> 良好习惯的养成需要21天的时间

MapReduce之InputFormat数据输入

FileInputFormat切片机制

FileInputFormat切片机制

1、切片机制

（1）简单地按照文件的内容长度进行切片

（2）切片大小，默认等于Block大小

（3）切片时不考虑数据集整体，而是逐个针对每一个文件单独切片

2、案例分析

（1）输入数据有两个文件：

```
file1.txt    320M
file2.txt    10M
```

（2）经过FileInputFormat的切片机制运算后，形成的切片信息如下：

```
file1.txt.split1--  0~128
file1.txt.split2--  128~256
file1.txt.split3--  256~320
file2.txt.split1--  0~10M
```

这里2.x以上的hadoop默认块大小为128M

FileInputFormat切片大小的参数配置

（1）源码中计算切片大小的公式

Math.max(minSize, Math.min(maxSize, blockSize));

mapreduce.input.fileinputformat.split.minsize=1 默认值为1

mapreduce.input.fileinputformat.split.maxsize= Long.MAXValue 默认值Long.MAXValue

因此，默认情况下，切片大小=blocksize。

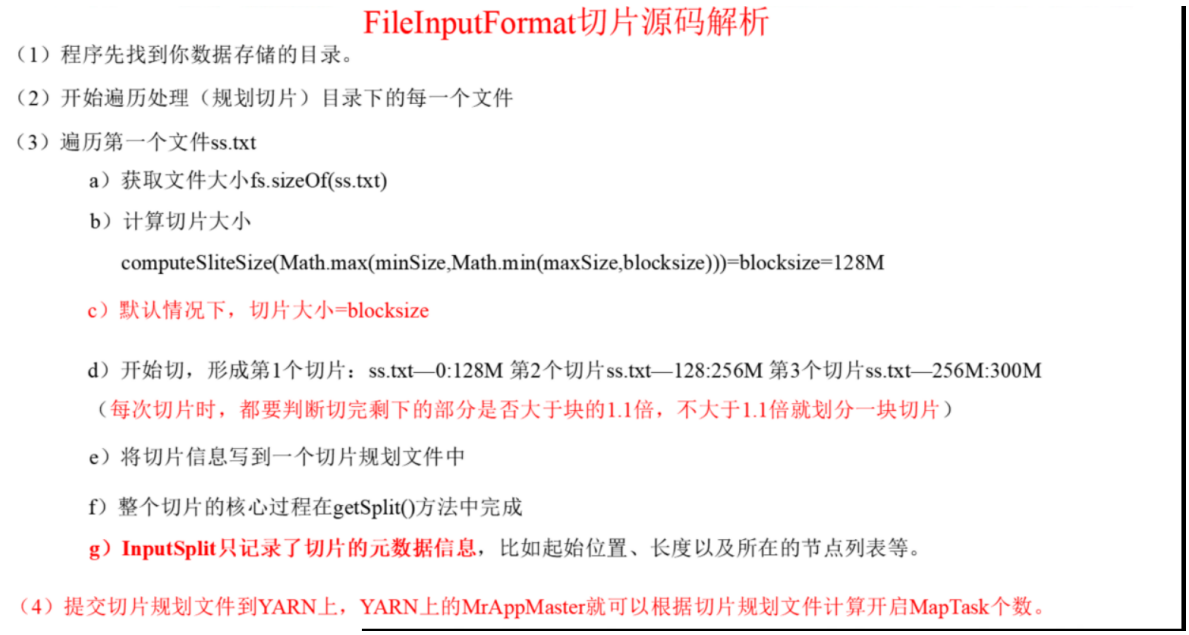
（2）切片大小设置

maxsize（切片最大值）：参数如果调得比blockSize小，则会让切片变小，而且就等于配置的这个参数的值。

minsize（切片最小值）：参数调的比blockSize大，则可以让切片变得比blockSize还大。

（3）获取切片信息API

```
// 获取切片的文件名称
String name = inputSplit.getPath().getName();
// 根据文件类型获取切片信息
FileSplit inputSplit = (FileSplit) context.getInputSplit();
```



FileInputFormat实现类

Map reduce任务的输入文件一般是存储在HDFS里面。输入的文件格式包括：

> 基于行的日志文件、二进制格式文件等。这些文件一般会很大，达到数十GB，甚至更大。那么Mapreduce是如何读取这些数据的呢？下面我们首先学习FileInputFormat接口。

FileInputformat常见实现类

> 包括：TextInputFormat、 Key value TextInput Format、 NLineInput Format、 Combine TextInputFormat和自定义InputFormat等。

TextInputFormat切片机制

TextInput Format是默认的InputFormat。

> 每条记录是一行输入。键是 Long writable类型，存储该行在整个文件中的起始字节偏移量。值是这行的内容，不包括任何行终止符（换行符和回车符）

以下是一个示例，比如，一个分片包含了如下4条文本记录。

```
Rich learning form
Intelligent learning engine
Learning more convenient
From the real demand for more close to the enterprise每条记录表示为以下键/值对：

（0, Rich learning form)
（19,  Intelligent learning engine)
（47,  Learning more convenient)
（72,  From the real demand for more close to the enterprise）
很明显，键并不是行号。一般情况下，很难取得行号，因为文件按字节而不是按行切分。
```

KeyValueTextInputFormat切片机制

> 每一行均为一条记录，被分隔符分割为key, value。可以通过在驱动类中设置conf.set.(Key ValueLineRecordReader.KEY_VALUE_SEPERATOR，"\t"）来设定分隔符。默认分隔符是tab（\t)

以下是一个示例，输入是一个包含4条记录的分片。其中->表示一个（水平方向的）制表符。
```
line1->Rich learning form
line2->Intelligent learning engine
line3>Learning more convenient
line4->From the real demand for more close to the enterprise
```
每条记录表示为以下键/值对
```
（line1, Rich learning form）
（line2， Intelligent learning engine
（line3， Learning more convenient）
（line4， From the real demand for more close to the enterprise）
```
此时的键是每行排在制表符之前的Text序列。

NLineInputFormat切片机制

> 如果使用 NlineInputFormat，代表每个map进程处理的 InputSplit不再按Block块去划分，而是按NlineInput Format指定的行数N来划分。即输入文件的总行数N=切片数，如果不整除，切片数=商+1

以下是一个示例，仍然以上面的4行输入为例
```
Rich learning form
Intelligent learning engine
Learning more convenient
From the real demand for more close to the enterprise

例如，如果N是2，则每个输入分片包含两行。开启2个Map Task
一个mapper接受
（0,Rich learning form）
（19，Intelligent learning engine）

另一个 mapper则收到后两行：
（47， Learning more convenient）
（72， From the real demand for more close to the enterprise）
```
这里的键和值与 TextInputFormat生成的一样。

CombineTextInputFormat切片机制

问题背景：

> 框架默认的TextInputformat切片机制是对任务按文件规划切片，不管文件多小，都会是一个单独的切片，都会交给一个MapTask，这样如果有大量小文件，就会产生大量的MapTask，处理效率极其低下。

应用场景：

> CombineTextInputFormat用于小文件过多的场景，它可以将多个小文件从逻辑上规划到一个切片中，这样，多个小文件就可以交给一个MapTask处理。

虚拟存储切片最大值设置

```
CombineTextInputFormat.setMaxInputSplitSize(job, 4194304);// 4m
```
虚拟存储切片最大值设置可以根据实际的小文件大小情况来设置具体的值。

切片机制

生成切片过程包括：虚拟存储过程和切片过程二部分。

（1）虚拟存储过程

（a）将输入目录下所有文件按照文件名称字典顺序一次读入，记录文件大小，并累加计算所有文件的总长度。
（b）根据是否设置setMaxInputSplitSize值，将每个文件划分成一个一个setMaxInputSplitSize值大小的文件。
（c）注意：当剩余数据大小超过setMaxInputSplitSize值且不大于2倍setMaxInputSplitSize值，此时将文件均分成2个虚拟存储块（防止出现太小切片）。
大于怎么办？
例如setMaxInputSplitSize值为4M，最后文件剩余的大小为4.02M，如果按照4M逻辑划分，就会出现0.02M的小的虚拟存储文件，所以将剩余的4.02M文件切分成（2.01M和2.01M）两个文件。

（2）切片过程

（a）判断虚拟存储的文件大小是否大于setMaxInputSplitSize值，大于等于则单独形成一个切片。
（b）如果不大于则跟下一个虚拟存储文件进行合并，共同形成一个切片。
（c）测试举例：有4个小文件大小分别为1.7M、5.1M、3.4M以及6.8M这四个小文件，则虚拟存储之后形成6个文件块，大小分别为：
1.7M，（2.55M、2.55M），3.4M以及（3.4M、3.4M）
最终会形成3个切片，大小分别为：
（1.7+2.55）M，（2.55+3.4）M，（3.4+3.4）M

自定义 InputFormat

> 在企业开发中，Hadoop框架自带的 IInputFormat类型不能满足所有应用场景，需要自定义InputFormat来解决实际问题。

自定义 InputFormat步骤如下：
（1）自定义一个类继承 FileInputFormat
（2）改写 RecordReader，实现一次读取一个完整文件封装为KV。
（3）在输出时使用 SequenceFileOutPutFormat输出合并文件。