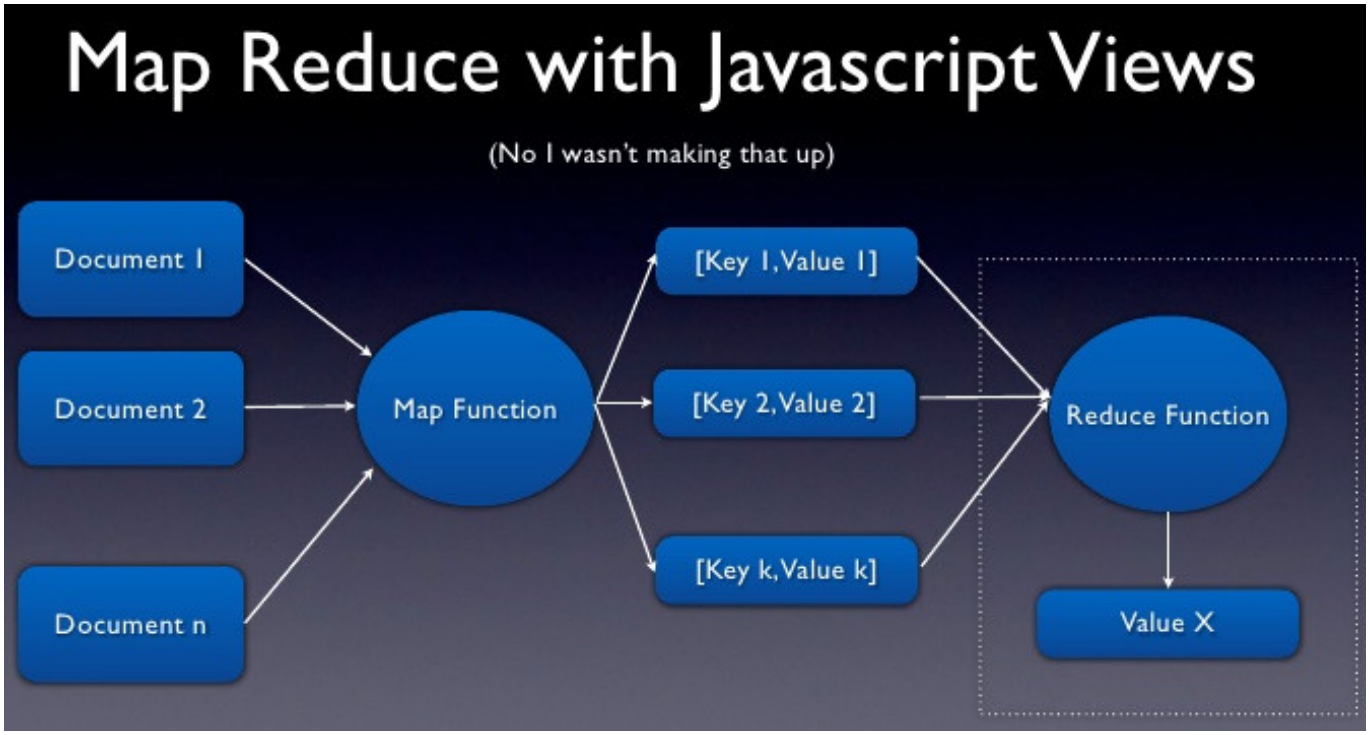
使用JavaScript编写MapReduce程序并运行在Hadoop集群上

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，从而充分利用 Hadoop 并行计算框架的优势和能力，来处理大数据。而我们在官方文档或者是Hadoop权威指南看到的Hadoop Streaming例子都是使用 Ruby 或者 Python 编写的，官方说可以使用任何可执行文件的方式来编写Hadoop Streaming程序，那么我们一定可以使用JavaScript语言来编写Hadoop Streaming程序。



如果想及时了解Spark、Hadoop、Flink或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

为了构建Hadoop Streaming程序，我们需要分别编写 mapper 程序和 reducer程序，这两个程序分别都可以从标准输入读取用户数据，出来完只会会发送到标准输出。本文我们使用NodeJS来从标准输入输出读写数据，并使用它完成一个经典的WordCount程序，我们编写的 mapper 程序如下：

```
#!/usr/bin/env node

// 引入readline模块
const readline = require('readline')

// 创建readline接口实例
```

```
const rl = readline.createInterface({
    input:process.stdin,
    output:process.stdout
})

rl.on('line', line => {
    // 分离每一行的单词
    line.split(' ').map((word) => {
        // 将单词以如下格式写入标准输出
        console.log(`${word}₩t1`)
    })
})

rl.on("close", () => {
    process.exit(0)
})
```

reduce 中的代码如下：

```
#!/usr/bin/env node
const readline = require('readline')

const  rl = readline.createInterface({
    input:process.stdin,
    output:process.stdout,
    terminal: false
})

// 存储键值对 <string , Number>
let words = new Map()

rl.on('line', line => {
    // 解构赋值
    const [word, count] = line.split('₩t')
    // 如果 Map 中没有该单词，则将该单词放入 Map ，即第一次添加
    if (!words.has(word)) {
        words.set(word, parseInt(count))
    } else {
        // 如果该单词已存在，则将该单词对应的 count 加 1
        words.set(word, words.get(word) + 1)
    }
})
```

```
rl.on("close", () => {
    words.forEach((v, k) => {
        // 将统计结果写入标准输出
        console.log(`${k}₩t${v}`)
    })
    process.exit(0)
})
```

map 和 reduce 程序写完之后，我们需要准备一个文件（iteblog.txt），这个文件就是我们的输入数据，内容随便：

```
hadoop streaming
spark iteblog
spark
hadoop
```

现在程序和数据已经准备好了，我们可以现在本地电脑试试能不能运行这个程序，依次操作如下：

```
$ chmod +x mapper reduce
```

然后将iteblog.txt中的内容作为map的输入，而map的输出作为reduce的输入，如下：

```
$ cat /tmp/iteblog.txt | ./mapper | ./reduce
hadoop  2
streaming   1
spark   2
iteblog 1
```

程序正确地统计出各个单词出现的频率。接下来我们使用Hadoop Streaming来提交刚刚写的程序，并让它在分布式环境下执行，依次操作如下：

```
#将待处理文件上传至 HDFS：
$ hadoop fs -put iteblog.txt /tmp/
```

```
#检查文件是否上传成功
$ hadoop fs -ls /tmp/iteblog.txt
Found 1 items
-rw-r--r--   3 iteblog supergroup         28 2017-03-06 18:52 /tmp/iteblog.txt
```

数据准备好之后，现在可以提交到Hadoop集群执行程序了：

```
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.2.0.jar  ₩
        -input /tmp/iteblog.txt                                  ₩
        -output /tmp/output/                                      ₩
        -mapper mapper                                           ₩
        -reducer reduce                                         ₩
        -file /home/iteblog/mapper                                   ₩
        -file /home/iteblog/reduce
```

检查计算结果：

```
$ hadoop fs -cat /tmp/output/*
streaming   1
hadoop  2
spark   2
iteblog 1
```

可以看到在Hadoop上面执行这个程序也一样可以得到正确的结果。

因为最终的 mapper 和 reduce程序是在Hadoop集群上运行的，而我们上面的JavaScript程序依赖了NodeJS，所以我们必须在Hadoop集群每个节点上安装配置好NodeJS，否则运行上面的程序将会出现下面的异常：

```
/usr/bin/env: node: No such file or directory
```