实验七

**Part 1：**

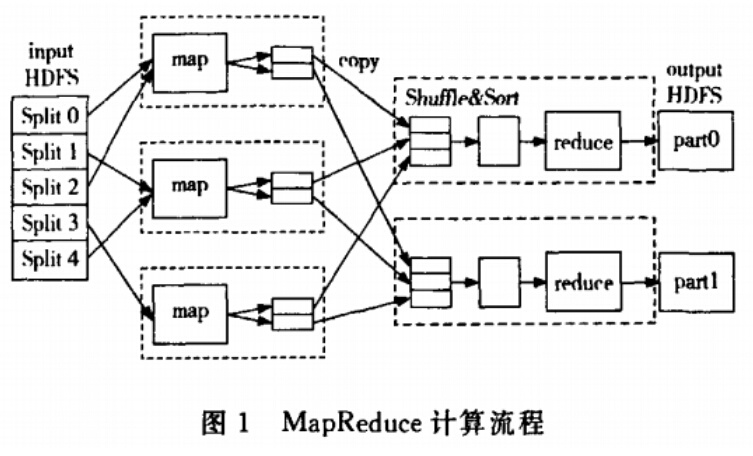
**实验目的**

了解map/reduce和hadoop的基本原理，完成简单的wordcount实验。

**实验内容**

**1、map/reduce**

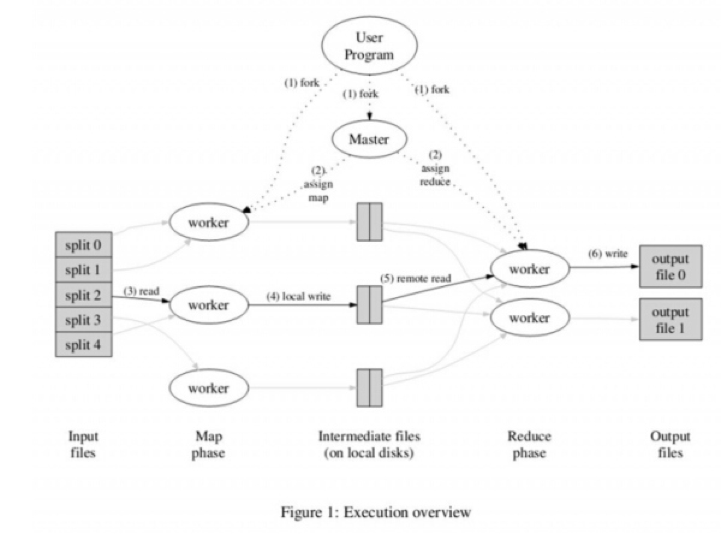
MapReduce是Google提出的分布式并行计算编程模型，用于大数据并行处理。其受函数式编程语言的启发，将大规模数据作业拆分为若干个可独立运行的Map任务，在不同的机器当中执行任务并生成中间文件，再有若干个Reduce任务合并这些中间文件获得最后的输出文件。用户在使用MapReuce时只需要将主要精力放在如何编写Map函数和Reduce函数上，其他计算机系统中的复杂问题（分布式文件，并发，调度，容错，通信等问题）交给系统来处理，很大程度上降低了编程难度。当下，MapReduce日益成为云计算平台的主流编程模型。可以用一个简单的例子来解释：我们想要统计图书馆的所有书籍，你数shelf#1，我数shelf#2，这就是map，我们的人越多，数的速度就越快。然后，我们把每个人数的结果加起来，这就是reduce。

​​

Map函数与Reduce函数

· map函数：接受一个键值对（key-value pair），产生一组中间键值对。MapReduce框架会将map函数产生的中间键值对里键相同的值传递给一个reduce函数。

· reduce函数：接受一个键，以及相关的一组值，将这组值进行合并产生一组规模更小的值（通常只有一个或零个值）。

​​

​

对于上面这个MapReduce过程详解图：

最初：由最上方的user program作为开始（其链接了MapReduce库，实现了基本的Map函数与Reduce函数）

①：MapReduce库将从user program中输入的程序划分为若干份。将用户进程fork到不同的机器中。

②：在MapReduce系统中有一个为Master，负责调度下面运行的worker，如果存在空闲的worker，则向其分配作业（Map作业或是Reduce作业），其中Master只有一个，worker的数量可以由用户指定。

③：每一个非空闲状态下的worker执行map作业，分别读取每一个worker所对应的输入作业。Map作业从输入数据中抽取键值对，产生的中间键值对被缓存在内存中。

④：在上一步中所缓存的中间键值对被定期写入磁盘，用户会事先指定一个数量R：这R个区中的每个区都会对应一个Reduce作业。

⑤：Master将中间键值对的位置发送给Reduce worker。

⑥：Reduce worker把所有它负责的中间键值对都读过来，首先进行排序以聚集相同键的键值。

⑦：reduce worker 遍历中间键值对，将唯一的键的键值传递给reduce函数。Reduce函数的输出会被输出到该分区的输出文件中。

⑧：当MapReduce整个作业完成，master唤醒正在等待的user program。

**hadoop**

Hadoop是一个由Apache基金会所开发的分布式系统基础架构。

用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。

Hadoop实现了一个分布式文件系统（Hadoop Distributed File System），简称HDFS。Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，则MapReduce为海量的数据提供了计算。

Hadoop原本来自于谷歌一款名为MapReduce的编程模型包。谷歌的MapReduce框架可以把一个应用程序分解为许多并行计算指令，跨大量的计算节点运行非常巨大的数据集。使用该框架的一个典型例子就是在网络数据上运行的搜索算法。Hadoop最初只与网页索引有关，迅速发展成为分析大数据的领先平台。

特点：

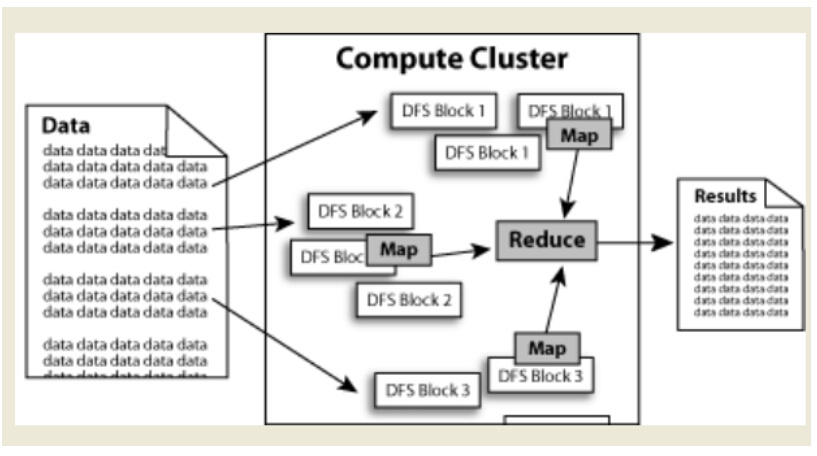
1.对大量数据进行分布式处理的软件框架

2.可靠，通过假定潜在的失败发生，Hadoop维护多个副本，对失败的节点可以重新处理。

3.高效：并发加快了Hadoop的处理速度，在节点之间动态移动数据，保证节点的动态平衡。

4.可伸缩：可伸缩性使其可以处理PB级的数据。

下图展示了hadoop的基本操作过程：

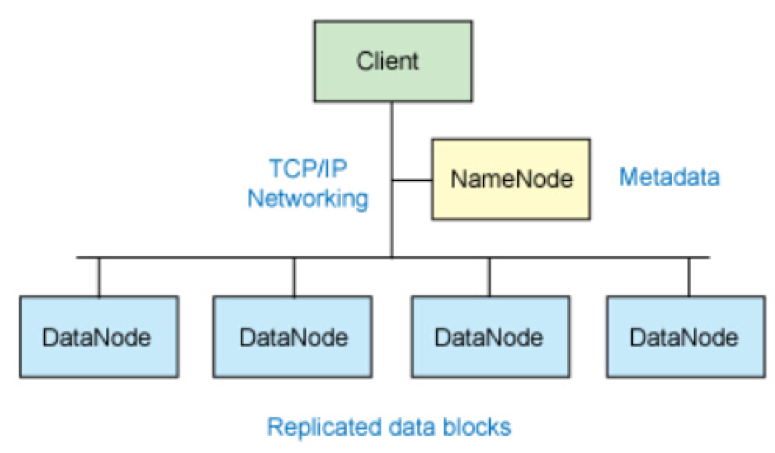
​​

Hadoop 框架

* 最底层：Hadoop Distributed File System（HDFS），用来存储Hadoop集群中所有存储节点上的文件。
* 向上一层：Map/Reduce引擎。

HDFS（Hadoop Distributed File System）

* 在用户视野中：HDFS像是一个传统的文件分级系统的存在，可以进行和个人计算机中的文件管理器类似的操作（增删改查等）。
* 底层构建：基于一组特定的节点所构建，如图:

​​

包括一个NameNode以及若干DataNode. NameNode用于提供元数据服务，DataNode用于为HDFS提供存储块。

NameNode（管理者）

* 包括一个NameNode以及若干DataNode. NameNode用于提供元数据服务，DataNode用于为HDFS提供存储块。

DataNode（工作者）

* DataNode通常是在HDFS实例中的单独机器运行的软件。DataNode通过以机架的形式组织，并由一个交换机将所有的系统连接起来。Hadoop的一个假设：机架内节点间传输速度快于机架间节点间传输速度。根据需要存储并检索数据块，响应对数据块的处理操作命令。

​

map函数

#!/usr/bin/env python

import sys

​

for line in sys.stdin:

line = line.strip()

words = line.split()

for word in words:

print '%s\t%s' %(word, 1)



reduce函数

#!/usr/bin/env python

​

from operator import itemgetter

import sys

​

current\_word = None

current\_count = 0

word = None

​

for line in sys.stdin:

words = line.strip()

word, count = words.split('\t')

​

try:

count = int(count)

except ValueError:

continue

​

if current\_word == word:

current\_count += count

else:

if current\_word:

print '%s\t%s' %(current\_word, current\_count)

current\_count = count

current\_word = word

​

if current\_word == word:

print '%s\t%s' %(current\_word, current\_count)



run.sh

hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.8.4.jar \

-files ./mapper.py,./reducer.py \

-mapper ./mapper.py \

-reducer ./reducer.py \

-input /user/input/\*.txt \

-output /user/output\_tianyi



​

**Part 2：**

**实验目的**

了解hadoop的基本文件操作，以及编写简单的mappereduce实现股票价格分析

**实验内容**

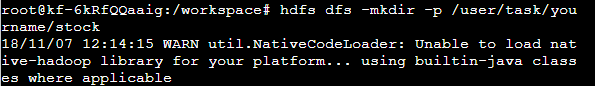
**一、HDFS基础操作**

1、hdfs的工作文件夹下创建以你的名字命名的文件夹

hdfs dfs -mkdir -p /user/task/yourname/stock



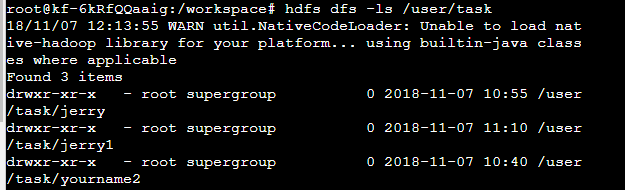
yourname 改成自己的名字

​​

2、查看文件夹下的内容

hdfs dfs -ls /user/task



​​

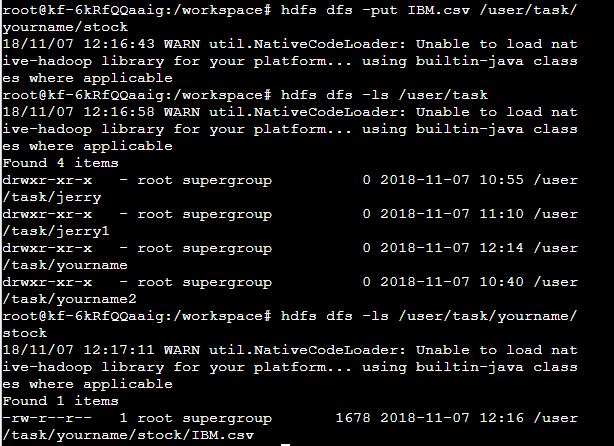
3、进入workspace，文件夹下,移动数据文件IBM.csv到input文件夹下

cd workspace



hdfs dfs -put IBM.csv /user/task/yourname/stock

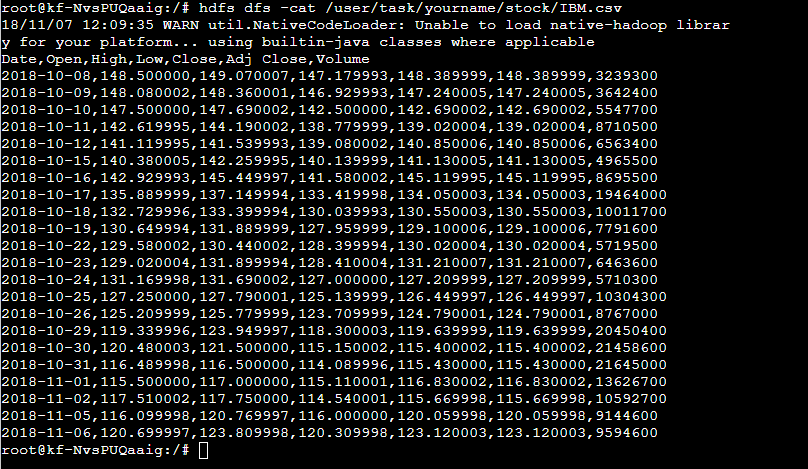


​​

4、打开一个已经存在的文件

hdfs dfs -cat /user/task/yourname/stock/IBM.csv



​​

5、将hadoop上某个文件down至本地已有目录下：hadoop dfs -get [文件目录] [本地目录]

hadoop dfs –get /user/task/yourname/IBM.csv /home



6、删除hadoop上指定文件：hdfs dfs –rm [文件地址]

hdfs dfs –rm /user/task/yourname/stock/IBM.csv



7、删除hadoop上指定文件夹（包含子目录等）

hdfs dfs –rmr /user/task/yourname



8、在hadoop指定目录下新建一个空文件

hdfs dfs -touchz /user/new.txt



9、将hadoop上某个文件重命名

hdfs dfs –mv /user/test.txt /user/ok.txt



10、将hadoop指定目录下所有内容保存为一个文件，同时down至本地

hdfs dfs –getmerge /user /home/t



\* 11、将正在运行的hadoop作业kill掉

hadoop job –kill [job-id]



12.查看帮助

hdfs dfs -help



**二、**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*编写两个代码文件： stock\_mapper.py和wordcount\_reduce.py，保存两个文件在workspace目录下 \*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vi stock\_mapper.p#!/usr/bin/python



# -\*- coding: utf-8 -\*-

​

import sys

​

# 从标准输入读取数据

line\_num = 0

for line in sys.stdin:

# Date,Open,High,Low,Close,Adj Close,Volume

line\_num = line\_num + 1

if(line\_num!=1):

​

row=line.split(',')

​

# Open Price

open\_price=float(row[1])

# Close Price

close\_price=float(row[-3])

​

# Compute Price Change

price\_change=((open\_price-close\_price)/open\_price)\*100

​

cheng\_text=str(round(price\_change, 2))+"%"

print "%s\t%d" % (cheng\_text, 1)

​



​



vi stock\_reducer.py



#!/usr/bin/python

# -\*- coding: utf-8 -\*-

​

import sys

​

# 定义变量存储 单词 和 词频

current\_word = None

current\_count = 1

​

for line in sys.stdin:

# 读取mapper函数输出的结果

word, count = line.strip().split("\t")

​

# 判断当前是否存单词

if current\_word:

if word == current\_word:

current\_count += int(count)

else:

print "%s\t%d" %(current\_word, current\_count)

current\_count = 1

# 赋值当前单词

current\_word = word

​

# 处理读物最后一行数据

if current\_count >= 1:

print "%s\t%d"%(current\_word,current\_count)

​



运行代码

more IBM.csv | python stock\_mapper.py | sort -k1| python stock\_reducer.py > stock-ibm.output

