Hadoop3.0新特性

基于Hadoop 3.0.0-alpha3版本

牛海胜

1、要求jdk8以上版本

All Hadoop JARs are now compiled targeting a runtime version of Java 8. Users still using Java 7 or below must upgrade to Java 8

2、HDFS支持 Erasure Encoding (纠删码)

Erasure coding

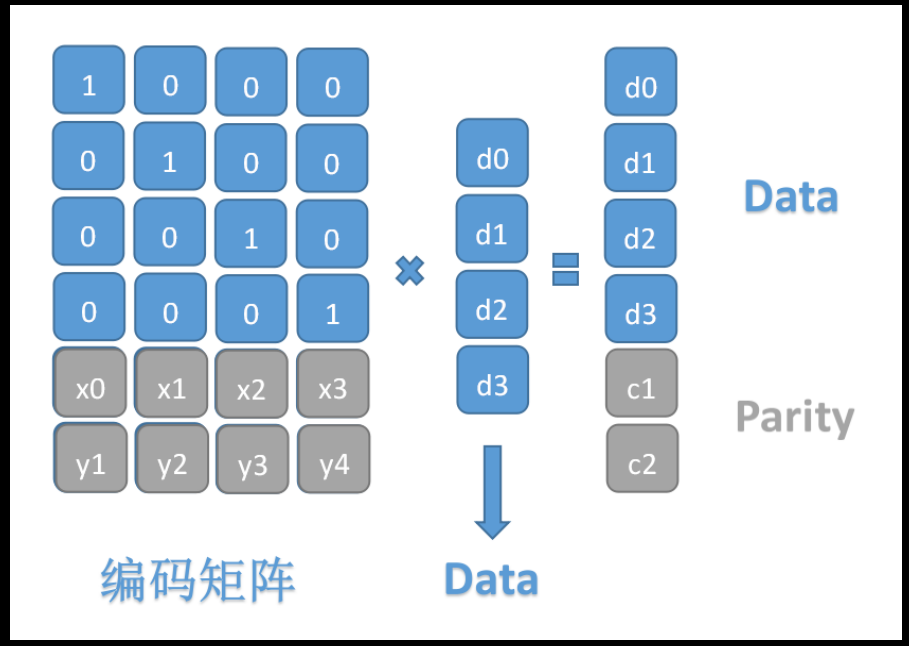
纠删码是一种编码容错技术。通过在原始数据中加入新的校验数据，使得各个部分的数据产生关联性。在一定范围的数据出错情况下，通过纠删码技术都可以进行恢复。

通常在存储系统中使用的纠删码可分为两类:

* XOR 码编、解码只需要按位异或即可完成，速度较快；
* RS 码编、解码需要有限域上的运算，速度慢于XOR码，更具一般性。

Erasure coding 实现

原始数据与校验数据建立关系：矩阵乘法

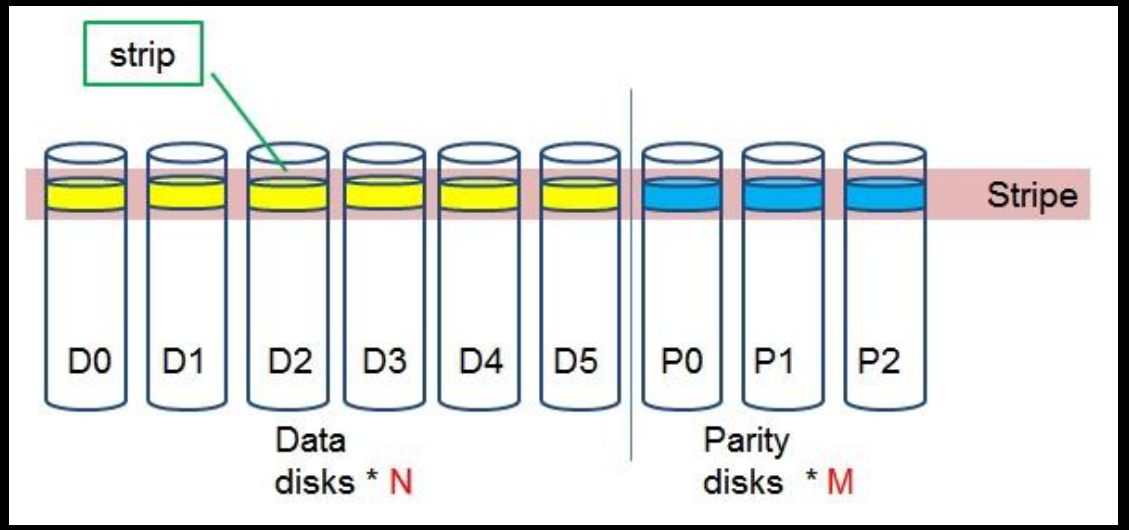


用编码矩阵和分块数据做乘法从而得到校验块。在建立完关联后，由于矩阵运算是可逆，系统就具备了容忍最大校验块个数失效的能力。

Erasure coding 在存储中的结构

我们把纠删码引入存储内部时，它需要一个特殊的组织。从图中可以看出，一个条带（Stripe）是由多个数据块（strip）构成，分为数据块和校验块。数据和校验的比例按N+M可调整，典型的如12+4，6+3等等。

* 连续布局：数据被依次写入一个块中，一个块写满之后再写入下一个块。
* 条形布局：条（stripe）是由若干个相同大小单元（cell）构成的序列。在条形布局下，数据被依次写入条的各个单元中，当条被写满之后就写入下一个条，一个条的不同单元位于不同的数据块中。



Hadoop Erasure Coding 的实现

对HDFS的一个普通文件来说，构成它的基本单位是块。对于EC模式下的文件，构成它的基本单位为块组。块组由一定数目的数据块加上生成的校验块放一起构成。以RS(6，3)为例，每一个块组包含1-6个数据块，以及3个校验块。

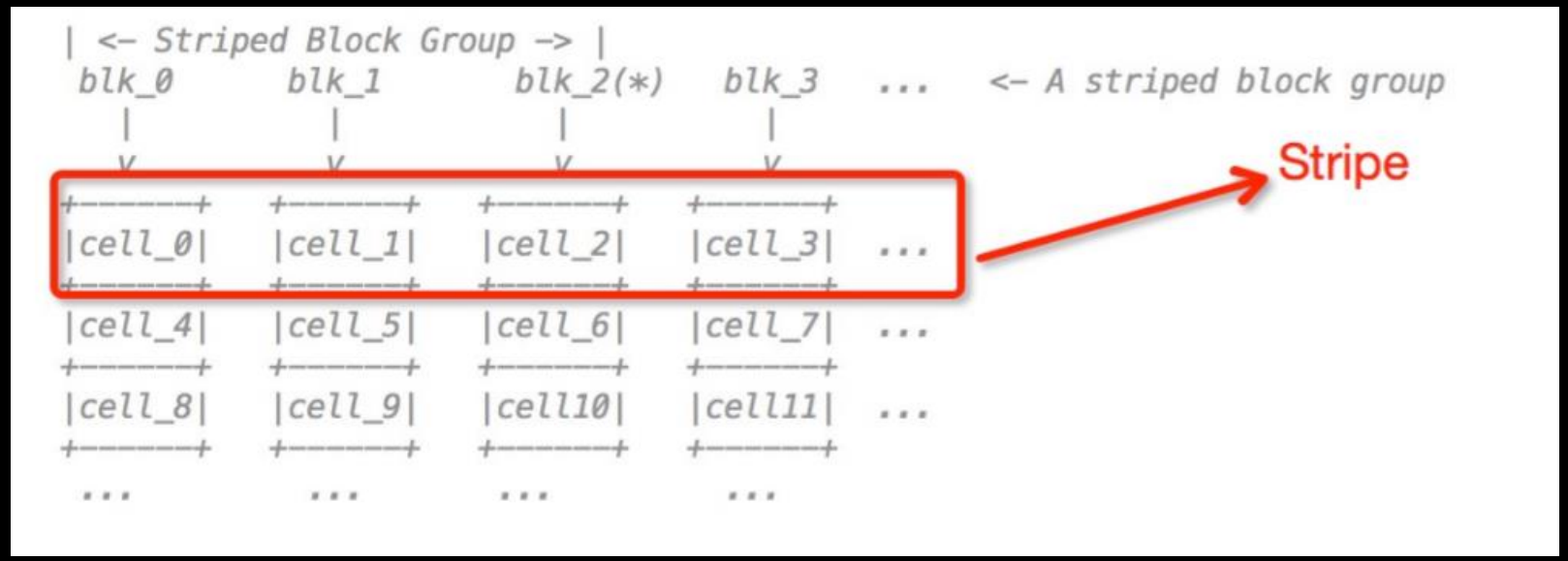
在一般的hdfs集群中，小文件一般占集群的3/4空间。为了更好的支持小文件，hdfs前期只支持条形布局（Striping Layout），以后会支持连续布局（Contiguous Layout）。

Hadoop Erasure Coding 的实现

EC概念中data block数据块,parity block校验块,stripe条带等这些概念在HDFS中的转化：

* data block,parity block在HDFS中的展现就是普通的block数据块.
* stripe条带的概念需要将每个block进行分裂,每个block由若干个相同大小的cell组成,然每个stripe由于一行cell构成,相当于所有的data block和parity block抽取出了一行.

下面用图形展示：



Hadoop Erasure Coding 的实现

目前hadoop3.0支持的编码格式有：

* RS-DEFAULT-10-4-64k
* RS-DEFAULT-3-2-64k
* RS-DEFAULT-6-3-64k
* RS-LEGACY-6-3-64k
* XOR-2-1-64k

使用方式：

hdfs erasurecode [generic options]

[-setPolicy [-p <policyName>] <path>]	某个路径设置EC格式
[-getPolicy <path>]	查看路径EC格式
[-unsetPolicy <path>]	取消EC
[-listPolicies]	显示已设置EC

Erasure Coding技术的优劣势

副本策略与纠删码策略对比：

	存储效率	计算开销	修复效率
副本策略	低	无	高
编码策略	高	有	低

优势：极大的减少存储开销
劣势：* 网络带宽的消耗，因为数据恢复需要去读其他的数据块和校验块
　　* 进行编码，解码计算需要消耗CPU资源

概括来说，编码策略既消耗网络又消耗cpu，用于线上服务可能会不够稳定，所以最好用冷数据集群。

3、YARN Timeline Service v.2

由于 MR 的 job history server 只能存储 MR 的历史记录， 而对于很多 on yarn 的非MR任务，它们的运行结果也要保存，因此 timelineservice 应运而生了 。在YARN中通过Timeline Server用一种通用的形式解决对application的当前和历史信息的存储和检索。

Timeline service v2相对于原先的版本而言，在可伸缩性上做了较大的改善，同时，v2版本在稳定性和性能上面也做出了提升，原先版本不适用于大集群，v2版本使用hbase取代了原先的leveldb作为后台的存储工具。

4、Shell脚本重写

* 增加了参数冲突检测，避免重复定义和冗余参数

* CLASSPATH, JAVA_LIBRARY_PATH, and LD_LIBRARY_PATH等参数的去重，缩短环境变量

* shell脚本重构，将更多的代码加入function中，提供重载，删除重复代码，便于测试

* 脚本清理和简化

* 尽可能与当前系统保持兼容

1 master

```
# limitations under the License.

function hadoop_usage
{
  echo "Usage: hadoop-daemon.sh [--config confdir] (start|stop|status) <hadoop-command> <args
...>"
}

# let's locate libexec...
if [[ -n "${HADOOP_HOME}" ]]; then
  HADOOP_DEFAULT_LIBEXEC_DIR="${HADOOP_HOME}/libexec"
else
  this="${BASH_SOURCE-$0}"
  bin=$(cd -P -- "$(dirname -- "${this}")" >/dev/null && pwd -P)
  HADOOP_DEFAULT_LIBEXEC_DIR="${bin}/../libexec"
fi

HADOOP_LIBEXEC_DIR="${HADOOP_LIBEXEC_DIR:-$HADOOP_DEFAULT_LIBEXEC_DIR}"
# shellcheck disable=SC2034
HADOOP_NEW_CONFIG=true
if [[ -f "${HADOOP_LIBEXEC_DIR}/hdfs-config.sh" ]]; then
  . "${HADOOP_LIBEXEC_DIR}/hdfs-config.sh"
else
  echo "ERROR: Cannot execute ${HADOOP_LIBEXEC_DIR}/hdfs-config.sh." 2>&1
  exit 1
fi

if [[ $# = 0 ]]; then
  hadoop_exit_with_usage 1
fi

daemonmode=$1
shift

if [[ -z "${HADOOP_HDFS_HOME}" ]]; then
  hdfsscript="${HADOOP_HOME}/bin/hdfs"
else
  hdfsscript="${HADOOP_HDFS_HOME}/bin/hdfs"
fi
```

53,1 71%

1 master

```
#   HADOOP_NICENESS The scheduling priority for daemon
##

usage="Usage: hadoop-daemon.sh [--config <conf-dir>] [--hosts hostlistfile] [--script script
] (start|stop) <hadoop-command> <args...>"

# if no args specified, show usage
if [ $# -le 1 ]; then
  echo $usage
  exit 1
fi

bin=`dirname "${BASH_SOURCE-$0}"`
bin=`cd "$bin"; pwd`

DEFAULT_LIBEXEC_DIR="$bin"/../libexec
HADOOP_LIBEXEC_DIR=${HADOOP_LIBEXEC_DIR:-$DEFAULT_LIBEXEC_DIR}
. $HADOOP_LIBEXEC_DIR/hadoop-config.sh

# get arguments

#default value
hadoopScript="$HADOOP_PREFIX"/bin/hadoop
if [ "--script" = "$1" ]
  then
    shift
    hadoopScript=$1
    shift
fi
startStop=$1
shift
command=$1
shift

hadoop_rotate_log ()
{
    log=$1;
    num=5;
    if [ -n "$2" ]; then
```

65,1 15%

仅将文本发送到当前选项卡

5、Shaded client jars

Client Jars的类路径隔离

2.x版本的hadoop-client Maven artifact将Hadoop的传递依赖加到了Hadoop应用程序的classpath。如果这些依赖的版本与应用程序所用的版本有冲突的话，可能会有问题。

如：Hadoop客户端可能需要在应用程序的类路径上存在特定版本的Java库，但应用程序已经使用了该特定版本Java库的不兼容版本。这可能导致在运行时产生ClassNotFoundException或NoSuchMethodError异常，或者其它未知的、未经测试的错误。

HADOOP-11804添加了新的 hadoop-client-api 和 hadoop-client-runtime artifacts 将hadoop的依赖区分到了一个单独的jar包，这就避免了hadoop的依赖泄露到应用程序，应用程序可以自由使用所选择的任何依赖项和版本。

6、MapReduce task-level native optimization

MR任务级本地优化。

MapReduce添加了Map输出collector的本地实现（用c++实现了一个mapoutput collector）。对于shuffle密集型作业，这将会有30%以上的性能提升。

使用方式：
mapreduce.job.map.output.collector.class=
org.apache.hadoop.mapred. nativetask.NativeMapOutputCollectorDelegator

局限：
currently only Text and BytesWritable is supported

7、Support for more than 2 NameNodes

支持两个以上Namenode

允许用户运行多个Standby NN，更高的容错性。

刚开始HDFS NameNode高可用提供了一个namenode,和Standby namenode. 这种架构能够容忍系统中的任何一个节点的失败。

然而，一些部署需要更高程度的容错性。这是通过这一新功能，它允许用户运行多个备用节点Standby namenode。例如，通过配置三个NameNodes和五个journalnodes，集群能够容忍两节点而不是一个失败。

8、 Default ports of multiple services have been changed

多个服务的默认端口更改

在此之前，多个Hadoop服务的默认端口都属于Linux的临时端口范围（32768-61000）。这就意味着我们的服务在启动的时候可能因为和其他应用程序产生端口冲突而无法启动。现在这些可能会产生冲突的端口已经不再属于临时端口的范围，这些端口的改变会影响NameNode, Secondary NameNode, DataNode以及KMS。

Namenode ports: 50470 --> 9871, 50070--> 9870, 8020 --> 9820
Secondary NN ports: 50091 --> 9869,50090 --> 9868
Datanode ports: 50020 --> 9867, 50010--> 9866, 50475 --> 9865, 50075 --> 9864
Kms server ports: 16000 --> 9600 (原先的16000与HMaster端口冲突)

9、Support for Microsoft Azure Data Lake and Aliyun Object Storage System filesystem connectors

支持Microsoft Azure Data Lake和Aliyun对象存储系统的连接器

Hadoop现在支持与Microsoft Azure Data Lake 和 Aliyun对象存储系统结合作为可供选择的hadoop兼容的文件系统。针对这两个系统，Apache Hadoop增加了文件系统连接器，允许用户通过正常的Hadoop文件系统API与这些存储系统进行交互。

10、Intra-datanode balancer

Intra-datanode 均衡器

一个DataNode可以管理多个磁盘，正常写入操作各磁盘会被均匀填满。然而，当添加或替换磁盘时可能导致此DataNode内部的磁盘存储的数据严重内斜。这种情况现有的HDFS balancer是无法处理的。这种情况是由新intra-DataNode平衡功能来处理，通过hdfs diskbalancer CLI来调用。

语法： hdfs diskbalancer

[-plan <datanode> -fs <namenodeURI>]	创建一个balancer计划
[-execute <planfile>]	在datanode上执行plan
[-query <datanode>]	从datanode获取balancer状态
[-cancel <planfile>]	停止一个running的plan
[-cancel <planID> -node <datanode>]	
[-report -node [<DataNodeID|IP|Hostname>,...]]	报告datanodes的信息
[-report -node -top <topnum>]	

11、Reworked daemon and task heap management

重做守护进程河任务堆内存管理

Hadoop守护进程和MapReduce任务的堆内存管理发生了一系列变化。

HADOOP-10950 介绍了定义守护进程的堆内存的新方法，并且可以根据主机内存大小可以自动调整，HADOOP_HEAPSIZE已弃用。

MAPREDUCE-5785 简化了map和reduce task堆大小的配置方法，所需的堆大小不再需要通过任务配置和Java选项实现。已经指定的现有配置不受此更改影响。

2.0： mapreduce.{map,reduce}.memory.mb
mapreduce.{map,reduce}.java.opts

3.0： mapreduce.job.heap.memory-mb.ratio
mapreduce.map/reduce.java.opts

REDOOP

专注产品　全面合作　开放自由