# 第四章 Hadoop的IO操作

Hadoop自带一套原子操作用于数据IO操作，其中一些技术比Hadoop本身更常用，比如数据完整性和压缩，其他是Hadoop工具和API，它所构建的模块可用于开发分布式系统，比如序列化操作和在盘数据结构

## 4.1 数据完整性

检验数据是否损坏：在数据第一次进入系统是计算校验和并在数据在传输后再次计算校验和，这样就能发现是否损坏了。注意一定要使用ECC内存，校验和也是可能损坏的

常用的错误检测码是CRC-32，任何大小的数据输入均计算得到一个32位的整数校验和。

### 4.1.1 HDFS的数据完整性

每个datanode均持久操作一个用于验证的检验和日志。所以他知道每个数据块最后一次验证的时间。

不只是在客户端读取datanode数据时才验证数据完整，datanode内部也会有一个后台线程运行的DataBlockScanner ,从而定期验证存储在这个datanode上的所有数据块。

由于HDFS存储着每个数据快的复本，因此它可以通过复本来修复损坏的数据块。

### 4.1.2 LocalFileSystem

校验和的计算代价是相当低的。

### 4.1.3 ChecksumFileSystem

localFileSystem通过ChecksumFileSystem来完成自己的任务

## 4.2 压缩

文件压缩有两个好处：减少存储文件所需要的磁盘空间，并加速数据在网络和磁盘上的传输。

所有的压缩算法都需要权衡空间/时间。

### 4.2.1 codec

Codec实现了一种压缩-解压缩算法

### 4.2.2 压缩和输入分片

使用哪种压缩格式？

1. 使用容器文件格式，例如顺序文件，这些文件格式都支持压缩和切分
2. 使用支持切分的压缩格式，例如bzip2
3. 在应用中将文件切块，确保压缩后的数据近似HDFS块的大小
4. 储存未经压缩的文件

### 4.2.3 在MapReduce中使用压缩

## 4.3 序列化

**序列化：将结构化对象转化为字节流以便在网络上传输或写入磁盘进行永久存储的过程。**

**反序列化：将字节流转回结构化对象的逆过程。**

序列化在分布式数据处理的两大领域经常出现：进程间通信，永久性存储。

在Hadoop中，系统中多个节点上的通信是通过RPC实现的，RPC协议将消息序列化成二进制流后发送到远程节点。远程的节点将二进制流反序列化为原始消息。

RPC序列化格式：

1. 紧凑（高效的使用存储空间）
2. 快速（以读写数据的额外开销比较小）
3. 可扩展（可以透明的读取老格式的数据）
4. 支持互操作（使用不同的语言读写永久性存储的数据）

Hadoop使用的是自己的序列化格式Writbale

### 4.3.1 Writable 接口

### 4.3.2 Writable类

1.JAVA基本类型的Writable封装器

Writable类对JAVA基本类型提供了封装，char除外。所有封装都包含get（）和set（）两种方法用于读取和储存封装的值。

2.Text类型

Text是针对UTF-8的writable类。

由于着重使用标准的utf-8编码，所以Text类和Java String 类之间存在一定的差异。

1. 索引，对text类的索引是根据编码后的字节序列中的位置实现的。Text的charAt（）方法返回的是表示unicode编码位置的int类型值。
2. Unicode
3. 迭代 对Text类中的unicode字符进行迭代是非常复杂的，无法根据增加索引值来实现迭代。
4. 可变性 与String相比，Text的另一个区别上就是它是可变的。
5. 对String重新排序 大多数需要将Text类转化成String类。

3．BytesWritable

BytesWritable 是对二进制数组的封装。它的序列化格式为一个指定所包含数据字节数的整数域（4个字节），后跟着数据本身内容。它也是可变的。

4.NullWritable

NullWritable是Writable的特殊类型。它的序列化长度为0；它不读取数据也不写入数据，它充当占位符。比如在MapReduce中，不需要键或值的序列化地址，就可以将其设置为NullWritable。它是一个不可变的单实例类型。

5.ObjectWritable和 GenericWritable

ObjectWritable 是对java基本类型的一个通用封装。

6.Writable集合类

### 4.3.3 实现定制的Writable集合

## 4.3 序列化框架

大多数MapReduce程序使用的都是Writable类型的键和值。并不是强制使用MapReduce的API，事实上可以使用任何类型，只要一种机制对每个类型进行类型和二进制的来回转化。

为什么不用 java object serialiation?

因为它看起来太复杂了，不满足前面所说的四个条件。

IDL（Interface Description Language）

## 4.4 Avro（暂停）

Apache Avro 是一个独立与编程语言的数据序列化系统。旨在解决Writable类型的不足：缺乏语言的可移植性。

Avro可以使用Avro IDL，类c语言来写模式，也可以使用JSON的数据编码方式。

Avro规范精确定义了所有实现都支持的二进制格式。

Avro有丰富的模式解析。

Avro为一系列对象指定了一个对象容器格式。

### 4.4.1 Avro数据类型和模式

Avro的基本数据类型： null boolean int long float double bytes string

Avro复杂类型：array,map,record,enum,fixed,union

每个Avro语言API都包含该语言特定的Avro类型。

所有语言都支持动态映射，即运行前并不知道具体模式。对此，JAVA称为“通用”映射

### 4.4.2 内存中的序列化和发序列化

## 4.5 基于文件的数据结构

对于基于MapReduce的数据处理，将每个二进制数据大对象（blob）单独放在各自的文件中不能实现可扩展性。

### 4.5.1 关于SequenceFile

1. SequenceFile的写操作

### 4.5.2 关于MapFile

MapFile是排序的SequenceFile。它有索引，所以可以按键查找。