



# Hadoop & MapReduce

Hong-Han Shuai

National Yang Ming Chiao Tung University

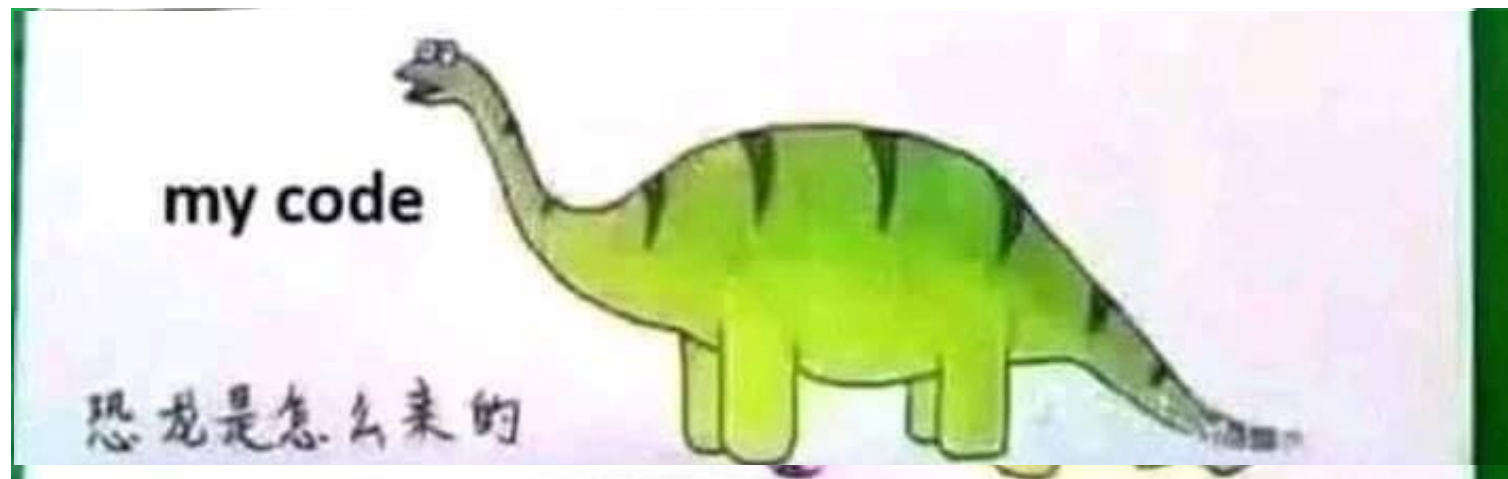
Spring, 2021

# How to create dinosaurs?

Using Python to write  
MapReduce and Spark  
Hadoop Streaming API

# Hadoop Streaming API

---



# Preparation

---



- Virtual Machine
  - <http://0rz.tw/uhmu9>
- Cloudera
  - [https://drive.google.com/file/d/1EmLszJ5e99ds2CH-sBu\\_ftnujFKaqqvq/view](https://drive.google.com/file/d/1EmLszJ5e99ds2CH-sBu_ftnujFKaqqvq/view)
  - Rename as training.exe
  - Double click (Unzip)
  - Machine -> Add
  - <https://drive.google.com/file/d/1h0rkmea8yQVDaX3PClAFz02Ny8Jm38-z/view?usp=sharing>

# Questions before we go through the code



<https://kahoot.it/>

# Block size of HDFS

---

- The default data block size of HDFS is 64/128MB.
  - If you store a file that's 1k or 60Mb, it'll take up one block. Once you cross the 64Mb boundary, you need a second block.
- The block size in disk is generally 4KB.
- A 1000Mb file.
  - With a 4k block size, you'd have to make 256,000 requests to get that file (1 request per block).
  - In HDFS, those requests go across a network and come with a lot of overhead. Each request has to be processed by the Name Node to figure out where that block can be found. That's a lot of traffic! If you use 128Mb blocks, the number of requests goes down to 7, greatly reducing the cost of overhead and load on the Name Node.

# Walking through the code

---

- Calculating the total sales per store
  - Scoop can be used for importing the database
- Input data
  - 2012-01-01 12:01 San Jose Music 12.99 Amex
- How to find the total sales per store?
- Here are the options:

	KEY	VALUE
(a)	time	store
(b)	cost	store
(c)	store name	cost
(d)	store name	product type



# Defensive mapper code

---



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

```
2012-01-01 12:01 San Jose Music 12.99 Amex  
2012-01-02 There was an error trying to connect to the database.
```

# Defensive mapper code

---

```
def mapper():  
    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)
```

```
2012-01-01 12:01 San Jose Music 12.99 Amex  
2012-01-02 There was an error trying to connect to the database.
```

# Between Map and Reduce

---



- Questions: What happens between generate  $\langle \text{key}, \text{value} \rangle$  pairs and reducer?

# Reducing

---

- Reducer will get data coming in something like this

Miami 12.34

Miami 99.07

Miami 97.87

NYC 99.77

NYC 88.99

- Hadoop streaming is used here for writing in Python.

# Question

---



- What variable we need to keep track of to calculate the total sales per store?
  - (a) Previous cost
  - (b) current cost
  - (c) total sales per store
  - (d) previous store
  - (e) current store
  - (f) all store names

# Reducer Code



```
def reducer():  
    salesTotal = 0  
    oldKey = None  
  
    for line in sys.stdin:  
        data = line.strip().split("\t")  
  
        if len(data) != 2:  
            continue  
  
        thisKey, thisSale = data  
  
        if oldKey and oldKey != thisKey:  
            print "{0}\t{1}".format(oldKey, salesTotal)  
  
            salesTotal = 0  
  
        oldKey = thisKey  
        salesTotal += float(thisSale)
```

Miami	12.34
Miami	99.07
Miami	97.87
NYC	99.77
NYC	88.99

# Reducer Code



```
def reducer():
```

```
    salesTotal = 0
    oldKey = None
```

```
    for line in sys.stdin:
        data = line.strip().split("\t")
```

```
        if len(data) != 2:
            continue
```

```
        thisKey, thisSale = data
```

```
        if oldKey and oldKey != thisKey:
            print "{0}\t{1}".format(oldKey, salesTotal)
```

```
            salesTotal = 0
```

```
        oldKey = thisKey
        salesTotal += float(thisSale)
```

```
    if oldKey != None:
        print "{0}\t{1}".format(oldKey, salesTotal)
```

Miami	12.34
Miami	99.07
Miami	97.87
NYC	99.77
NYC	88.99

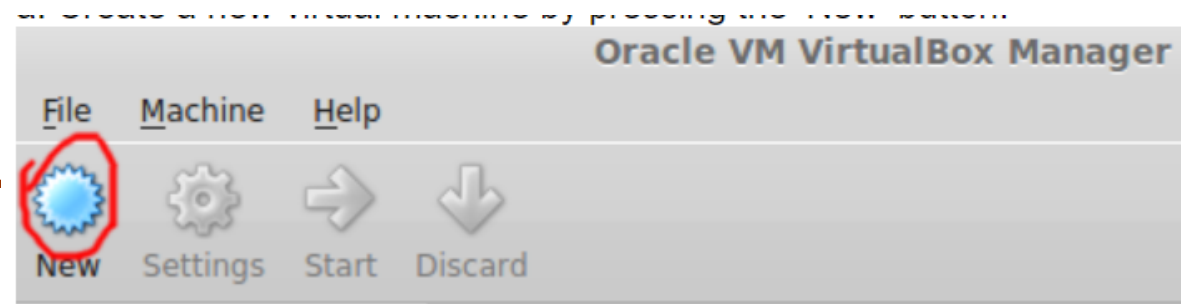
# Hadoop Streaming

---

- <https://hadoop.apache.org/docs/r1.2.1/streaming.html#Hadoop+Streaming>
- Hadoop can access stdin and stdout.
  - Use stdin and stdout as channel to combine Hadoop framework and python
- It is worth noting that mapper.py and reducer.py are stored in local, while **DATA** is stored in Hadoop environments (HDFS).
  - If the locations are incorrect, the program cannot be executed.
- Make sure the file has execution permission
  - If not, **chmod +x /home/hduser/mapper.py** should do the trick or you will run into problems.
- If you want to modify some Hadoop settings on the fly like increasing the number of Reduce tasks, you can use the -D option:



# Lab 1: Sales Statistics



## Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Hadoop

Type: Linux

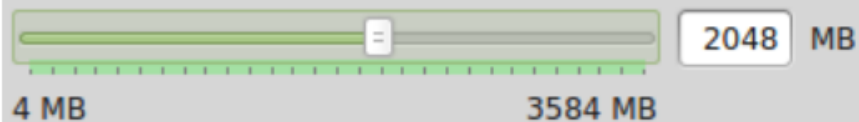
Version: Ubuntu

- b. Choose a name, use 'Type': 'Linux':
- c. Press Next

## Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **512 MB**.



- d. Select memory size for the VM.

# Upload purchases.txt to HDFS

---

- `cd udacity_training/data`
- **#Put purchase.txt into HDFS**
- `hadoop fs -ls`
- `hadoop fs -put purchases.txt`
- `hadoop fs -ls`
- **# Make directory and put txt into it**
- `hadoop fs -mkdir myinput`
- `hadoop fs -put purchases.txt myinput`
- **#Useful Comments**
- `hadoop fs -tail/hadoop fs -cat`
- Rename: `-mv`, Delete: `-rm` New directory: `-mkdir`

# Running a job

---

- cd code
- **#How to run?**
- **#Full command**
- `hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh4.1.1.jar -mapper mapper.py -reducer reducer.py -file mapper.py -file reducer.py -input myinput -output joboutput`
- **#Alias**
- `hs mapper.py reducer.py myinput joboutput`

# Find results

---



- **#List files**
- `hadoop fs -ls`
- **#Browse part of results**
- `hadoop fs -cat joboutput/part-00000 | less`
- **#Take the file out of HDFS**
- `hadoop fs -get joboutput/part-00000  
mylocalfile.txt`

# Pipe (stdin stdout)

---

- **#Find the results of first 40 lines and write to testfile**
- `head -50 ./data/purchases.txt > testfile`
- **#Read testfile and pipe to mapper.py**
- `cat testfile | ./mapper.py`
- **#Read testfile, pipe to mapper.py, sort, and pipe to reducer.py**
- `cat testfile | ./mapper.py | sort | ./reducer.py`

# Exercise 1

---

- Instead of breaking the sales down by store, give us a sales breakdown by product category across all of our stores.
- (Hint) variables should be modified with type casting before counting
- **[Implicit Example]**
  - `num_int = 123, num_flo = 1.23`
  - `num_new = num_int + num_flo`
  - `num_int = 123`
  - `num_str = "456"`
  - `num_int+ num_str`
- **[Explicit Example]**
  - `num_str = int(num_str), y=float(x)`

# Check point

---



Baby	57491808.44
Books	57450757.91
CDs	57410753.04
Cameras	57299046.64
Children's Clothing	57624820.94
Computers	57315406.32
Consumer Electronics	57452374.13
Crafts	57418154.5
DVDs	57649212.14
Garden	57539833.11
Health and Beauty	57481589.56
Men's Clothing	57621279.04
Music	57495489.7
Pet Supplies	57197250.24
Sporting Goods	57599085.89
Toys	57463477.11
Video Games	57513165.58
Women's Clothing	57434448.97



# Exercise 2 (2pts)

---



- Find the monetary value for the highest individual sale for each separate store.

# Check point



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

# Exercise 3 (2pts)

---



- Find the total sales value across all the stores, and the total number of sales. Assume there is only one reducer.

# Check point

---



- Total sales across all the stores: 1034457953.26
- Total number of sales: 4138476