BigData (Prof. Mrudula Mukadam)

Projects: MapReduce, Spark and Hbase

I. Map Reduce

This project is to calculate the relative frequency of co-occurrence of products that were purchased by customers. Example: The output's result can help in predicting the product that a custom would buy after buying a product.
The sample input text file for testing can be found in directory input

```
input/
    record.txt
```

This sample text file contains the following content

```
Mary 34 56 29 12 34 56 92 29 34 12
Kelly 92 29 12 34 79 29 56 12 34 18
```

Note:

- we are going to use this file for testing the 3 approaches: Pair, Stripe and Hybrid approaches
- number of reducer is set to 4 for all approaches

1. Setup

1.1 **Prerequisites**

- 64bit host that support virtualization
- Minimum of 4GB of RAM available

1.2 **Virtual Box**

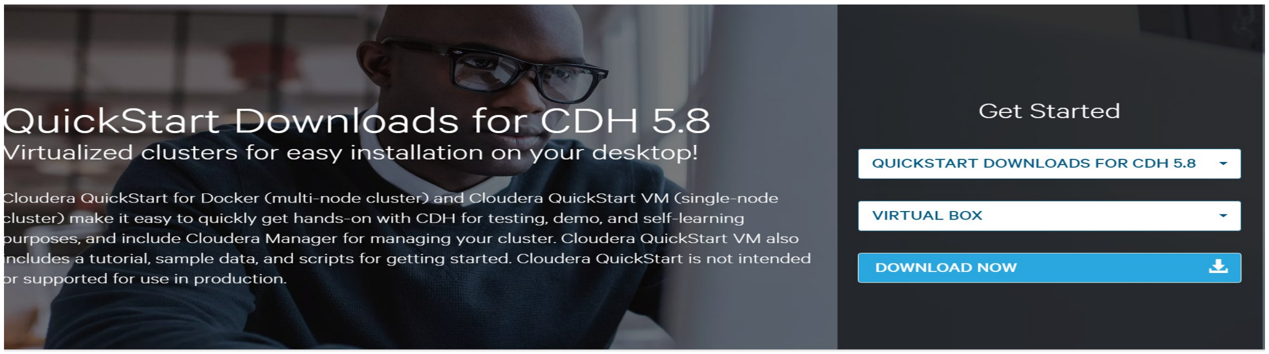
Dowanload the latest Oracle VirtualBox from the following link
https://www.virtualbox.org/wiki/Downloads
(At the time of making this document the latest version is *VirtualBox-5.1.6-110634*) then install the program

1.3 **Cloudera Quickstart**

This project, we are going to use Hadoop 2 in Cloudera Quickstart.
- Download Cloudera QuickStart for Hadoop from the following
http://www.cloudera.com/downloads/quickstart_vms/5-8.html

- Select the platform that you want to use (Here we are using Virtual Box)
- Click on DOWNLOAD NOW

– Sign in or fill out the product interest form to continue
– After the download completed, unzip the file, open the unzipped directory and click on the ovf file extension (eg: cloudera-quickstart-vm-5.8.0-0-virtualbox.ovf) this will taking care of setup the cloudera for virtual box.


QuickStart Downloads for CDH 5.8

Virtualized clusters for easy installation on your desktop!

Cloudera QuickStart for Docker (multi-node cluster) and Cloudera QuickStart VM (single-node cluster) make it easy to quickly get hands-on with CDH for testing, demo, and self-learning purposes, and include Cloudera Manager for managing your cluster. Cloudera QuickStart VM also includes a tutorial, sample data, and scripts for getting started. Cloudera QuickStart is not intended or supported for use in production.

2. Generating jar and Testings

Extract file from the folder hadoop.zip

input/	contains sample text file
partone/	class files
src/	source code (java files)
generateJar.sh	shell script for generating jar
runpair.sh	shell script for running pair approach
runstripe.sh	shell script for running stripe approach
runhybrid.sh	shell script for running hybrid approach
Result.txt	output result from shell scripts run for the 3 approac

2.1 Generating jar file

Go to unzipped folder (hadoop) right click → Open in Terminal

type: ./generateJar.sh

This should generate a jar class name hadoop1

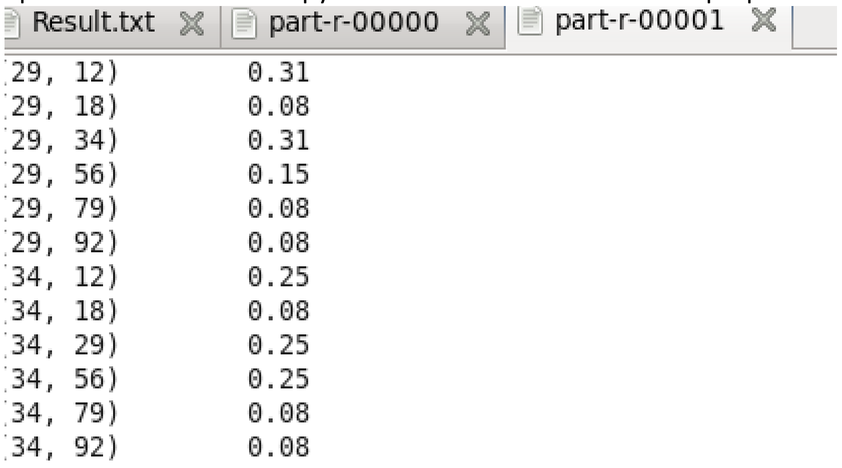
2.2 Running Script and Result

2.2.1 Pair

Go to unzipped folder (hadoop) right click → Open in Terminal

type: ./runpair.sh

This will process the data and copy the result from HDFS FS to outputpair directory.



```
29, 12)      0.31
29, 18)      0.08
29, 34)      0.31
29, 56)      0.15
29, 79)      0.08
29, 92)      0.08
34, 12)      0.25
34, 18)      0.08
34, 29)      0.25
34, 56)      0.25
34, 79)      0.08
34, 92)      0.08
```

pair's output result from reducer2

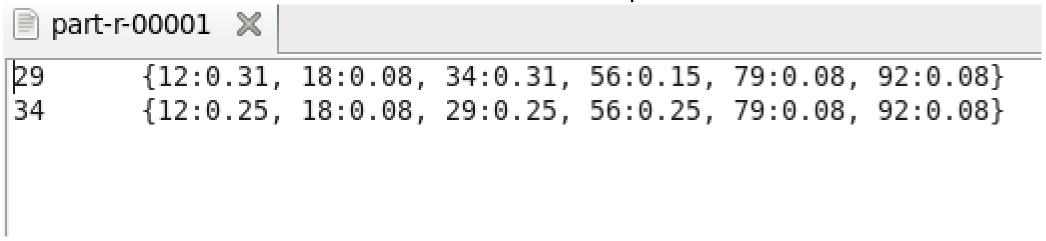
2.2.2 Stripe

Go to unzipped folder (hadoop) right click → Open in Terminal

type: ./runstripe.sh

This will process the data and copy the result from HDFS FS to outputstripe directory

Ex: here is the result from one of the reducer 2 output



part-r-00001

```
29	{12:0.31, 18:0.08, 34:0.31, 56:0.15, 79:0.08, 92:0.08}
34	{12:0.25, 18:0.08, 29:0.25, 56:0.25, 79:0.08, 92:0.08}
```

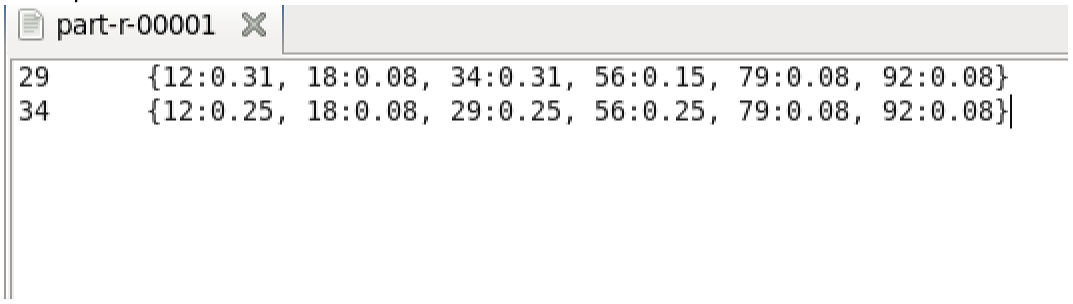
2.2.3 Hybrid

Go to unzipped folder (hadoop) right click → Open in Terminal

type: ./runhybrid.sh

This will process the data and copy the result from HDFS FS to outputhybrid directory.

output result from reducer2



part-r-00001

```
29	{12:0.31, 18:0.08, 34:0.31, 56:0.15, 79:0.08, 92:0.08}
34	{12:0.25, 18:0.08, 29:0.25, 56:0.25, 79:0.08, 92:0.08}
```

3. **Comparison**

	GC time elapsed (ms)	**CPU time spent (ms)**	**Physical memory (bytes) snapshot**	**Virtual memory (bytes) snapshot**
Pair	693	3800	696680448	7528714240
Stripe	735	4380	710283264	7528988672
Hybrid	638	3890	706478080	7529783296

Note: Pair's cpu time spent is less than Stripe. As we have learnt stripe is faster than pair, but this due to the fact that our input result contain only two lines of text.

Details output result can be found in Result.txt

II. Spark

Introduction

This application is written in scala and is used to display the unique ip address and count number of request between March, 8th 2004 and March, 11th 2004.
The sample input file is taken from Apache Access log file.
We are using cloudera, so all the requirements such as Apache hadoop, spark or maven installation won't be mention here.

Instruction:

Step 1: building jar

In SparkProject directory, right click => Open in Terminal

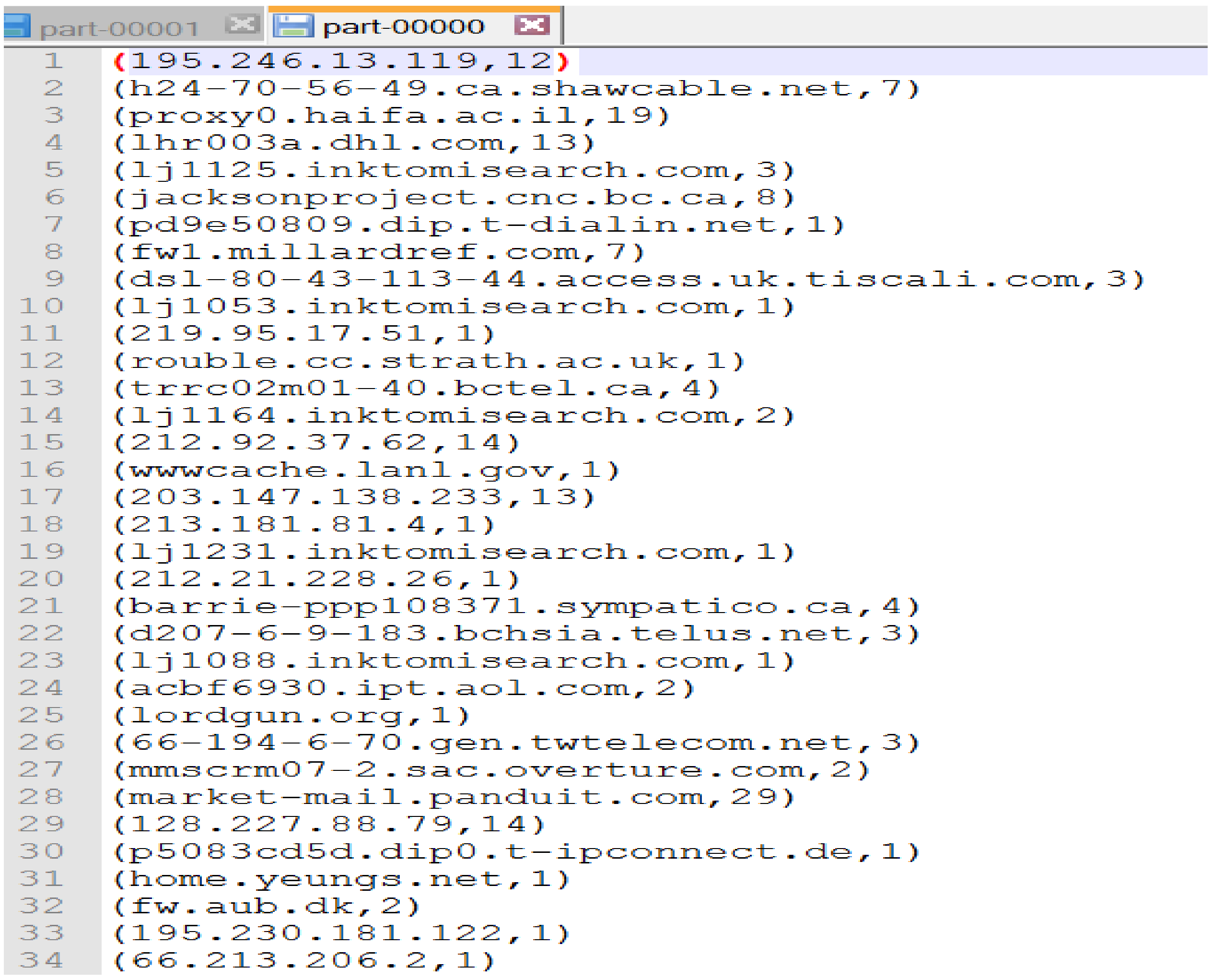
Type : ./build.sh

Note: here we are using bash script to build jar file and clasess using maven build, which mean you can build this project from any machine that have maven installed

Step 2: run the program

Type : ./run.sh

Number of reducer is 3: *This screenshot is the result from reducer 2*



```
(195.246.13.119,12)
(h24-70-56-49.ca.shawcable.net,7)
(proxy0.haifa.ac.il,19)
(lhr003a.dhl.com,13)
(lj1125.inktomisearch.com,3)
(jacksonproject.cnc.bc.ca,8)
(pd9e50809.dip.t-dialin.net,1)
(fw1.millardref.com,7)
(dsl-80-43-113-44.access.uk.tiscali.com,3)
(lj1053.inktomisearch.com,1)
(219.95.17.51,1)
(rouble.cc.strath.ac.uk,1)
(trrc02m01-40.bctel.ca,4)
(lj1164.inktomisearch.com,2)
(212.92.37.62,14)
(wwwcache.lanl.gov,1)
(203.147.138.233,13)
(213.181.81.4,1)
(lj1231.inktomisearch.com,1)
(212.21.228.26,1)
(barrie-ppp108371.sympatico.ca,4)
(d207-6-9-183.bchsia.telus.net,3)
(lj1088.inktomisearch.com,1)
(acbf6930.ipt.aol.com,2)
(lordgun.org,1)
(66-194-6-70.gen.twtelecom.net,3)
(mmscrm07-2.sac.overture.com,2)
(market-mail.panduit.com,29)
(128.227.88.79,14)
(p5083cd5d.dip0.t-ipconnect.de,1)
(home.yeungs.net,1)
(fw.aub.dk,2)
(195.230.181.122,1)
(66.213.206.2,1)
```

Description:

src/	source code
target/	classes files and other
input/	contains access_log file
build.sh	script file for building classes and jar
run.sh	script for processing log file and generate output files

III. Hbase

Introduction

The application is written in Java using Hbase and MapReduce. It will :

- create a sample table called "Employee" and its column families
- add some data to table Employee
- output a result that count current employee position

Instruction:

Step 1: building jar

right click => Open in Terminal

Type : ./build.sh

Note: here we are using bash script to build jar file and clasess using maven build, which mean you can build this project from any machine that have maven installed

Step 2: run the program

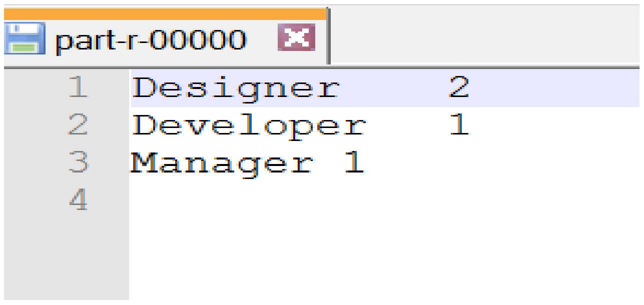
Type : ./run.sh

Number of reducer is 3: *This screenshot is the result from reducer 2*

Description:

src/	*source code*
target/	*classes files and other*
outputhbase/	*contain output result*
build.sh	*script file for building classes and jar*
run.sh	*script for processing log file and generate output files*

Here is the output result



```
Designer	2
Developer	1
Manager 1
```

***PS**: Remember if you are running these project in IDE, not matter how many number of reducer you are set you will see only one number of reducer, because it's local mode.*