深入理解Hadoop Streaming

Hadoop Streaming 是 Hadoop 提供的一个 MapReduce 编程工具，它允许用户使用任何可执行文件、脚本语言或其他编程语言来实现 Mapper 和 Reducer 作业。比如下面的例子

```
mapred streaming ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper /bin/cat ₩
  -reducer /usr/bin/wc
```

Hadoop Streaming程序是如何工作的

Hadoop Streaming 使用了 Unix 的标准输入输出作为 Hadoop 和其他编程语言的开发接口，因此在其他的编程语言所写的程序中，只需要将标准输入作为程序的输入，将标准输出作为程序的输出就可以了。

在上面的示例中，mapper 和 reducer 都是能够从stdin逐行（line by line）读取输入的可执行文件，然后把处理完的结果发送到stdout。这个实用工具将会创建 一个 Map / Reduce 作业，并将作业提交到适当的集群，监控作业的运行进度直到作业运行完成。

如果一个文件（可执行或者脚本）作为 mapper，mapper 初始化时，每一个 mapper 任务会把该文件作为一个单独进程启动，mapper 任务运行时，它把输入切分成行并把每一行提供给可执行文件进程的标准输入。 同时，mapper 收集可执行文件进程标准输出的内容，并把收到的每一行内容转化成 key/value 对，作为 mapper 的输出。 默认情况下，一行中第一个 tab 之前的部分作为 key，之后的（不包括tab）作为 value。如果没有 tab，整行作为 key 值，value 值为 null。

reducer的运行过程和这个类似，就不介绍。

以上是 Map/Reduce 框架和 streaming mapper/reducer 之间的基本通信协议。

用户可以定义 stream.non.zero.exit.is.failure 参数为 true 或者 false 以定义一个以非0状态退出的 streaming 的任务是失败(Failure)还是成功(Success)。默认情况下，以非0状态退出的任务都任务是失败的。

Streaming命令行选项(Streaming Command Options)

Hadoop Streaming除了支持流命令选项（Streaming Command

Options）外，还支持Hadoop的通用命令选项（generic command options），通用命令选项这个会在本文的下面进行介绍。命令得使用规则如下：

mapred streaming [genericOptions] [streamingOptions]

需要注意的是，在提交Streaming作业中使用到通用命令选项的时候，需要把通用命令选项设置在流命令选项之前，否则将会出现一些错误。

目前的 Hadoop streaming （Hadoop 3.0.0）支持的流命令选项如下：

参数	是否可选	描述
-input directoryname or filename	Required	mapper的输入路径
-output directoryname	Required	reducer输出路径
-mapper executable or JavaClassName	Optional	Mapper可执行程序或 Java 类名
-reducer executable or JavaClassName	Optional	Reducer 可执行程序或 Java 类名
-file filename	Optional	mapper, reducer 或 combiner 依赖的文件
-inputformat JavaClassName	Optional	key/value 输入格式，默认为 TextInputFormat
-outputformat JavaClassName	Optional	key/value 输出格式，默认为 TextOutputformat
-partitioner JavaClassName	Optional	Class that determines which reduce a key is sent to
-combiner streamingCommand or JavaClassName	Optional	map 输出结果执行 Combiner 的命令或者类名
-cmdenv name=value	Optional	环境变量
-inputreader	Optional	向后兼容，定义输入的 Reader 类，用于取代输出格式
-verbose	Optional	输出日志

-lazyOutput	Optional	延时输出
-numReduceTasks	Optional	定义 reduce 数量
-mapdebug	Optional	map 任务运行失败时候，执行的脚本
-reducedebug	Optional	reduce 任务运行失败时候，执行的脚本

指定一个Java类作为Mapper/Reducer

我们可以指定一个Java类作为Mapper/Reducer，使用如下：

```
mapred streaming ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -inputformat org.apache.hadoop.mapred.KeyValueTextInputFormat ₩
  -mapper org.apache.hadoop.mapred.lib.IdentityMapper ₩
  -reducer /usr/bin/wc
```

提交作业的时候打包文件

正如上面介绍的，我们可以指定任意的可执行文件作为 mapper 或者 Reduce。在提交Hadoop Streaming作业的时候， mapper 或者 Reduce程序不需要事先部署在Hadoop集群的任意一台机器上，我们仅仅需要在提交Streaming作业的时候指定 -file
参数，这样Hadoop会自动将这些文件分发到集群。使用如下：

```
mapred streaming ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper myPythonScript.py ₩
  -reducer /usr/bin/wc ₩
  -file myPythonScript.py
```

上面命令中-file myPythonScript.py会导致Hadoop将这个文件自动分发到集群。

除了可以指定可执行文件之外，我们还可以打包 mapper 或者 Reduce
程序会用到的文件（包括目录，配置文件等），比如：

```
mapred streaming ₩
```

```
-input myInputDirs ₩
-output myOutputDir ₩
-mapper myPythonScript.py ₩
-reducer /usr/bin/wc ₩
-file myPythonScript.py ₩
-file myDictionary.txt
```

为作业指定其他插件

与正常的 Map / Reduce 作业一样，我们还可以为流式作业指定其他插件，选项如下：

```
-inputformat JavaClassName
-outputformat JavaClassName
-partitioner JavaClassName
-combiner streamingCommand or JavaClassName
```

我们为-inputformat指定的class文件必须返回Text类型的 key/value 键值对。如果你没有指定 input format 类，默认使用的是TextInputFormat类。TextInputFormat中key的返回类型是Long Writable，这个并不是输入数据的一部分，所以key部分将会被忽略，而仅仅返回value部分。

为-outputformat指定的class文件接收的数据类型是Text类型的 key/value 键值对。如果我们没有指定 output format 类，默认使用TextOutputFormat。

设置环境变量

我们可以在提交Streaming作业的时候设置环境变量，使用如下：

```
-cmdenv EXAMPLE_DIR=/home/example/dictionaries/
```

通用命令选项(Generic Command Options)

在提交流作业的时候，可支持的通用命令选项主要有以下几个：

参数	是否可选	描述
-conf configuration_file	Optional	定义应用的配置文件

-D property=value	Optional	定义参数
-fs host:port or local	Optional	定义 namenode 地址
-files	Optional	定义需要拷贝到 Map/Reduce 集群的文件，多个文件以逗号分隔
-libjars	Optional	定义需要引入到 classpath 的 jar 文件，多个文件以逗号分隔
-archives	Optional	定义需要解压到计算节点的压缩文件，多个文件以逗号分隔

通过-D选项指定配置变量

我们可以通过-D <property>=<value>的方式指定额外的配置变量(configuration variables)。

指定目录

为了改变默认的本地临时目录，可以使用下面的命令：

-D dfs.data.dir=/tmp

增加额外的本地临时目录可以使用下面命令：

-D mapred.local.dir=/tmp/local
-D mapred.system.dir=/tmp/system
-D mapred.temp.dir=/tmp/temp

设置只有Map的作业

有时候我们仅仅想跑只有Map的Hadoop作业，只需要将 mapreduce.job.reduces 设置为0即实现。这会导致Map/Reduce框架不会启动Reduce类型的task。map task的输出就是作业的最终结果输出，设置如下：

-D mapreduce.job.reduces=0

为了向后兼容，Hadoop Streaming还支持-reducer NONE选项，其含义等同于-D mapreduce.job.reduces=0。

设置Reduce的个数

下面例子中将程序的reduce个数设置为2：

```
mapred streaming ₩
  -D mapreduce.job.reduces=2 ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper /bin/cat ₩
  -reducer /usr/bin/wc
```

自定义行行数据如何拆分成Key/Value键值对

本文开头介绍过，当Map/Reduce框架从stdout读取行数据的时候，它会把一行数据拆分成一个key/value键值对。默认情况下，tab制表符分割的前一部分数据是作为key的；后一部分数据作为value。当然，我们可以自定义行数据的分隔符。如下所示：

```
mapred streaming ₩
  -D stream.map.output.field.separator=. ₩
  -D stream.num.map.output.key.fields=4 ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper /bin/cat ₩
  -reducer /bin/cat
```

在上面例子中，stream.map.output.field.separator指定.为 key 和 value的分隔符。

使用大文件或归档文件

我们可以使用-files 和 -archives 选项分别指定文件或者归档文件(archives)，这些文件可以被tasks使用。使用这个选项时，需要我们把这些文件或者archives上传到HDFS。这些文件在作业执行的时候会被缓存到所有的jobs中。

Making Files Available to Tasks

-files选项会在当前tasks的工作目录(current working directory)下创建一个符号链接(symlink)，这个链接指定的就是从HDFS拷贝文件的副本。下面例子中，我们指定了HDFS上的testfile.txt文件

，在使用-files选项之后，其会在Tasks的当前工作目录下创建名为testfile.txt的符号链接。

-files hdfs://host:fs_port/user/testfile.txt

当然，我们也可以自己通过#设置符号链接的名字：

-files hdfs://host:fs_port/user/testfile.txt#testfile

如果需要指定多个文件，使用如下：

-files hdfs://host:fs_port/user/testfile1.txt,hdfs://host:fs_port/user/testfile2.txt

Making Archives Available to Tasks

-archives选项允许我们指定一些压缩好的文件（比如jar、tgz），这些压缩文件会被拷贝到Tasks的当前工作目录，然后会被自动解压。在下面的例子中，我们指定了HDFS上的iteblog.jar压缩文件，Hadoop会自动为我们在Tasks的当前工作目录下创建一个名为iteblog.jar的符号链接。这个链接指定的是解压之后的文件夹名称：

-archives hdfs://host:fs_port/user/iteblog.jar

同样，我们也可以自己设置符号链接的名字：

-archives hdfs://host:fs_port/user/iteblog.tgz#tgzdir

下面的例子中，input.txt文件里面只有两行数据，分别是两个文件的名字：
cachedir.jar/cache.txt 和
cachedir.jar/cache2.txt；cachedir.jar是符号链接，其目录下包含了两个文件：cache.txt 和
cache2.txt

mapred streaming ₩

```
        -archives 'hdfs://iteblog.com/user/me/samples/cachefile/cachedir.jar' ₩
        -D mapreduce.job.maps=1 ₩
        -D mapreduce.job.reduces=1 ₩
        -D mapreduce.job.name="Experiment" ₩
        -input "/user/me/samples/cachefile/input.txt" ₩
        -output "/user/me/samples/cachefile/out" ₩
        -mapper "xargs cat" ₩
        -reducer "cat"

$ ls test_jar/
cache.txt  cache2.txt

$ jar cvf cachedir.jar -C test_jar/ .
added manifest
adding: cache.txt(in = 30) (out= 29)(deflated 3%)
adding: cache2.txt(in = 37) (out= 35)(deflated 5%)

$ hdfs dfs -put cachedir.jar samples/cachefile

$ hdfs dfs -cat /user/me/samples/cachefile/input.txt
cachedir.jar/cache.txt
cachedir.jar/cache2.txt

$ cat test_jar/cache.txt
This is just the cache string

$ cat test_jar/cache2.txt
This is just the second cache string

$ hdfs dfs -ls /user/me/samples/cachefile/out
Found 2 items
-rw-r--r-* 1 me supergroup       0 2013-11-14 17:00 /user/me/samples/cachefile/out/_SUCCESS
-rw-
r--r-* 1 me supergroup      69 2013-11-14 17:00 /user/me/samples/cachefile/out/part-00000

$ hdfs dfs -cat /user/me/samples/cachefile/out/part-00000
This is just the cache string
This is just the second cache string
```

更多的使用例子

Hadoop Partitioner Class

Hadoop内置提供了一个名为
KeyFieldBasedPartitioner的类，这个类在很多程序中使用。这个类可以将 map
输出的内容按照分隔后的一定列，而不是整个 key 内容进行分区，例如：

```
mapred streaming ₩
  -D stream.map.output.field.separator=. ₩
  -D stream.num.map.output.key.fields=4 ₩
  -D map.output.key.field.separator=. ₩
  -D mapreduce.partition.keypartitioner.options=-k1,2 ₩
  -D mapreduce.job.reduces=12 ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper /bin/cat ₩
  -reducer /bin/cat ₩
  -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

map.output.key.field.separator=.：设置 map 输出分区时 key 内部的分割符为 .
mapreduce.partition.keypartitioner.options=-k1,2：设置按前两个字段分区
mapreduce.job.reduces=12：reduce 数为12
比如上面例子map输出的key如下：

```
11.12.1.2
11.14.2.3
11.11.4.1
11.12.1.1
11.14.2.2
```

按照前两个字段进行分区，则会分为三个分区：

```
11.11.4.1
-----------
11.12.1.2
11.12.1.1
-----------
11.14.2.3
11.14.2.2
```

在每个分区内对整行内容排序后为：

```
11.11.4.1
----------
11.12.1.1
11.12.1.2
----------
11.14.2.2
11.14.2.3
```

Hadoop Comparator Class

Hadoop 中有一个类 KeyFieldBasedComparator，提供了 Unix/GNU
中排序的一部分特性。使用如下：

```
mapred streaming ₩
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.
KeyFieldBasedComparator ₩
  -D stream.map.output.field.separator=. ₩
  -D stream.num.map.output.key.fields=4 ₩
  -D mapreduce.map.output.key.field.separator=. ₩
  -D mapreduce.partition.keycomparator.options=-k2,2nr ₩
  -D mapreduce.job.reduces=1 ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper /bin/cat ₩
  -reducer /bin/cat
```

mapreduce.partition.keycomparator.options=-k2,2nr：指定第二个字段为排序字段，-n
是指按自然顺序排序，-r 指倒叙排序。
比如上面例子map输出的key如下：

```
11.12.1.2
11.14.2.3
11.11.4.1
11.12.1.1
11.14.2.2
```

那么Reduce的输出结果如下

11.14.2.3
11.14.2.2
11.12.1.2
11.12.1.1
11.11.4.1

Hadoop Aggregate Package

Hadoop 中有一个类 Aggregate，Aggregate 提供了一个特定的 reduce 类和 combiner 类，以及一些对 reduce 输出的聚合函数，例如 sum、min、max等等。为了使用 Aggregate，我们只需要定义 -reducer aggregate参数，如下：

```
mapred streaming ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper myAggregatorForKeyCount.py ₩
  -reducer aggregate ₩
  -file myAggregatorForKeyCount.py ₩
```

myAggregatorForKeyCount.py 文件大概内容如下：

```
#!/usr/bin/python

import sys;

def generateLongCountToken(id):
    return "LongValueSum:" + id + "₩t" + "1"

def main(argv):
    line = sys.stdin.readline();
    try:
        while line:
            line = line&#91;:-1];
            fields = line.split("₩t");
            print generateLongCountToken(fields&#91;0]);
            line = sys.stdin.readline();
    except "end of file":
        return None
```

```
if __name__ == "__main__":
    main(sys.argv)
```

Hadoop Field Selection Class

Hadoop 中有一个类 FieldSelectionMapReduce，运行你像 unix 中的 cut 命令一样处理文本。使用如下：

```
mapred streaming ₩
  -D mapreduce.map.output.key.field.separator=. ₩
  -D mapreduce.partition.keypartitioner.options=-k1,2 ₩
  -D mapreduce.fieldsel.data.field.separator=. ₩
  -D mapreduce.fieldsel.map.output.key.value.fields.spec=6,5,1-3:0- ₩
  -D mapreduce.fieldsel.reduce.output.key.value.fields.spec=0-2:5- ₩
  -D mapreduce.map.output.key.class=org.apache.hadoop.io.Text ₩
  -D mapreduce.job.reduces=12 ₩
  -input myInputDirs ₩
  -output myOutputDir ₩
  -mapper org.apache.hadoop.mapred.lib.FieldSelectionMapReduce ₩
  -reducer org.apache.hadoop.mapred.lib.FieldSelectionMapReduce ₩
  -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

mapreduce.fieldsel.map.output.key.value.fields.spec=6,5,1-3:0-：意思是 map 的输出中 key 部分包括分隔后的第 6、5、1、2、3列，而 value 部分包括分隔后的所有的列
mapreduce.fieldsel.reduce.output.key.value.fields.spec=0-2:5-：意思是 map 的输出中 key 部分包括分隔后的第 0、1、2列，而 value 部分包括分隔后的从第5列开始的所有列