Hadoop 3

本文根据2017年7月14日Strata Data Conference的主题演讲《Apache Hadoop 3.0的特性和开发进展的更新 (Apache Hadoop 3.0 features and development update)》（By Andrew Wang and Daniel Templeton in Cloudera）总结而成。

Apache Hadoop已经伴随着开源的大数据分析同步地发展了十多年。伴随着即将发布的3.0版，Apache Hadoop持续地进化着，并不断地加入显著的新特性，比如HDFS纠删码、YARN的时间线服务version 2和MapReduce的任务级别的优化。这些新特性增强了Hadoop的性能、可扩展性和多租户的能力。本文会综合介绍Apache Hadoop 3.0版的特性。

# Hadoop3重要更新

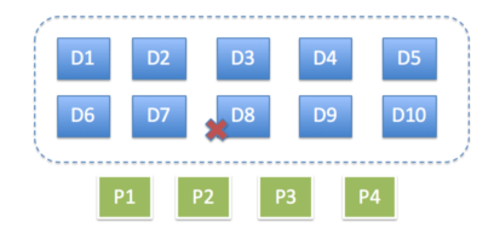
1. 存储
   1. HDFS：使用纠删码（Erasure Code）代替分片（Replication）
2. Classpath隔离
3. YARN的更新
   1. 新的YARN UI
   2. ATSv2(Application Timeline Server version2)
4. 其他更新与bug修复

# HDFS Erasure Code(HDFS-7285)

## Erasure Code(纠删码)介绍

纠删码（erasure coding，EC）是一种数据保护方法，它将数据分割成片段，把冗余数据块扩展、编码，并将其存储在不同的位置，比如磁盘、存储节点或者其它地理位置。

下图简单阐述了纠删码原理：D1…D10 这10个数据块（甚至可以理解为单个磁盘），P1、P2、P3、P4是对这10个数据块计算出来的纠删码块，这14个块分布在不同的server磁盘上。



根据10个数据块算出4个校验块，即可以容忍任意4个Block的丢失。此处存储开销为： 14/10=140%。任意一个块损坏，需要通过网络读取10个Block进行恢复。

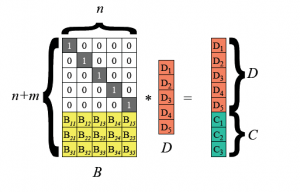
纠删码可以写作EC(k,m)，其中k为数据块的数量，m为纠删码块的数量。给定k个数据块d1, d2,…, dk，和一个正整数m， RS根据k个数据块生成m个校验块， c1, c2,…, cm。 对于任意的k和m， 从k个原始数据块和m 个校验块中任取k块就能解码出原始数据， 即RS最多容忍m个数据块或者校验块同时丢失（纠删码只能容忍数据丢失，无法容忍数据篡改，纠删码正是得名与此）。

## Reed-Solomon纠删码介绍

Reed Solomon Coding是存储领域常用的一种纠删码。Hadoop 3中使用的纠删码即为Reed Solomon Coding。它被广泛的应用于各种商业用途，最显著的是在CD、DVD和蓝光光盘上的使用；在数据传输中，它也被用于DSL和WiMAX；广播系统中DVB和ATSC也有它的身影；在计算机科学里，它是RAID 6标准的重要成员。

### 编码原理

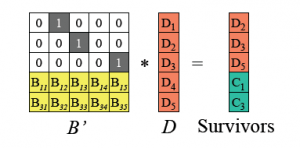
把输入数据视为向量D=(D1，D2，…, Dn）, 编码后数据视为向量（D1, D2,…, Dn, C1, C2,.., Cm)，RS编码可视为如图1所示矩阵运算。 下图最左边是编码矩阵， 矩阵上部是单位阵（n行n列），下边是vandermonde矩阵B(m行n列), vandermode矩阵如图2所示， 第i行，第j列的原数值为j^(i-1)。



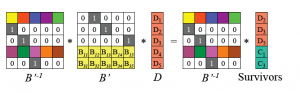
### 数据恢复原理

RS最多能容忍m个删除错误。 数据恢复原理的过程如下：

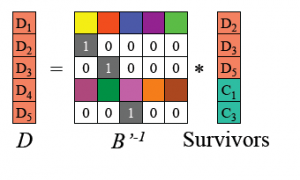
1. 从编码矩阵中删去丢失数据块和丢失编码块对应行。 假设D1、C2丢失， 根据图1所示RS编码运算等式，我们得到如下B’以及等式。



1. 由于B‘是可逆的， 两边乘上B’逆矩阵。



1. 得到如下原始数据D的计算公式



1. 对D重新编码，得到丢失的校验码矩阵求逆采用高斯消元法， 需要进行实数加减乘除四则运算，无法作用于字长为w的二进制数据。 为了解决这个问题， RS采用伽罗华群GF（2^w)中定义的四则运算法则。 GF(2^w）域有2^w个值， 每个值都对应一个低于w次的多项式， 这样域上的四则运算就转换为多项式空间的运算。 GF(2^w)域中的加法就是XOR。

### RS码的特点

* 低冗余度，高可靠性。
* 数据恢复代价高。丢失数据块或者编码块时，RS需要读取n个数据块和校验块才能恢复数据，数据恢复效率也在一定程度上制约了RS的可靠性。
* 数据更新代价高。数据更新相当于重新编码，代价很高，因此常常针对只读数据，或者冷数据。
* RS编码依赖于两张2^w-1大小的log表， 通常只能采用16位或者8位字长，不能充分利用64位服务器的计算能力， 具体实现上可能要做一些优化。

## Hadoop3中Reed Solomon Coding与Replication的对比

RS最大的优点就是存储开销比Replication小得多。假设有一个大文件需要存储在HDFS中。对于Replication方案，此文件有2个replica。存储额外的开销（overhead）为2/1=200%。

但对于RS Coding方案（一般选择RS(3,2), RS(6,3)与RS(10,4)）：

* RS(3,2)：额外开销为2/3=67%，最多容忍5个block中2个block有数据丢失
* RS(6,3)：额外开销为3/6=50%，最多容忍9个block中3个block有数据丢失
* RS(10,4)：额外开销为4/10=40%，最多容忍14个block中4个block有数据丢失

由此可见，RS Coding的开销远远小于Replication。

然而，纠删码方案应用在HDFS中也有一些缺陷。

首先对于一个Hadoop集群，1个文件的block分布在不同的节点上，因此跨机房（cross-rack）通信非常频繁。然而，文件的重建（reconstruction）需要多个block，这个过程是一个network-intensive的过程，对网络连接要求较高。

其次，编码/解码的过程计算代价很大，社区推荐使用Intel公司的ISA-L(Intelligent Storage Acceleration Library)，其速度远快于Java的实现。

再次，由于更新文件需要对整个文件重新编码，代码很高，所以此方案最好用在冷数据上（HDFS本身也是基于数据多读少写的假设来设计的）。

最后，为了避免block数量过快膨胀，文件最好被合并成大文件。

# Classpath isolation(HADOOP-11656)

在Hadoop2中，很多Hadoop的依赖关系被暴露在外，导致其与一些其他的库同时使用时会产生冲突。比如比如google Guava在混合使用Hadoop、HBase和Spark时，很容易产生冲突。

在Hadoop3中，Hadoop client的依赖关系是被隐藏的。（HADOOP-11804）。

# YARN的更新

## 新UI

新UI使用NodeJS与Ember编写，是一个富客户端（Rich client）应用。比起以前的版本，它能更详细地显示集群的使用情况，比如Memory与CPU的使用，任务队列的suburst graph与NodeManager的Heatmap等。另外此UI集成了ATSv2。

* 1. **ATSv2(Application Timeline Server version2)**

Timeline Server是hadoop2.4新引入的一个功能。由于hadoop2开始支持各种计算框架，而目前各个框架自己都提供独立的history server机制，所以Timeline Server的主要意义在于将各个框架自己提供的history server统一集成到一个history server中，方便用户统一查询已完成的所有任务，主要功能分为两块：

* 提供已完成任务的通用信息，包括任务名，队列等等，这个跟YARN主界面看到的类似。
* 集中收集各个框架提供的任务信息，比如mapreduce的map数，reduce数，counter等等，以便进行统一查询。

Hadoop3使用了ATSv2，主要解决了v1中的2个问题：

1. 可扩展性

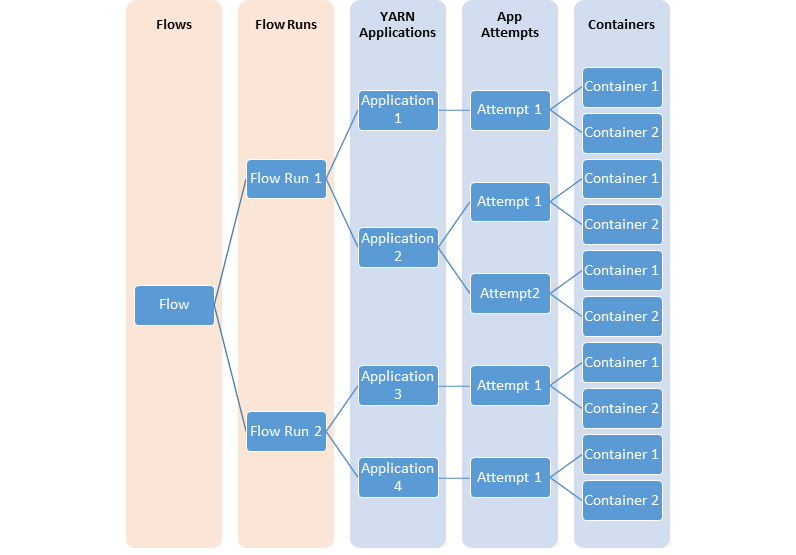
V1只能使用单个writer/reader实例，并且只能用在小型集群。 V2使用可扩展性更强的分布式writer架构和后端存储。

ATSv2解耦了数据的writing和reading。 它使用分布式的collector(writer)，每个YARN应用程序持有一个collector。 reader是独立的实例，专用于通过REST API提供查询。

ATSv2选择Apache HBase作为主要的后端存储，因为Apache HBase可扩展性强并且读写较快。

2. 可用性

在许多情况下，我们需要使用一系列YARN应用程序来完成业务逻辑。 ATSv2明确支持”flow”的概念。此外，配置和指标有进一步更新。下图说明了不同YARN实体建模流程之间的关系。



# 其他更新与bug修复

## 5.1 基于Java8重新开发

由于Java7在2015年7月被停止支持，这直接迫使Hadoop社区基于JDK 1.8重新开发Hadoop3。Hadoop3精简了Hadoop内核，剔除了过期的API和实现，将默认组件实现替换成最高效的实现（比如将FileOutputCommitter缺省实现换为v2版本，废除hftp转由webhdfs替代，移除Hadoop子实现序列化库org.apache.hadoop.Records。并且Shell脚本也进行了重构，修复了大量bug，增加了新特性，支持动态命令等。

## 5.2 MapReduce的更新

### 5.2.1 实现了C/C++的map output collector

Hadoop3为MapReduce增加了C/C++的map output collector实现（包括Spill，Sort和IFile等），通过作业级别参数调整就可切换到该实现上。对于shuffle密集型应用，其性能可提高约30%。

### 5.2.2 MapReduce参数自动推断

Hadoop3实现了MapReduce内存参数自动推断。在Hadoop 2.0中，为MapReduce作业设置内存参数非常繁琐，一旦设置不合理，则会使得内存资源浪费严重。

# 参考文献

* <http://www.jianshu.com/p/3936a8698d36>
* <http://dataunion.org/24423.html>
* <http://ju.outofmemory.cn/entry/327902>
* https://issues.apache.org/jira/browse/YARN-4237
* https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/TimelineServiceV2.html