CCBDA HW4

游翔竣 311511056

1.

- **Infrastructure as a Service(IaaS)**：基礎設施即服務，是消費者使用處理、儲存、網路以及各種基礎運算資源，部署與執行作業系統或應用程式等各種軟體。客戶端無須購買伺服器、軟體等網路設備，即可任意部署和運行處理、存儲、網絡和其它基本的計算資源，不能控管或控制底層的基礎設施，但是可以控制作業系統、儲存裝置、已部署的應用程式，有時也可以有限度地控制特定的網路元件，像是主機端防火牆。

- **Platform as a Service(PaaS)**：平台即服務，是一種雲端運算服務，提供運算平台與解決方案服務。在雲端運算的典型層級中，PaaS層介於軟體即服務與基礎設施即服務之間。

- **Software as a Service(SaaS)**：軟體即服務，有時被作為「即需即用軟體」（即「一經要求，即可使用」）提及，它是一種軟體交付模式。在這種交付模式中雲端集中式代管軟體及其相關的資料，軟體僅需透過網際網路，而不須透過安裝即可使用。

- 若台灣電費昂貴，則「IaaS」可能較不適合，因為IaaS業者使用伺服器或其他硬體設施等電費成本可能會因此而提高服務價格。

2

- 若要成立一家對花朵分類的公司，可能會需要採購大量的伺服器設備以應付訓練機器學習所需要的大量運算，若不自行採購可能就需要向雲端運算服務公司購買服務，而目前最廣為人之的雲端服務就是google的GCP，但目前大部分公司多半使用微軟的Azure(市占率約為10.9%)，而Amazon的AWS為最價格最高的，若考慮費用的話我可能

會傾向於選擇google的GCP

	GCP	AWS	Microsoft Azure
地區	美國	美國	美國
機器類型	n1-standard-1	m3.medium	D1 v2
虛擬 CPU 數	1	1	1
記憶體	3.75 GB	3.75 GB	3.5 GB
作業系統	Linux	Linux	Linux
硬碟空間	50 GB	50 GB	50 GB
硬碟費用	USD$2	USD$5	已包含
主機費用	USD$34.67	USD$49.05	USD$52.08
每月費用 (主機+硬碟)	USD$26.27 (持續使用折扣[註2-1])	USD$54.05	USD$52.08
優惠後費用 (主機+硬碟)	USD$23.84 (承諾使用折扣[註2-2]—年)	USD$40.04 (預留執行個體一年無需預付)	USD$52.08 (暫無優惠)

source (https://mile.cloud/zh/resources/blog/cloud-platform-traffic-cost-comparison-google-GCP-Amazon-Microsoft_185)

3

- **Docker**：
  - 較方便使用，快速，不同container之間可以達到環境隔離，
  - 且佔用記憶體資源較少，
  - 但仍然依賴於原主機的作業系統。
- **VM**：
  - 安全性較高，作業系統的選擇較彈性
  - 應用程式不須要被拆分，因此不需要大幅更改應用程式的架構。簡單來說不需要降低應用程式內服務的耦合性，不需要將程式內的服務個別拆開來部屬。
  - 但佔用記憶體資源較高，且執行速度較慢
- 適合doker但vm不適用的場景舉例來說像是自駕車系統，自駕車通常需要包含許多不同的套件與函式庫(ex:ROS)，使用docker便可以在不同需要的自駕場域切換不同需求的cotainer，若使用vm可能會因為執行速度較慢而產生延遲。

4. Exercise 1

- reducer

```
#!/usr/bin/python

import sys


salesTotal = 0
oldKey = None


# Loop around the data
# It will be in the format key\tval
# Where key is the store name, val is the sale amount
#
# All the sales for a particular store will be presented,
# then the key will change and we'll be dealing with the next store



for line in sys.stdin:
        data_mapped = line.strip().split("\t")
        if len(data_mapped) != 2:
                # Something has gone wrong. Skip this line.
                continue

        thisKey, thisSale = data_mapped

        if oldKey and oldKey != thisKey:
                print oldKey, "\t", salesTotal

                oldKey = thisKey
                salesTotal = 0


        oldKey = thisKey
        salesTotal += float(thisSale)


if oldKey != None:
        print oldKey, "\t", salesTotal
```

- mapper

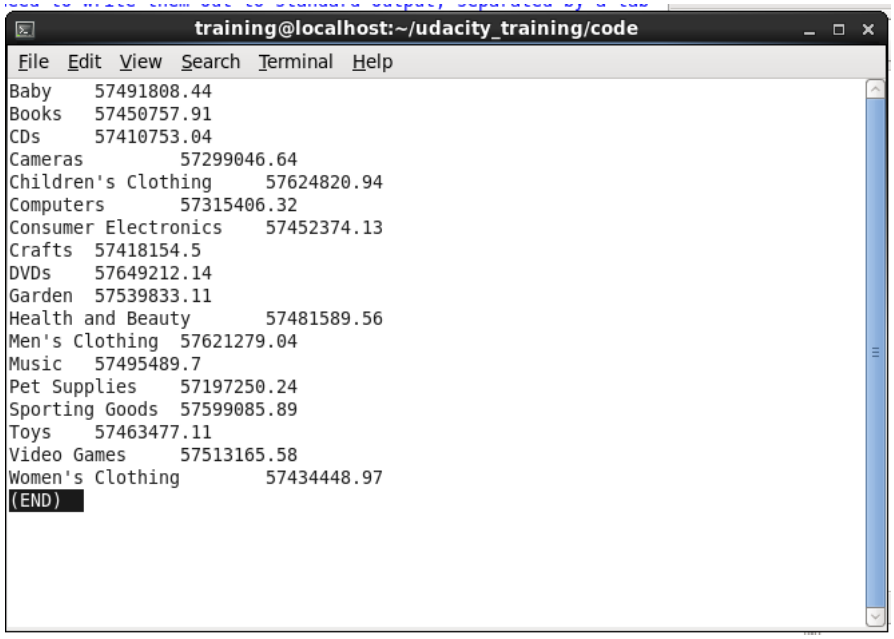
```
#!/usr/bin/python

# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab

import sys

for line in sys.stdin:
        data = line.strip().split("\t")
        if len(data) == 6:
                date, time, store, item, cost, payment = data
                #print "{0}\t{1}".format(store, cost)
                print "{0}\t{1}".format(item, cost)
```

- result



5. Exercise 2

- reducer

```
import sys


salesTotal = 0
oldKey = None
highestSale = 0.

# Loop around the data
# It will be in the format key\tval
# Where key is the store name, val is the sale amount
#
# All the sales for a particular store will be presented,
# then the key will change and we'll be dealing with the next store



for line in sys.stdin:
        data_mapped = line.strip().split("\t")
        if len(data_mapped) != 2:
                # Something has gone wrong. Skip this line.
                continue

        thisKey, thisSale = data_mapped
        thisSale = float(thisSale)
        if oldKey and oldKey != thisKey:
                print oldKey, "\t", highestSale

                oldKey = thisKey
                #salesTotal = 0
                highestSale = 0

        oldKey = thisKey
        #salesTotal += float(thisSale)
        if float(thisSale) > highestSale:
                highestSale = float(thisSale)

if oldKey != None:
        print oldKey, "\t", highestSale
        #print oldKey, "\t", salesTotal
```

- mapper

```
#!/usr/bin/python

# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab

import sys

for line in sys.stdin:
        data = line.strip().split("\t")
        if len(data) == 6:
                date, time, store, item, cost, payment = data
                print "{0}\t{1}".format(store, cost)
```

- result

```
Albuquerque     499.98
Anaheim         499.98
Anchorage       499.99
Arlington       499.95
Atlanta         499.96
Aurora  499.97
Austin  499.97
Bakersfield     499.97
Baltimore       499.99
Baton Rouge     499.98
Birmingham      499.99
Boise   499.98
Boston  499.99
Buffalo         499.99
Chandler        499.98
Charlotte       499.98
Chesapeake      499.98
Chicago         499.99
Chula Vista     499.99
Cincinnati      499.98
Cleveland       499.98
Colorado Springs        499.99
Columbus        499.98
```

6. Exercise 3

- reducer

```
#!/usr/bin/python

import sys


salesTotal = 0.
oldKey = None
salesCount = 0
# Loop around the data
# It will be in the format key\tval
# Where key is the store name, val is the sale amount
#
# All the sales for a particular store will be presented,
# then the key will change and we'll be dealing with the next store



for line in sys.stdin:
        data_mapped = line.strip().split("\t")
        if len(data_mapped) != 2:
                # Something has gone wrong. Skip this line.
                continue

        thisKey, thisSale = data_mapped
        thisSale = float(thisSale)
        if oldKey and oldKey != thisKey:
                #print oldKey, "\t", highestSale

                oldKey = thisKey

        oldKey = thisKey
        salesTotal += thisSale
        salesCount += 1

if oldKey != None:
         print "Total sale across all the stores:" , salesTotal
         print "Total number of sales: ", salesCount
```

- mapper

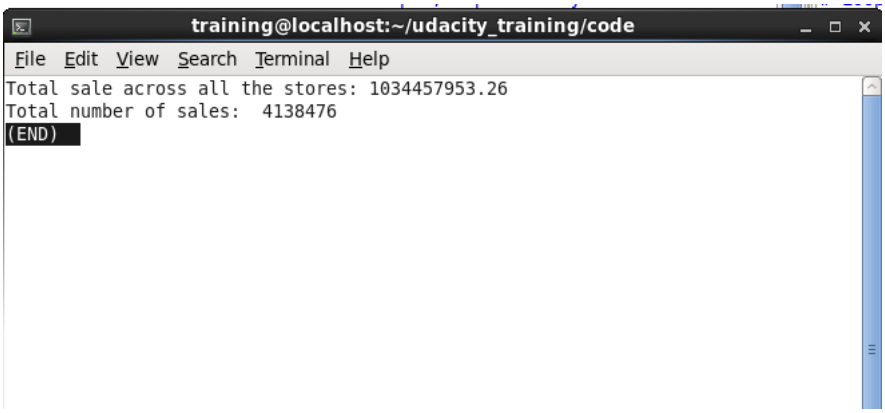
```
#!/usr/bin/python

# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab

import sys

for line in sys.stdin:
        data = line.strip().split("\t")
        if len(data) == 6:
                date, time, store, item, cost, payment = data
                print "{0}\t{1}".format(item, cost)
```

- result



7.

- reducer

```
#!/usr/bin/python

import sys
salesCount = 0
salesTotal = 0
oldKey = None
salesMean = 0.
sales = []
for line in sys.stdin:
        data_mapped = line.strip().split("\t")
        if len(data_mapped) != 2:
                # Something has gone wrong. Skip this line.
                continue
        thisKey, thisSale = data_mapped
        thisSale = float(thisSale)
        if oldKey and oldKey != thisKey:
                sum_ = 0.
                salesVar = 0.
                salesMean =  salesTotal / salesCount
                for i in sales:
                        sum_ += (i - salesMean)**2
                salesVar = sum_ / salesCount
                print oldKey, "\t", salesVar
                oldKey = thisKey
                salesTotal = 0
                salesCount = 0
                sales = []
        sales.append(thisSale)
        oldKey = thisKey
        salesTotal += thisSale
        salesCount += 1
if oldKey != None:
        sum_ = 0.
        salesVar = 0.
        salesMean =  salesTotal / salesCount
        for i in sales:
                sum_ += (i - salesMean)**2
        salesVar = sum_ / salesCount
        print oldKey , "\t" , salesVar
```

- mapper

```
#!/usr/bin/python

# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab

import sys

for line in sys.stdin:
        data = line.strip().split("\t")
        if len(data) == 6:
                date, time, store, item, cost, payment = data
                print "{0}\t{1}".format(store, cost)
```

- result

```
training@localhost:~/udacity_traini
File  Edit  View  Search  Terminal  Help
Albuquerque     20863.564927
Anaheim         20831.6107064
Anchorage       20759.3012781
Arlington       20966.6398122
Atlanta         20728.7602526
Aurora  20692.9646915
Austin  20808.4259878
Bakersfield     20731.7423748
Baltimore       20996.1867555
Baton Rouge     20843.9500752
Birmingham      20779.824888
Boise   20900.4489806
Boston  20852.7740002
Buffalo         20968.4153964
Chandler        20791.4898805
Charlotte       21063.0266127
Chesapeake      20722.5537571
Chicago         20825.0823641
Chula Vista     20916.254721
Cincinnati      20855.2909207
Cleveland       20596.1849171
Colorado Springs        21044.353937
Columbus        20765.6563682
:
```