责执行由JobTracker指派的任务，这里我们就可以将其理解为开发工程师，完成项目经理安排的开发任务即可。

**MapReduce的输入输出**：MapReduce框架运转在<key,value>键值对上，也就是说，框架把作业的输入看成是一组<key,value>键值对，同样也产生一组<key,value>键值对作为作业的输出，这两组键值对有可能是不同的。

**反思：**

1.今天终于完成了单机Hadoop环境的配置，成功运行了WordCount示例，成就感十足

2.对于昨天的反思还应注意：在创建文件或者调用进程时，一定要注意当前用户和所属组，在成为root用户进行完某些操作后应立马切换回来，以保证前后用户的一致性，避免产生权限之类的问题导致失败

3.今天对于上述知识的学习只停留在理论部分，明天应通过实例的方式进一步加深对其理解