HDFS 块和 Input Splits 的区别与联系

相信大家都知道，HDFS 将文件按照一定大小的块进行切割，（我们可以通过 dfs.blocksize 参数来设置 HDFS 块的大小，在 Hadoop 2.x 上，默认的块大小为 128MB。）也就是说，如果一个文件大小大于 128MB，那么这个文件会被切割成很多块，这些块分别存储在不同的机器上。当我们启动一个 MapReduce 作业去处理这些数据的时候，程序会计算出文件有多少个 Splits，然后根据 Splits 的个数来启动 Map 任务。那么 HDFS 块和 Splits 到底有什么关系？

为了简便起见，下面介绍的文件为普通文本文件。

HDFS 块

现在我有一个名为 iteblog.txt 的文件，如下：

```
[iteblog@iteblog.com /home/iteblog]$ ll  iteblog.txt
-rw-r--r-- 1 iteblog iteblog 454669963 May 15 12:07 iteblog.txt
```

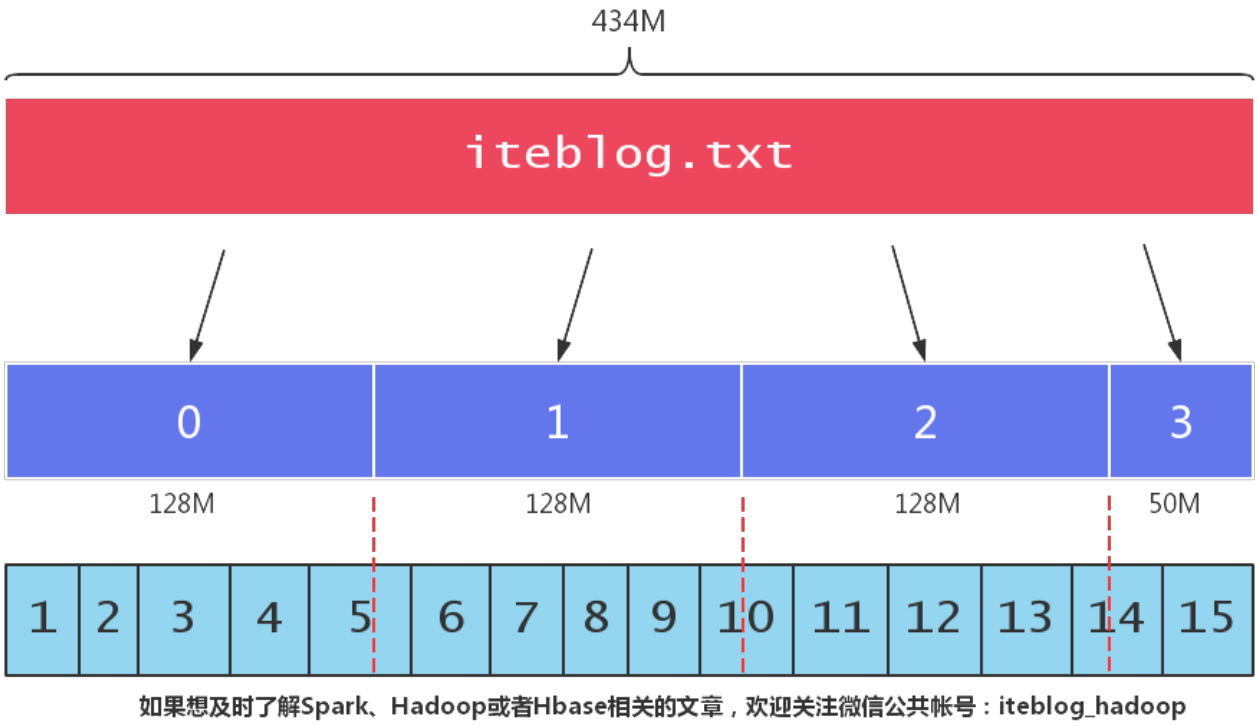
很明显，这个文件大于一个 HDFS 块大小，所有如果我们将这个文件存放到 HDFS 上会生成 4 个 HDFS 块，如下（注意下面的输出做了一些删除操作）：

```
[iteblog@iteblog.com /home/iteblog]$ hadoop -put iteblog.txt /tmp
[iteblog@iteblog.com /home/iteblog]$ hdfs fsck /tmp/iteblog.txt -files -blocks
/tmp/iteblog.txt 454669963 bytes, 4 block(s):  OK
0. BP-1398136447-192.168.246.60-1386067202761:blk_8133964845_1106679622318 len=1342
17728 repl=3
1. BP-1398136447-192.168.246.60-1386067202761:blk_8133967228_1106679624701 len=1342
17728 repl=3
2. BP-1398136447-192.168.246.60-1386067202761:blk_8133969503_1106679626977 len=1342
17728 repl=3
3. BP-1398136447-192.168.246.60-1386067202761:blk_8133970122_1106679627596 len=5201
6779 repl=3
```

可以看出 iteblog.txt 文件被切成 4 个块了，前三个块大小正好是 128MB（134217728），剩下的数据存放到第 4 个 HDFS 块中。

如果文件里面有一行记录的偏移量为 134217710，长度为 100，HDFS 如何处理？

答案是这行记录会被切割成两部分，一部分存放在 block 0 里面；剩下的部分存放在 block 1 里面。具体的，偏移量为134217710，长度为18的数据存放到 block 0
里面；偏移量134217729，长度为82的数据存放到 block 1 里面。
可以将这部分的逻辑以下面的图概括



说明：

- 图中的红色块代表一个文件
- 中间的蓝色矩形块代表一个 HDFS 块，矩形里面的数字代表 HDFS 块的编号，读整个文件的时候是从编号为0的 HDFS 块开始读，然后依次是1,2,3...
- 最下面的一行矩形代表文件里面存储的内容，每个小矩形代表一行数据，里面的数字代表数据的编号。红色的竖线代表 HDFS 块边界(block boundary)。

从上图我们可以清晰地看出，当我们往 HDFS 写文件时，HDFS 会将文件切割成大小为 128MB 的块，切割的时候不会判断文件里面存储的到底是什么东西，所以逻辑上属于一行的数据会被切割成两部分，这两部分的数据被物理的存放在两个不同的 HDFS 块中，正如上图中的第5、10以及14行被切割成2部分了。

File Split

现在我们需要使用 MapReduce 来读取上面的文件，由于是普通的文本文件，所以可以直接使用 TextInputFormat 来读取。下面是使用 TextInputFormat获取到的 FileSplit 信息：

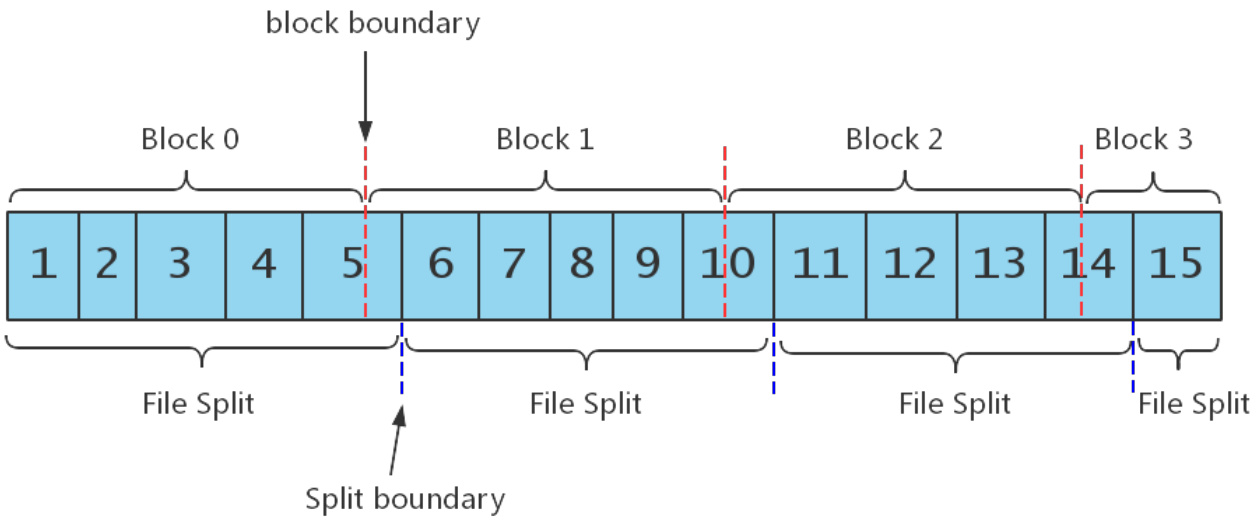
```
scala> FileInputFormat.addInputPath(job,new Path("/tmp/iteblog.txt"));

scala> val format = new TextInputFormat;

scala> val splits = format.getSplits(job)

scala> splits.foreach(println)
hdfs://iteblogcluster/tmp/iteblog.txt:0+134217728
hdfs://iteblogcluster/tmp/iteblog.txt:134217728+134217728
hdfs://iteblogcluster/tmp/iteblog.txt:268435456+134217728
hdfs://iteblogcluster/tmp/iteblog.txt:402653184+52016779
```

可以看出，每个 FileSplit 的起始偏移量和上面 HDFS
每个文件块一致。但是具体读数据的时候，MapReduce
是如何处理的呢？我们现在已经知道，在将文件存储在 HDFS 的时候，文件被切割成一个一个 HDFS Block，其中会导致一些逻辑上属于一行的数据会被切割成两部分，那 TextInputFormat 遇到这样的数据是如何处理的呢？

对于这种情况，TextInputFormat 会做出如下两种操作：

- 在初始化 LineRecordReader 的时候，如果 FileSplit 的起始位置 start 不等于0， 说明这个 Block 块不是第一个 Block，这时候一律丢掉这个 Block 的第一行数据。
- 在读取每个 Block 的时候，都会额外地多读取一行，如果出现数据被切割到另外一个 Block 里面，这些数据能够被这个任务读取。

使用图形表示可以概括如下：



说明：

- 图中的红色虚线代表 HDFS 块边界(block boundary)；
- 蓝色的虚线代表Split 读数的边界。

从图中可以清晰地看出：

- 当程序读取 Block 0 的时候，虽然第五行数据被分割并被存储在 Block 0 和 Block 1 中，但是，当前程序能够完整的读取到第五行的完整数据。
- 当程序读取 Block 1 的时候，由于其 FileSplit 的起始位置 start 不等于0，这时候会丢掉第一行的数据，也就是说 Block 1 中的第五行部分数据会被丢弃，而直接从第六行数据读取。这样做的原因是，Block 1 中的第五行部分数据在程序读取前一个 Block 的时候已经被读取了，所以可以直接丢弃。
- 其他剩下的 Block 读取逻辑和这个一致。

总结

从上面的分析可以得出以下的总结

- Split 和 HDFS Block 是一对多的关系；
- HDFS block 是数据的物理表示，而 Split 是 block 中数据的逻辑表示；
- 满足数据本地性的情况下，程序也会从远程节点上读取少量的数据，因为存在行被切割到不同的 Block 上。