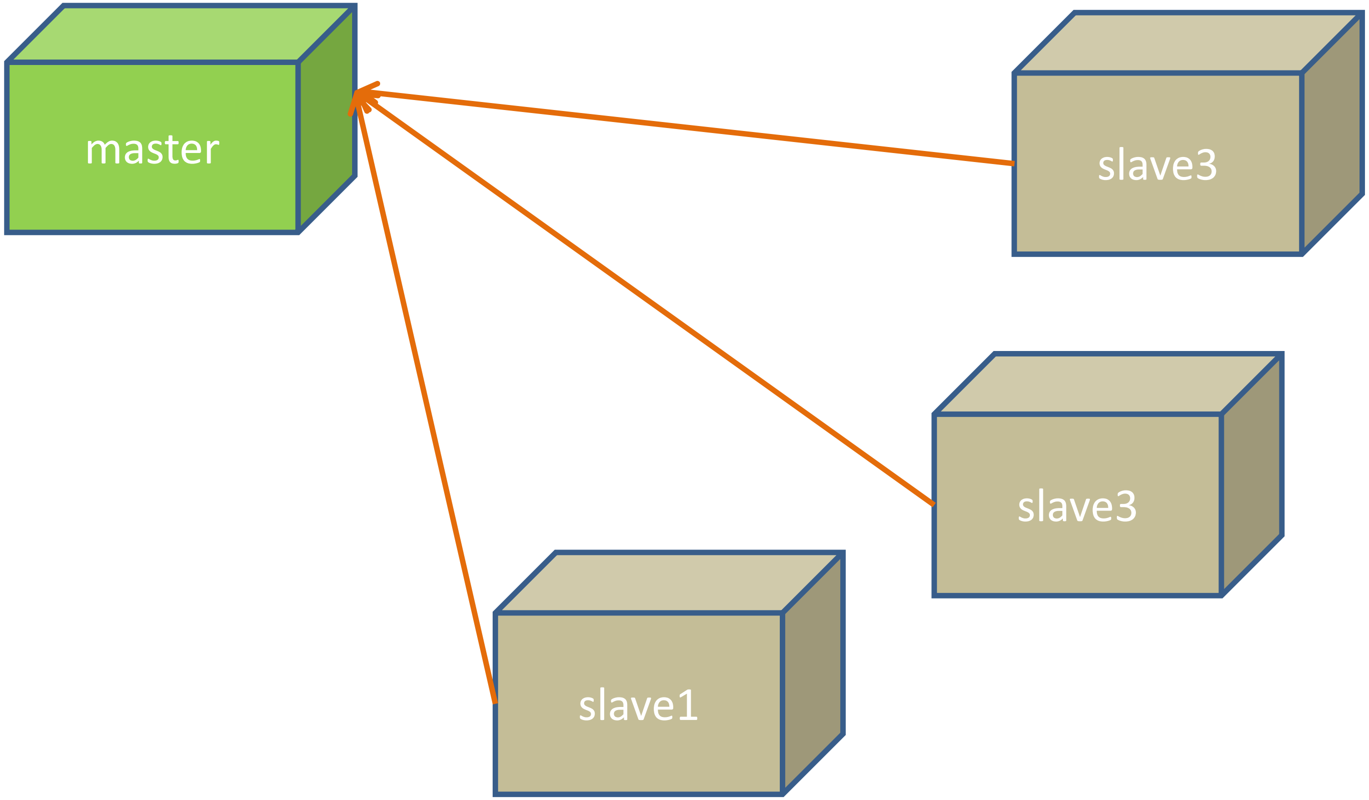
Hadoop MapReduce Programming

國網中心
莊家雋 博士

Distributed System

- Master /slave architecture

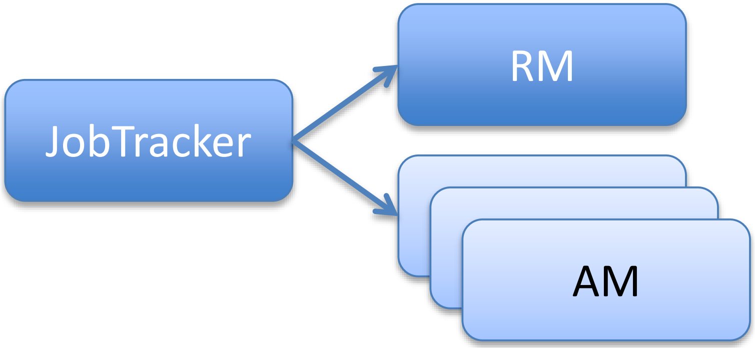


MRv1 v.s. MRv2

- Mapreduce 框架包含
  - 編程模型：map()與reduce()
    - MRv1與MRv2的程式寫法都相同
  - 運行環境：
    - MRv1：
      - JobTracker & TaskTracker
      - JobTracker同時負責資源管理與所有工作的控制
    - MRv2：
      - 由YARN提供
      - NodeManager & ResourceManager

MRv1 v.s. MRv2

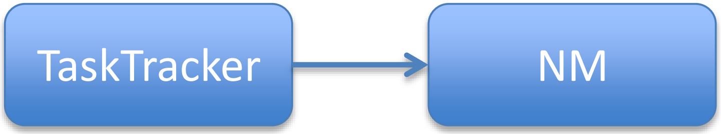
MRv1	Function	MRv2
JobTracker	Resource Management Scheduling	Resource Manager
	Job Management	Application Master
TaskTracker	Job Execution	Node Manager



JobTracker

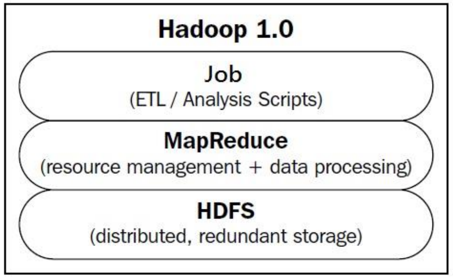
RM

AM



TaskTracker

NM

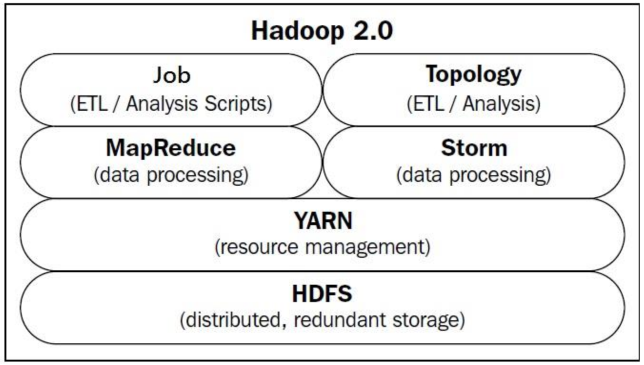


Hadoop 1.0

Job
(ETL / Analysis Scripts)

MapReduce
(resource management + data processing)

HDFS
(distributed, redundant storage)



Hadoop 2.0

Job
(ETL / Analysis Scripts)

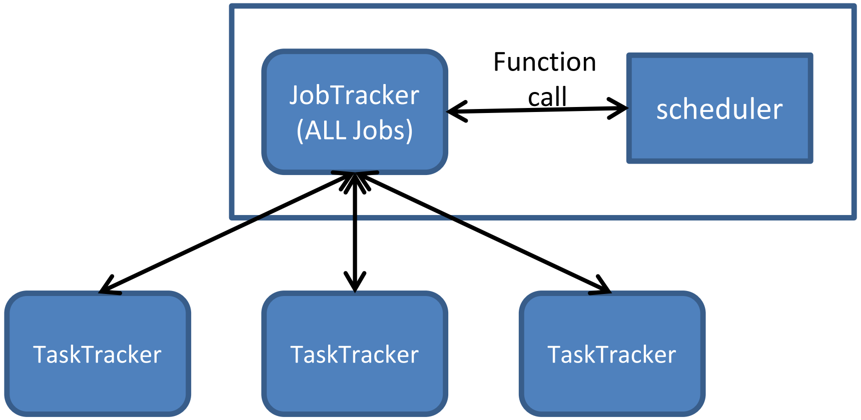
Topology
(ETL / Analysis)

MapReduce
(data processing)

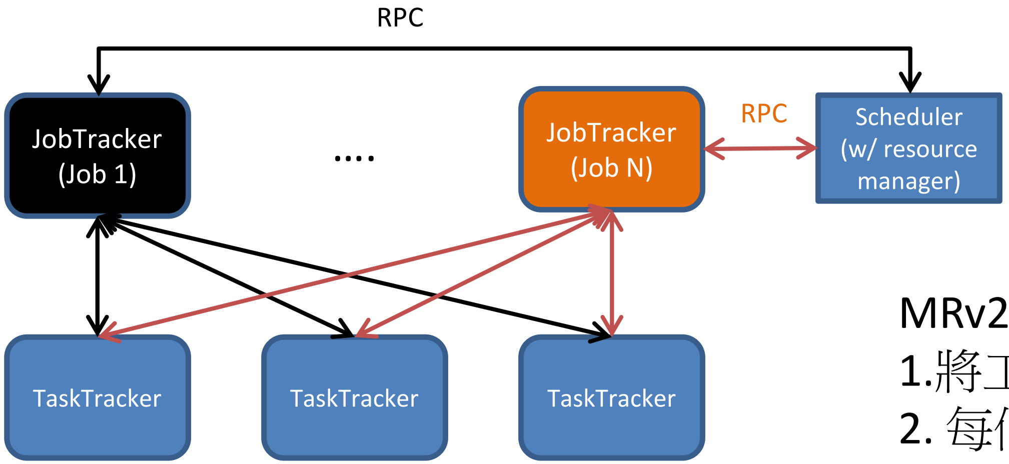
Storm
(data processing)

YARN
(resource management)

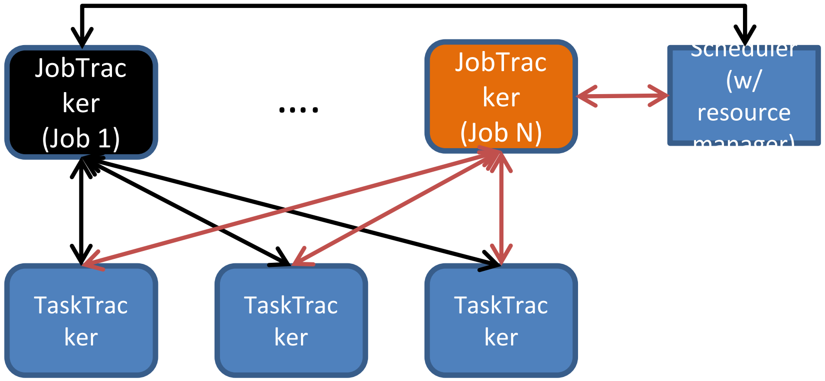
HDFS
(distributed, redundant storage)



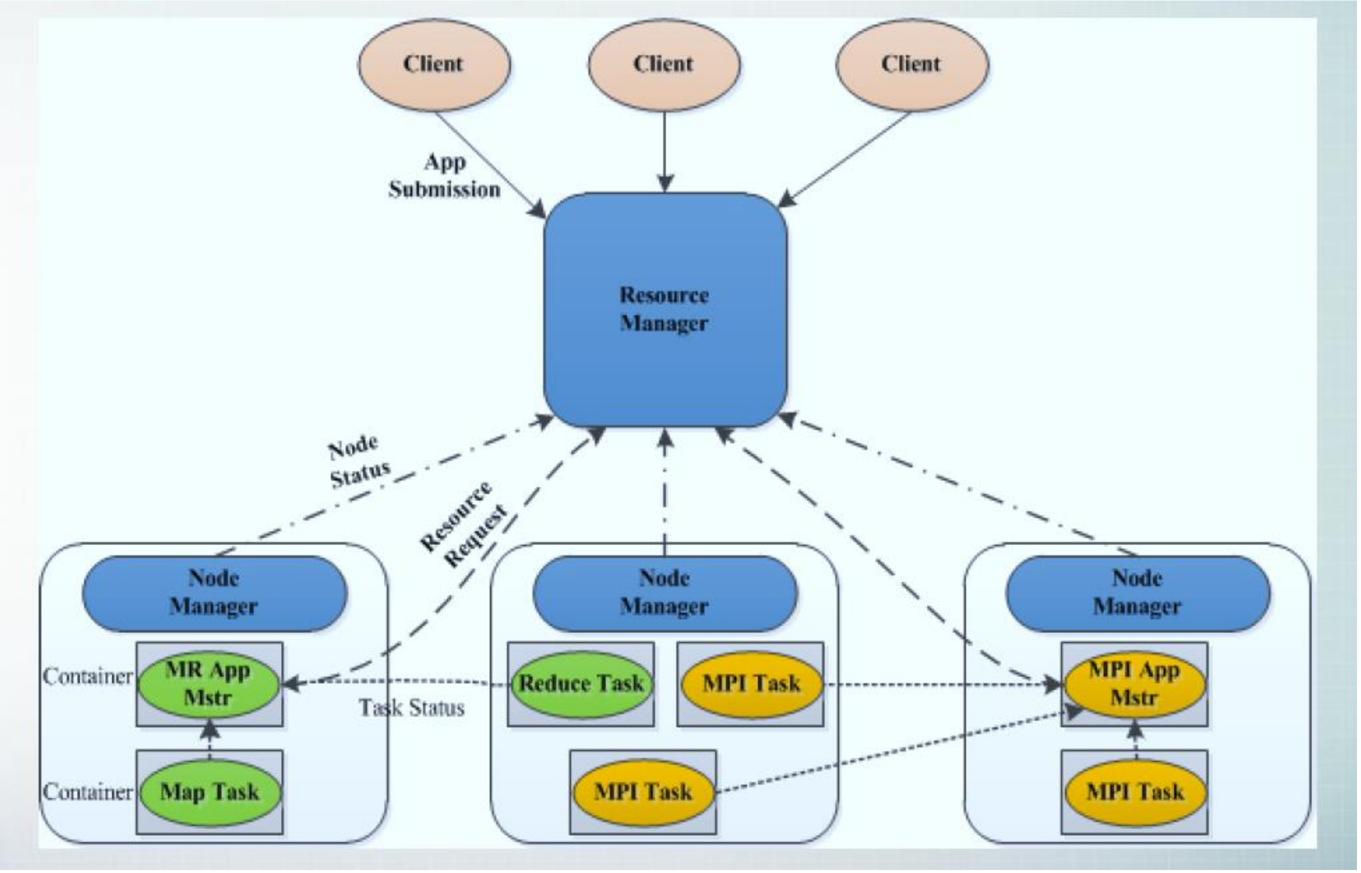
MRv1.
JobTracker負責工作控制與資源管理



MRv2.
1.將工作控制與資源管理分開
2. 每個Job有自己的JobTracker
3.全域的資源管理

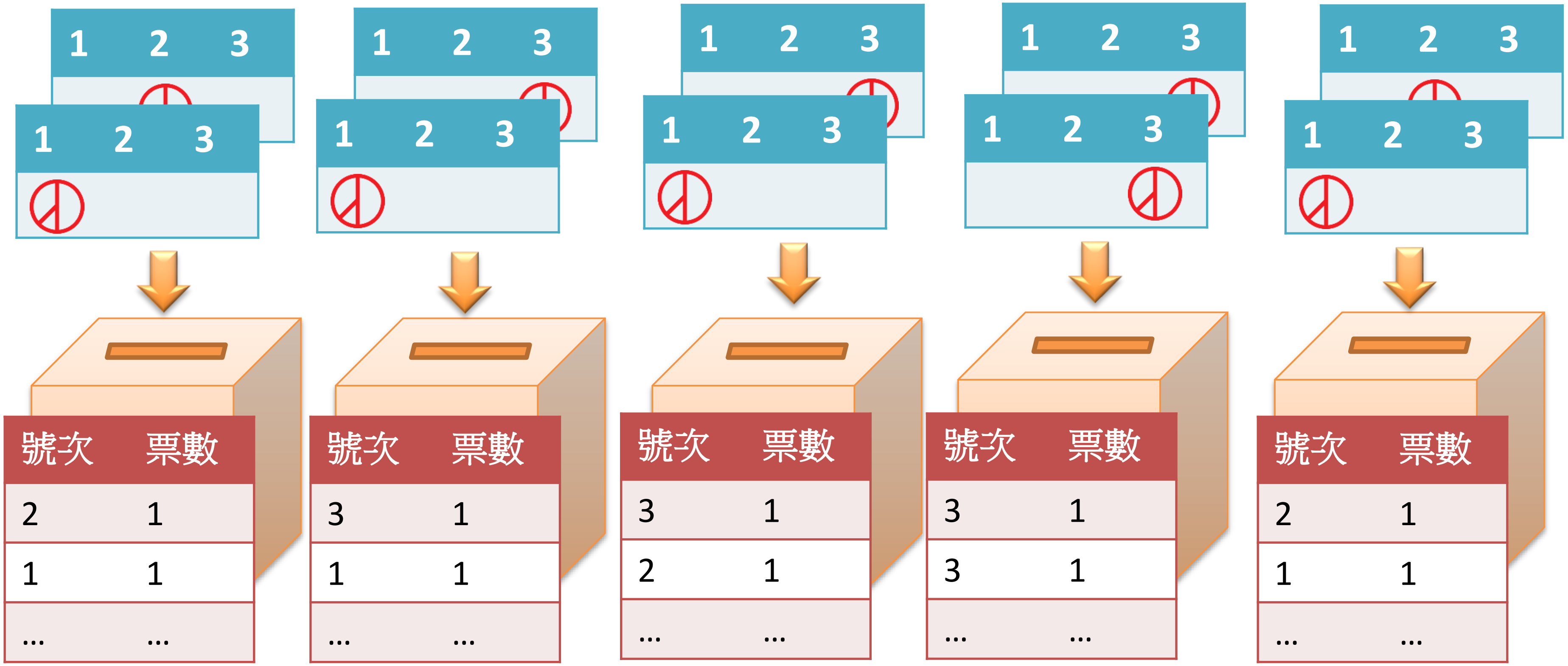


1. 由RM做全局的資源分配
2. NM定時回報目前的資源使用量
3. 每個JOB會有一個負責的AppMaster控制Job
4. 將資源管理與工作控制分開
5. YARN為一通用的資源管理系統
   可達成在YARN上運行多種框架



Mapreduce Example

- 台中市5個選區，共100萬票，要算出每個候選人的得票數



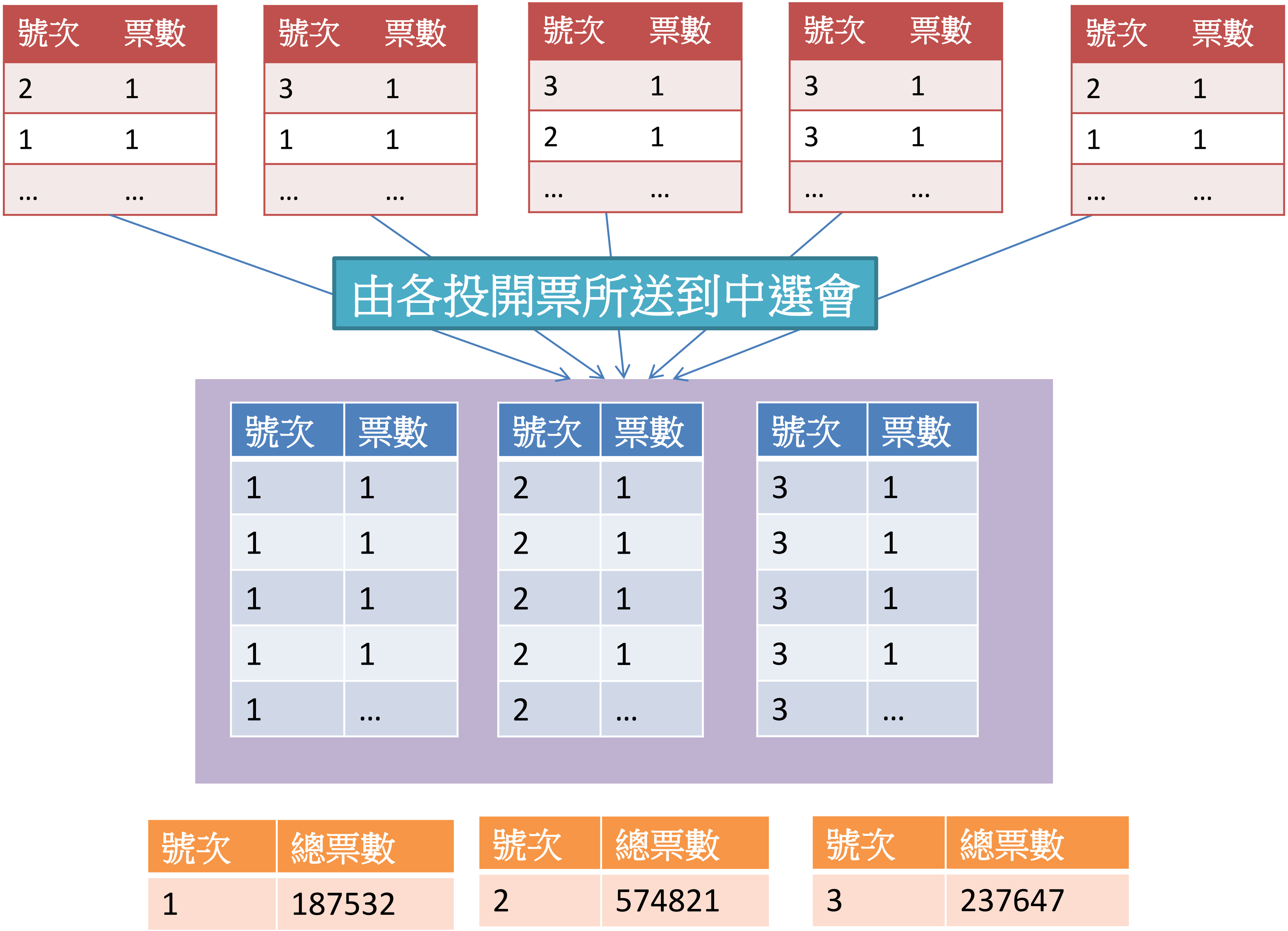
號次	票數
2	1
1	1
…	…

號次	票數
3	1
1	1
…	…

號次	票數
3	1
2	1
…	…

號次	票數
3	1
3	1
…	…

號次	票數
2	1
1	1
…	…



號次	票數
2	1
1	1
…	…

號次	票數
3	1
1	1
…	…

號次	票數
3	1
2	1
…	…

號次	票數
3	1
3	1
…	…

號次	票數
2	1
1	1
…	…

由各投開票所送到中選會

號次	票數
1	1
1	1
1	1
1	1
1	…

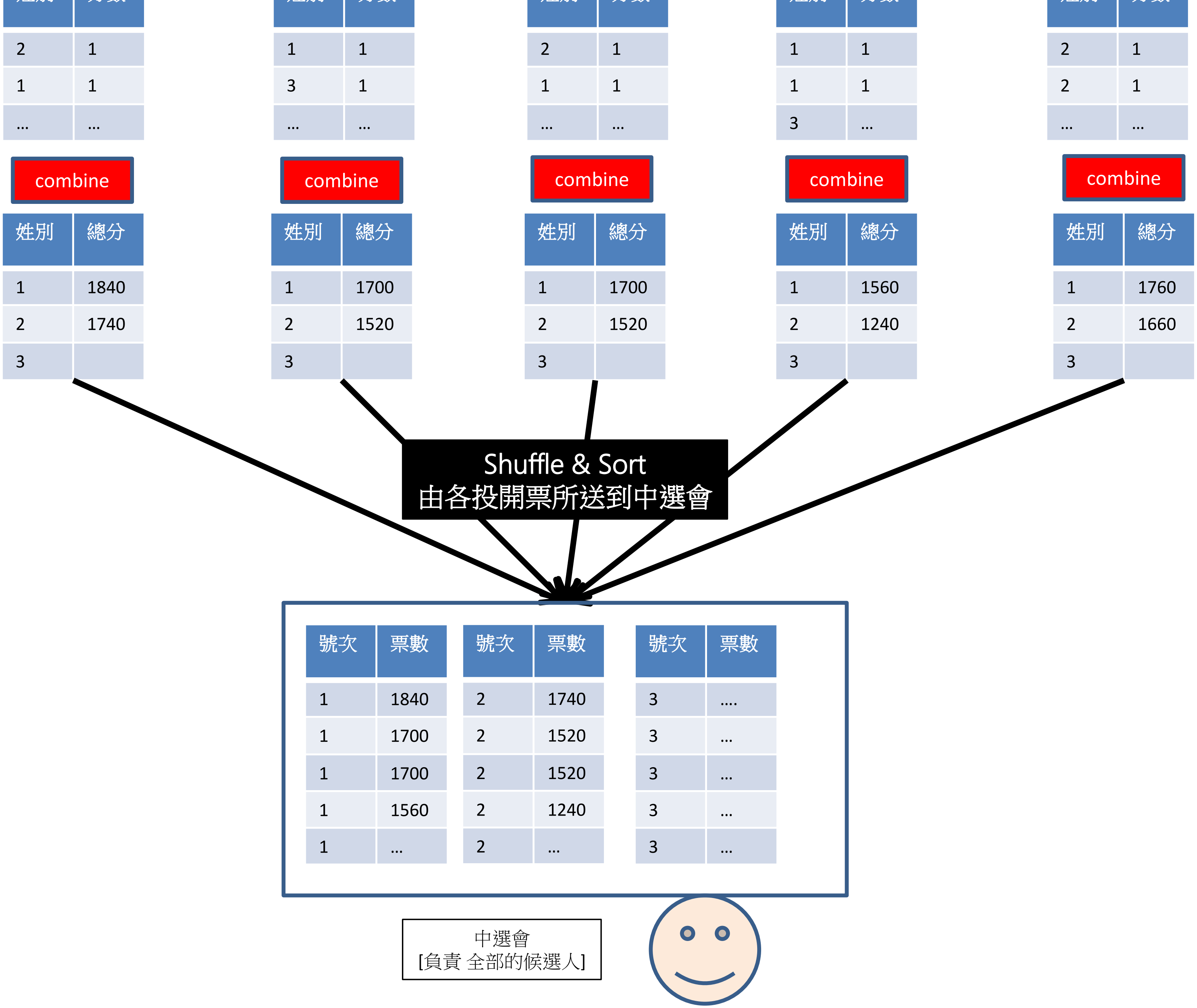
號次	票數
2	1
2	1
2	1
2	1
2	…

號次	票數
3	1
3	1
3	1
3	1
3	…

號次	總票數
1	187532

號次	總票數
2	574821

號次	總票數
3	237647



2	1
1	1
…	…

combine

姓別	總分
1	1840
2	1740
3	

1	1
3	1
…	…

combine

姓別	總分
1	1700
2	1520
3	

2	1
1	1
…	…

combine

姓別	總分
1	1700
2	1520
3	

1	1
1	1
3	…

combine

姓別	總分
1	1560
2	1240
3	

2	1
2	1
…	…

combine

姓別	總分
1	1760
2	1660
3	

Shuffle & Sort
由各投開票所送到中選會

號次	票數
1	1840
1	1700
1	1700
1	1560
1	…

號次	票數
2	1740
2	1520
2	1520
2	1240
2	…

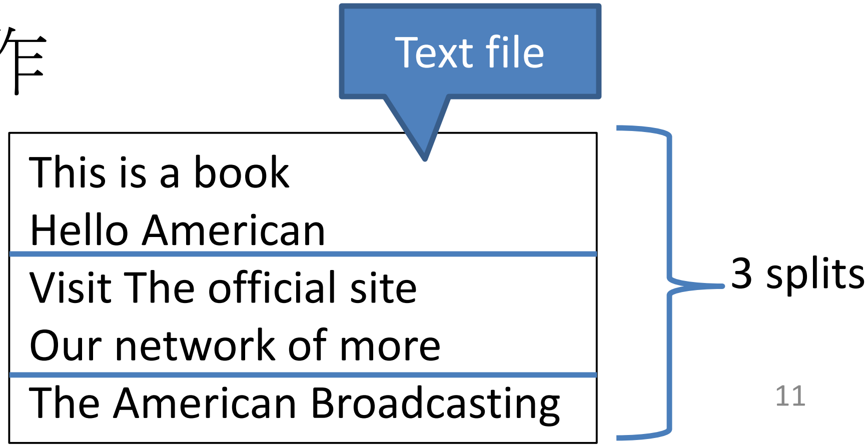
號次	票數
3	….
3	…
3	…
3	…
3	…

中選會
[負責 全部的候選人]

MapReduce Example
- Word Count

Word Count - Mapper

- 將輸入的文字檔案切成split
  - 每個mapper負責一個split
  - 由InputFomrat決定有多少個split
- Mapper處理split中的每一筆record
  - 由RecordReader定義一筆key/value record
- 將每一筆record內的字輸出 (字, 1)
  - 真正map()所執行的工作



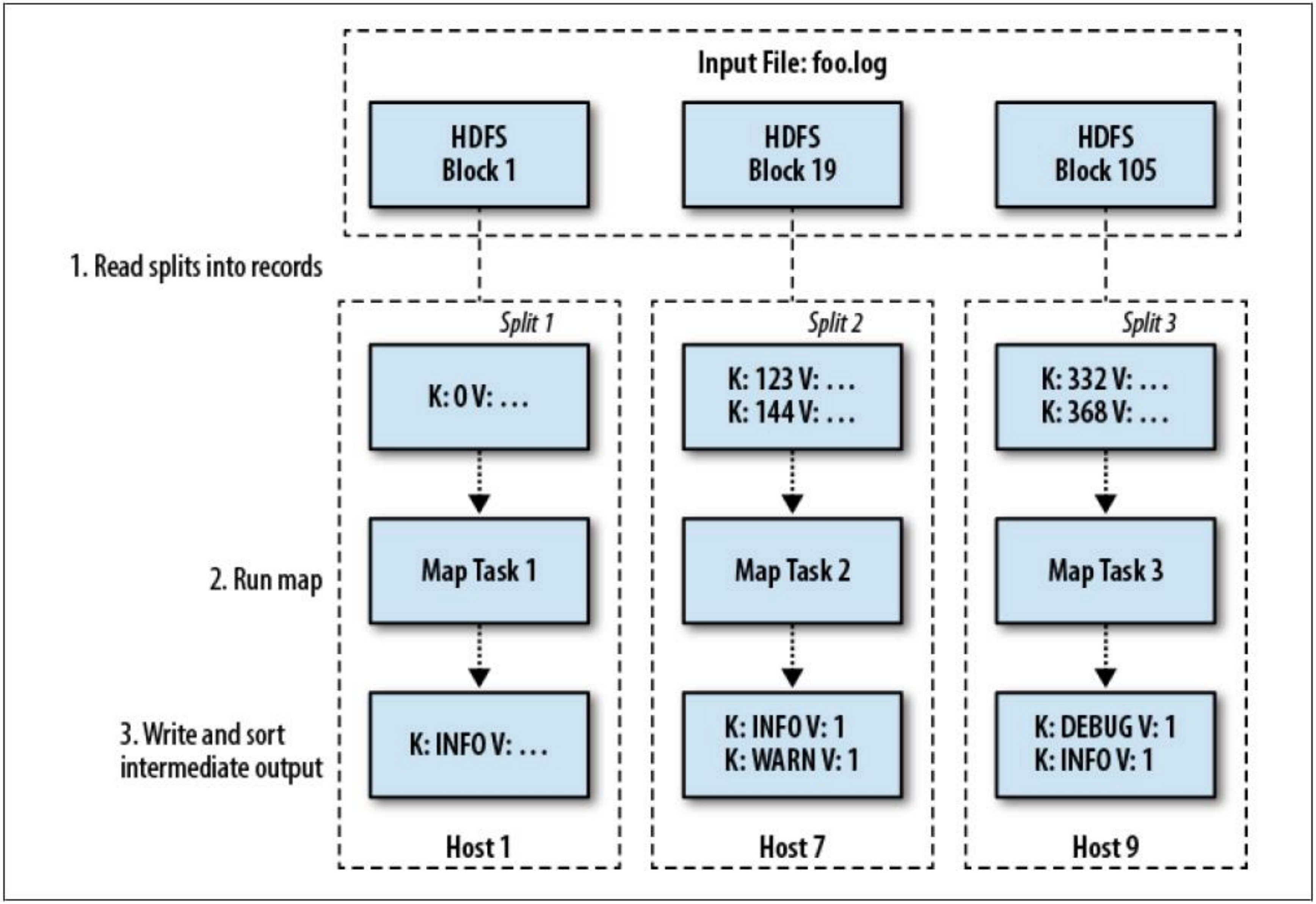
Text file

This is a book
Hello American

Visit The official site
Our network of more

The American Broadcasting

3 splits

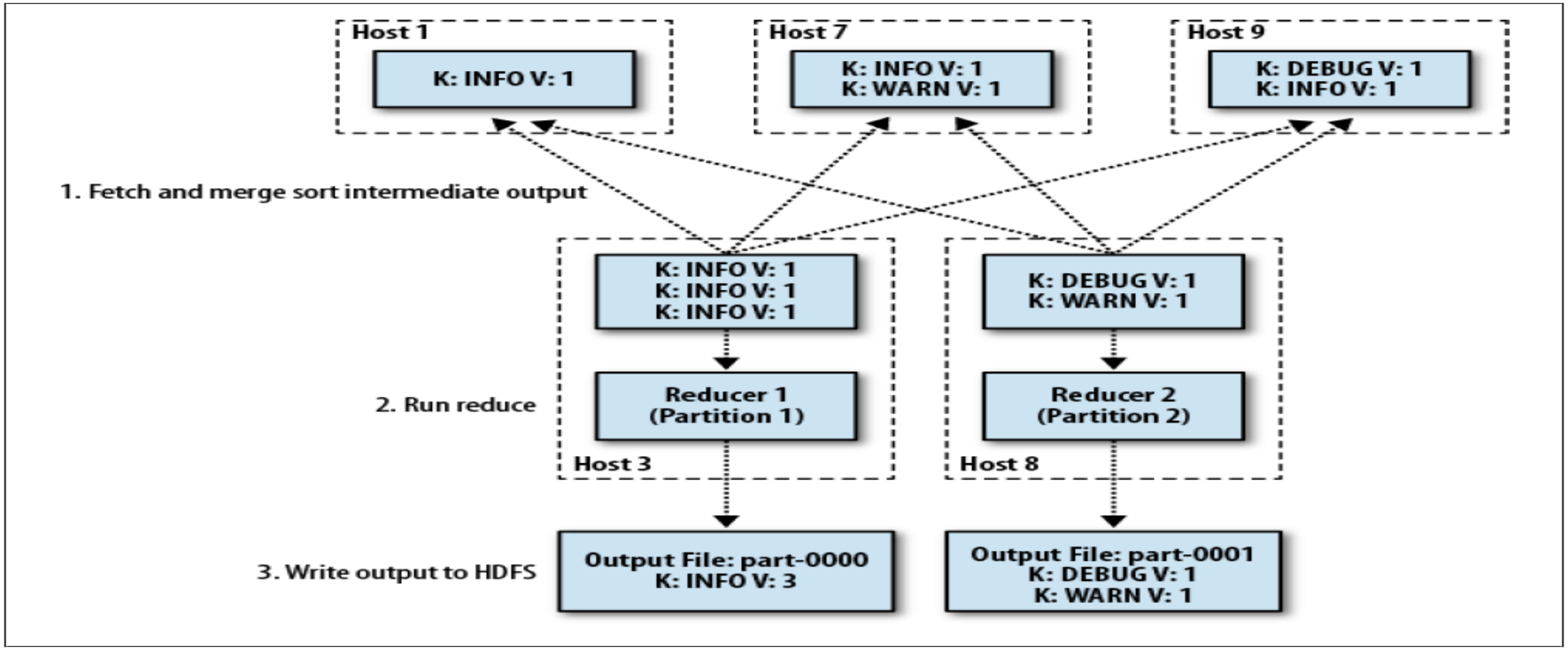

Input File: foo.log
HDFS Block 1
HDFS Block 19
HDFS Block 105
1. Read splits into records
Split 1
Split 2
Split 3
K: 0 V: . . .
K: 123 V: . . .
K: 144 V: . . .
K: 332 V: . . .
K: 368 V: . . .
2. Run map
Map Task 1
Map Task 2
Map Task 3
3. Write and sort intermediate output
K: INFO V: . . .
K: INFO V: 1
K: WARN V: 1
K: DEBUG V: 1
K: INFO V: 1
Host 1
Host 7
Host 9

Word Count – Shuffle & Sort

- Black box
  - 開發人員不用煩腦，framework會自行處理
- 在給Reducer之前完成
- 保證Reducer得到的資訊有下列三個特性
  - 可有多個Reducer
  - 同一個Reducer有可能處理多個Key值
  - 若Reducer看到某個Key1，會看到相對應的所有value
    - 給定Key1，所有Key1的值都會被同一個Reducer處理
    - Reducer 1收到 This這個字，會收到很多 1

Word Count - Reducer

- Reducer收到許多 key與相對應的value list
  - Reducer1 收到 (INFO, [1,1,1])
  - Reducer 2 收到 (DEBUG, [1]), (WARN, [1])
  - Reducer 對每個字的出現次數做加總



用正規的語法描述…

- Mapper：
  - (k1, v1) → list(k2, v2)
  - (0 , "This is a book book") →
    ("This", 1), ("is",1), ("a",1), ("book",1),("book",1)
  - (0, 第一張選票) → (一號,0),(二號,1),(三號,0)
- Reducer：
  - (k2, list(v2)) → (k3,v3)

  - ("This", [1]) → ("This", 1)
  - ("is",[1]) → ("is",1)
  - ("a",[1]) → ("a",1)
  - ("book",[1, 1]) → ("book",2)

  (一號,[1,0,0,1,1,1,0,1,0,1,0]) → (一號, 6)
  (二號, [0,1,1,0,0,0,0,0,1,0,0]) → (二號, 3)
  (三號, [0,0,0,0,0,0,1,0,0,0,1]) → (三號,2)

算字數 - Pseudocode

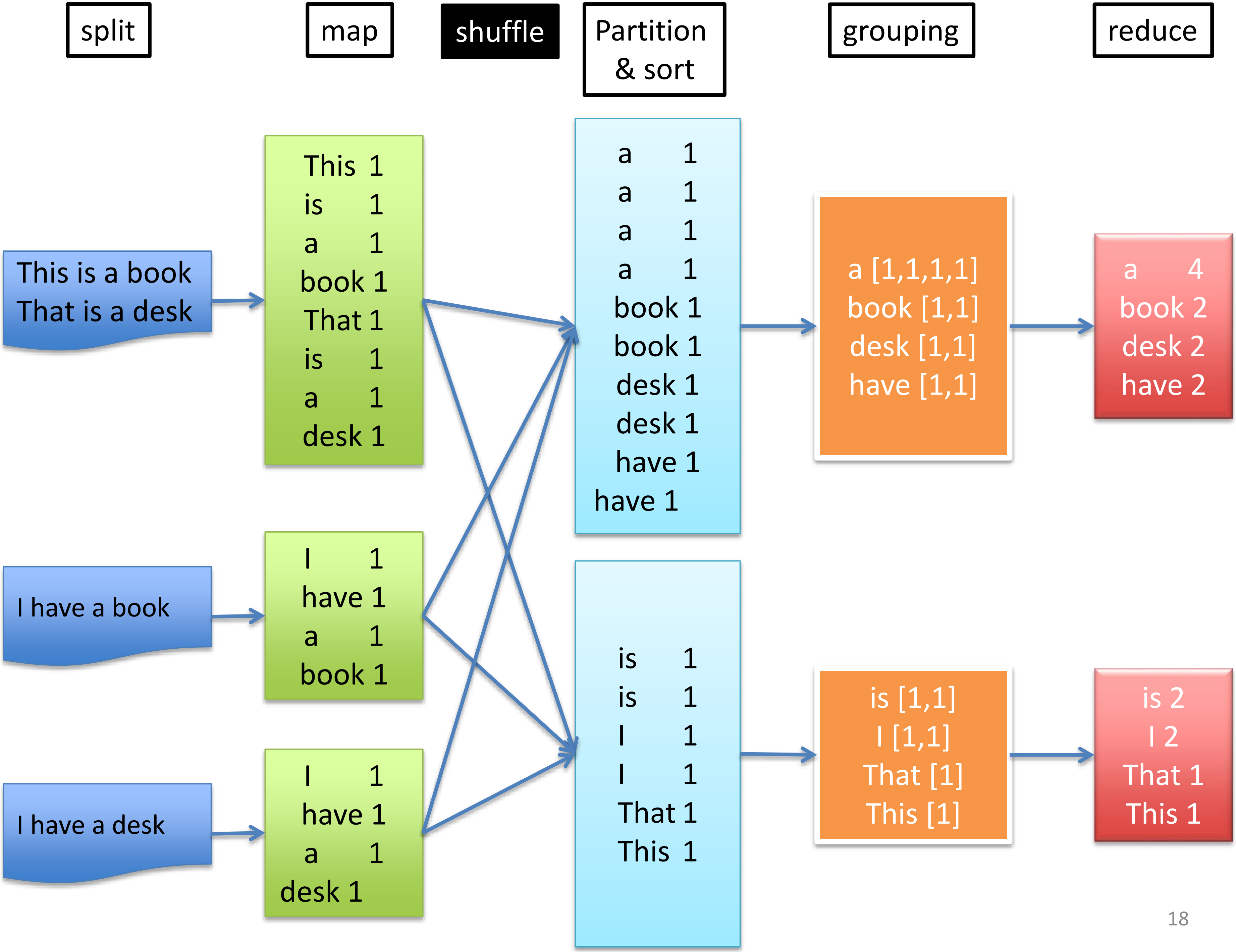
```
void Map (key, value){
        for each word x in value:
            output.collect(x, 1);
}
```

```
void Reduce (keyword, <list of value>){
        for each x in <list of value>:
            sum+=x;
        final_output.collect(keyword, sum);
}
```

算字數 – real code

```
public void map(LongWritable key, Text value, Context context
                ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString()); // line to string token
        while (itr.hasMoreTokens()) {
          word.set(itr.nextToken());    // set word as each input keyword
          context.write(word, one);     // create a pair <keyword, 1>
        }
 }
```

```
public void reduce(Text key, Iterable<IntWritable> values,  Context context
                        ) throws IOException, InterruptedException {
            int sum = 0; // initialize the sum for each keyword
            for (IntWritable val : values) {
              sum += val.get();
            }
            result.set(sum);
            context.write(key, result); // create a pair <keyword, number of occurences>
}
```



split

This is a book
That is a desk

I have a book

I have a desk

map

This 1
is 1
a 1
book 1
That 1
is 1
a 1
desk 1

I 1
have 1
a 1
book 1

I 1
have 1
a 1
desk 1

shuffle

Partition & sort

a 1
a 1
a 1
a 1
book 1
book 1
desk 1
desk 1
have 1
have 1

is 1
is 1
I 1
I 1
That 1
This 1

grouping

a [1,1,1,1]
book [1,1]
desk [1,1]
have [1,1]

is [1,1]
I [1,1]
That [1]
This [1]

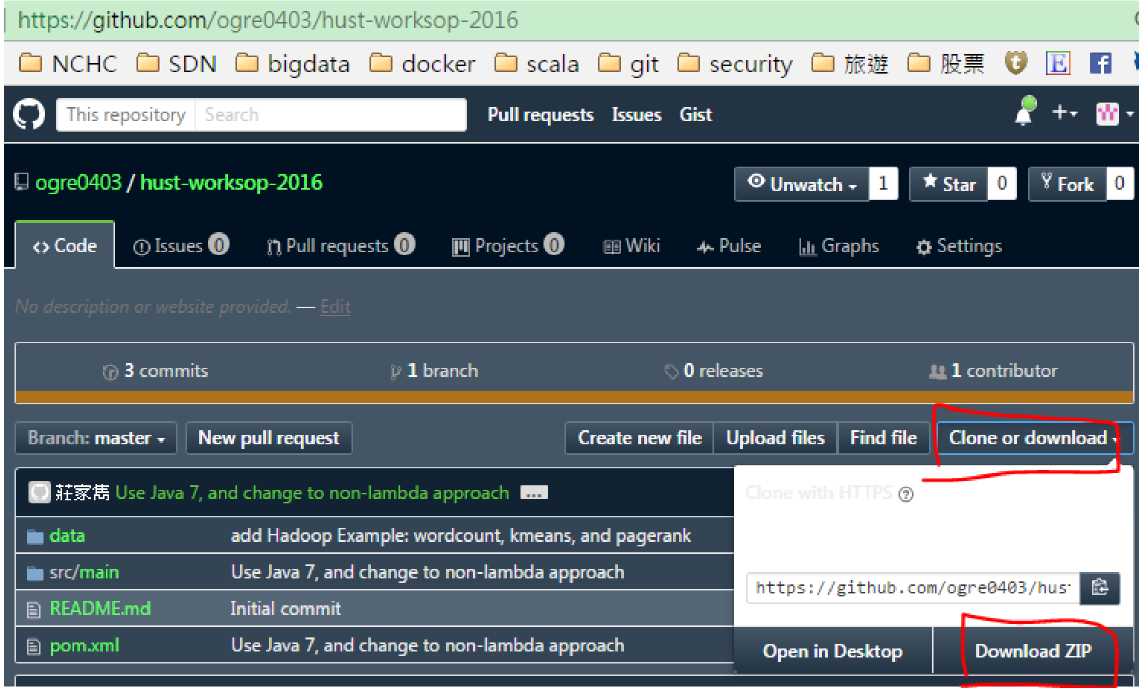
reduce

a 4
book 2
desk 2
have 2

is 2
I 2
That 1
This 1

Labs 0: IntelliJ IDEA Setup

- Install IntelliJ IDEA Community Version
- Download labs code
  - https://github.com/ogre0403/ntcu-workshop-2016
- Import labs project into Intellij IDEA



https://github.com/ogre0403/hust-worksop-2016

ogre0403 / hust-worksop-2016

No description or website provided. — Edit

3 commits · 1 branch · 0 releases · 1 contributor

莊家雋 Use Java 7, and change to non-lambda approach …

data	add Hadoop Example: wordcount, kmeans, and pagerank
src/main	Use Java 7, and change to non-lambda approach
README.md	Initial commit
pom.xml	Use Java 7, and change to non-lambda approach

https://github.com/ogre0403/hus

Java Programing

- Code skeleton
  - POM.xml snippet
  - Driver code snippet
  - Map class snippet
  - Reduce class snippet

POM.xml

```
<properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <hadoop.version>2.6.0</hadoop.version>
        <java.version>1.7</java.version>
</properties>
```

```
<dependencies>
      <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>${hadoop.version}</version>
      </dependency>
      <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-common</artifactId>
        <version>${hadoop.version}</version>
      </dependency>
</dependencies>
```

```
public  class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
      ...
  }
```

Map code

```
  public  class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
      ...
   }
```

Reduce code

```
public class MyMR {
  public static void main(String[] args) throws Exception {
      ...
   }
}
```

Driver setup

Driver setup

```
Configuration conf = new Configuration();
Job job = new Job(conf, “New MR job");
job.setJarByClass(MyMR.class);
job.setMapperClass(MyMapper.class);
job.setReducerClass(MyReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
```

Configuration & Run

```
Configuration conf = new Configuration();
Job job = new Job(conf, “New MR job");
…
System.exit(job.waitForCompletion(true) ? 0 : 1);
```

Set Map/Reduce/Combine Class

```
job.setJarByClass(MyMR.class);
job.setMapperClass(MyMapper.class);
job.setReducerClass(MyReducer.class);
```

Set input/output format

```
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
```

- Inputformat
  - Hadoop 如何讀取來源資料
  - plain text, DB, or customer source…
  - 預設為TextInputFormat class
    - 每一行為一筆record,
    - key 為在文件中的offset
    - value為整行內容

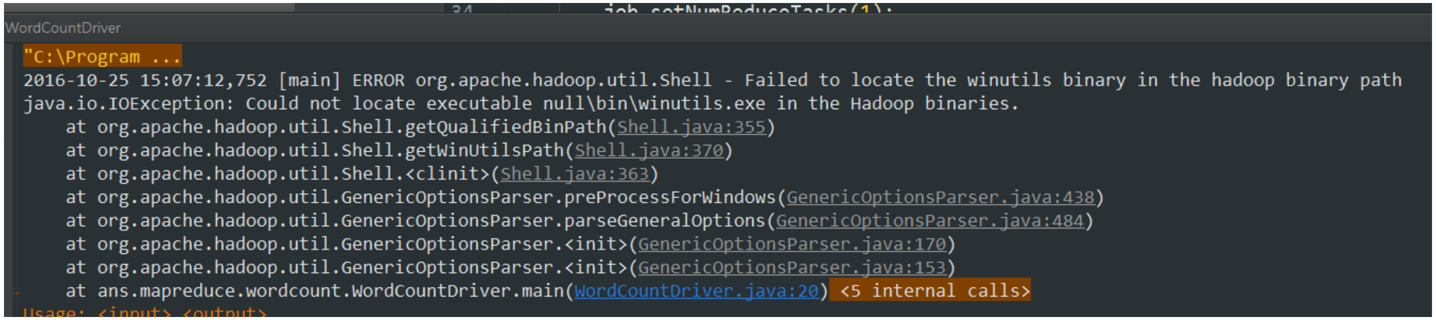
- Outputformat
  - Hadoop如何將分析完的結果輸出
  - 預設為TextOutputFormat class
  - 每一筆結果為輸出文件中的一行
  - 每一行包含key/value，預設以tab分隔
  - Key/value可為任意class, 但需在Driver中設定

- 若使用預設的TextInputFormat/TextOutputFormat, 無需在Driver中設定

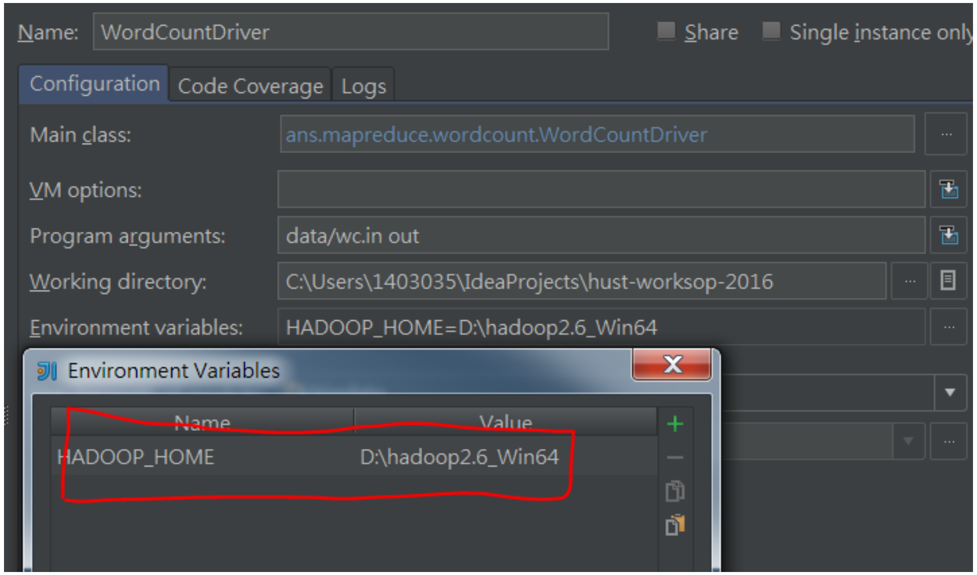
- 若使用非預設的input/output format
  - job.setInputFormatClass(SequenceFileInputFormat.class);
  - job.setOutputFormatClass(NullOutputFormat.class);

Labs 1: 在IDE裡執行

- Windows 64
  - 設定HADOOP_HOME
  - 將hadoop.dll 放至C:\windows\system32



```
34        job.setNumReduceTasks(1);
"C:\Program ...
2016-10-25 15:07:12,752 [main] ERROR org.apache.hadoop.util.Shell - Failed to locate the winutils binary in the hadoop binary path
java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.
    at org.apache.hadoop.util.Shell.getQualifiedBinPath(Shell.java:355)
    at org.apache.hadoop.util.Shell.getWinUtilsPath(Shell.java:370)
    at org.apache.hadoop.util.Shell.<clinit>(Shell.java:363)
    at org.apache.hadoop.util.GenericOptionsParser.preProcessForWindows(GenericOptionsParser.java:438)
    at org.apache.hadoop.util.GenericOptionsParser.parseGeneralOptions(GenericOptionsParser.java:484)
    at org.apache.hadoop.util.GenericOptionsParser.<init>(GenericOptionsParser.java:170)
    at org.apache.hadoop.util.GenericOptionsParser.<init>(GenericOptionsParser.java:153)
    at ans.mapreduce.wordcount.WordCountDriver.main(WordCountDriver.java:20) <5 internal calls>
Usage: <input> <output>
```



Name: WordCountDriver □ Share □ Single instance only

Main class: ans.mapreduce.wordcount.WordCountDriver

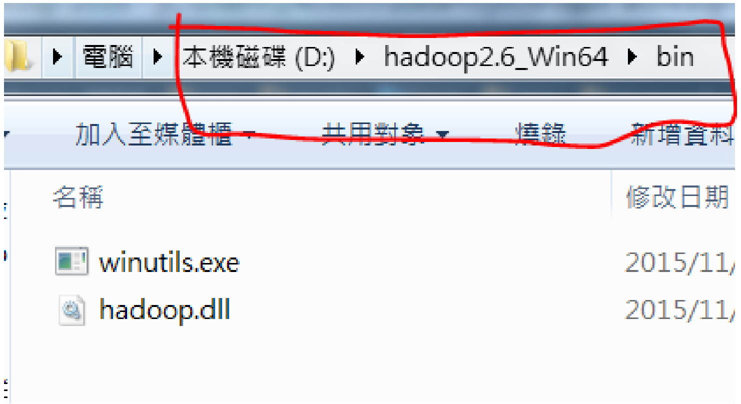
VM options:

Program arguments: data/wc.in out

Working directory: C:\Users\1403035\IdeaProjects\hust-worksop-2016

Environment variables: HADOOP_HOME=D:\hadoop2.6_Win64

Name	Value
HADOOP_HOME	D:\hadoop2.6_Win64



名稱	修改日期
winutils.exe	2015/11/
hadoop.dll	2015/11/

Labs 1: 在虛擬機器執行

- 設定虛擬機器
- 上傳至 hdfs
- 執行Hadoop 指令

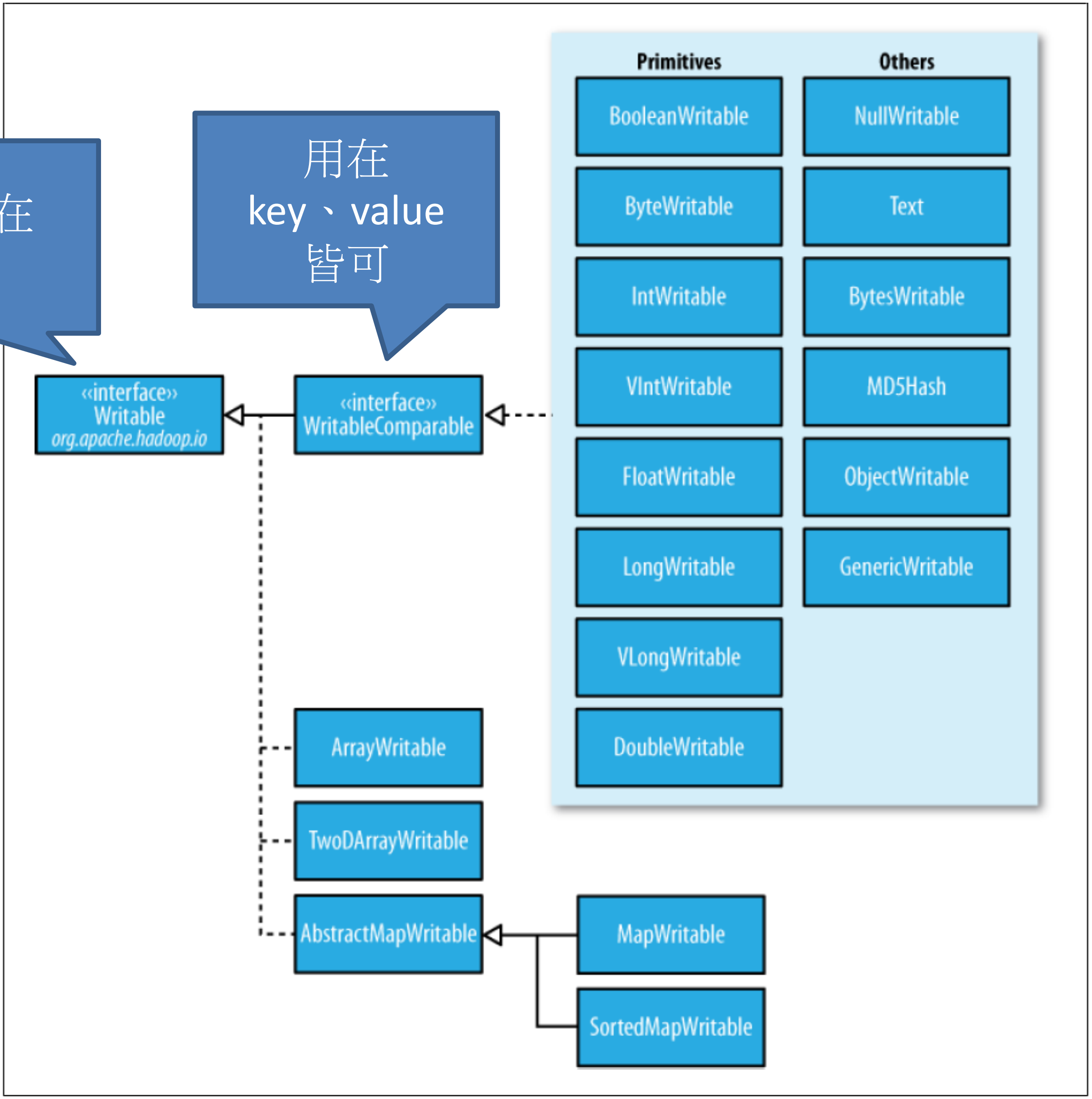
MapReduce 進階概念

What is Writable Class

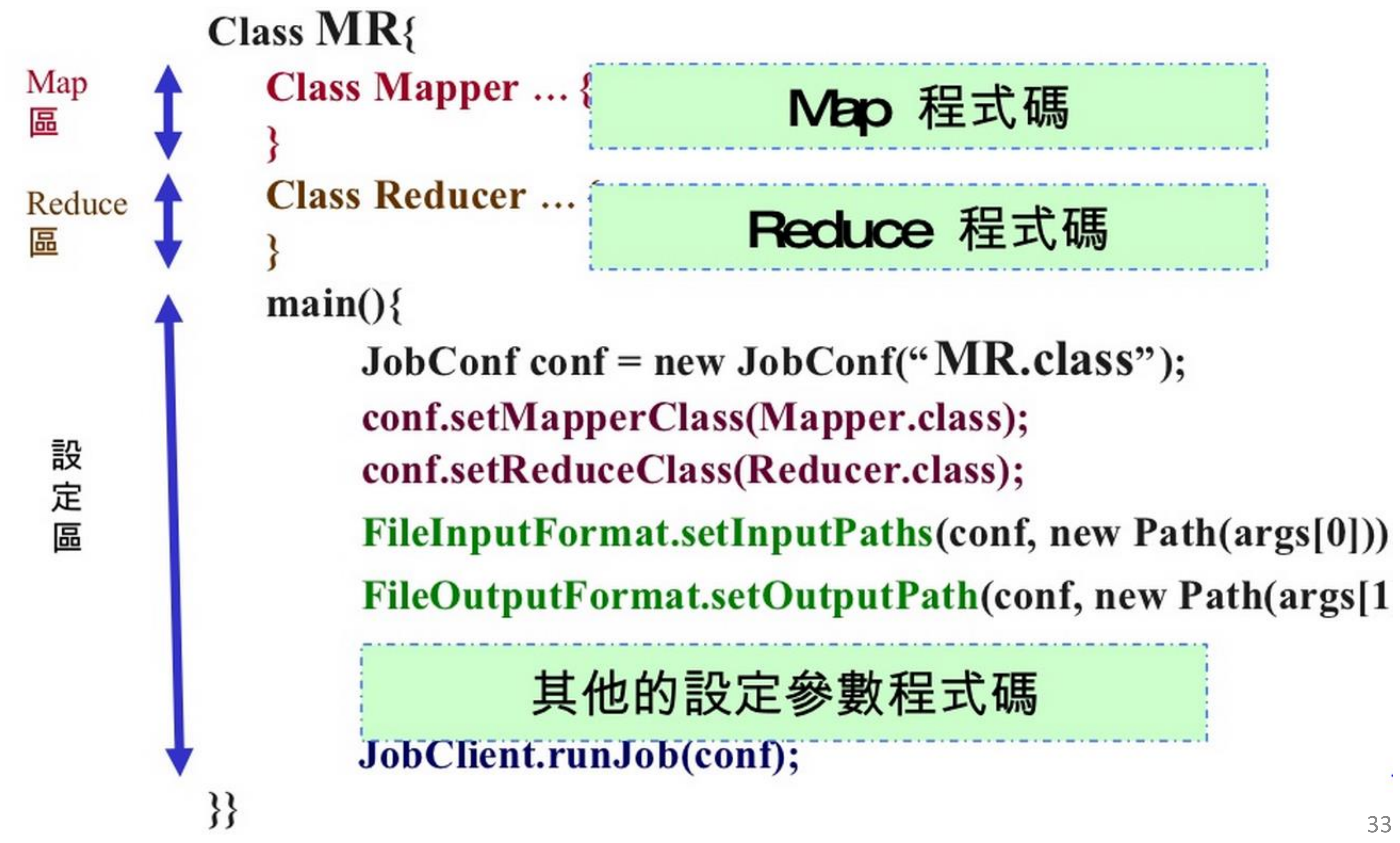
- 什麼是Text類型、什麼是IntWritable類型
  - Text: Wrapper for Java String class
  - IntWritable: Wrapper for Java int
- 序列化框架
  - 物件在網路上傳遞要透過serialize/deserializae
  - Java 本身有Serializable
  - Hadoop自行設計Writable 序列化框架
- 若內建的writable不合需求，需自行定義
  - Implement writable : 用在value
  - Implement writablecomparable: 用在key、value

只可用在
value

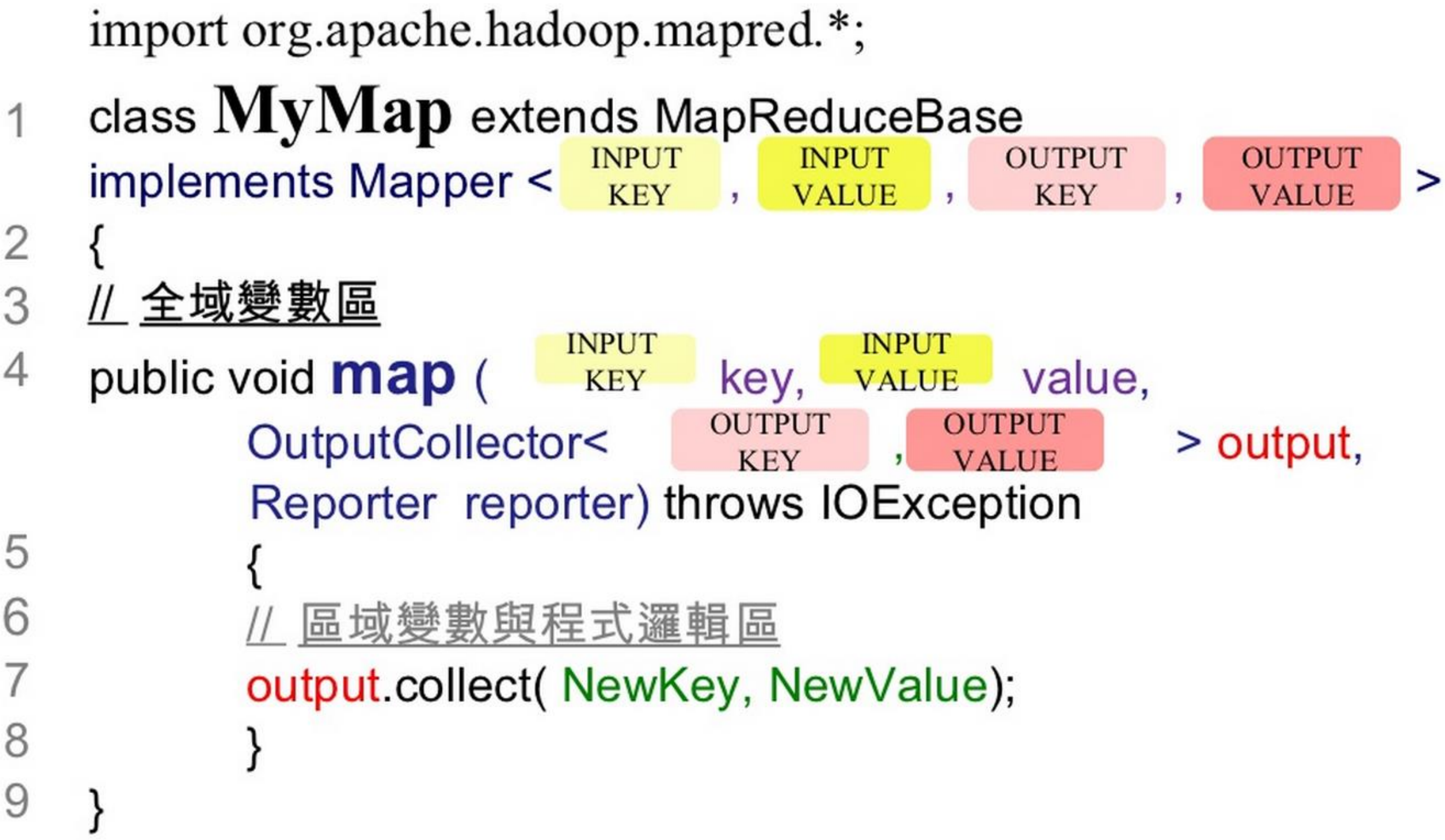
用在
key、value
皆可



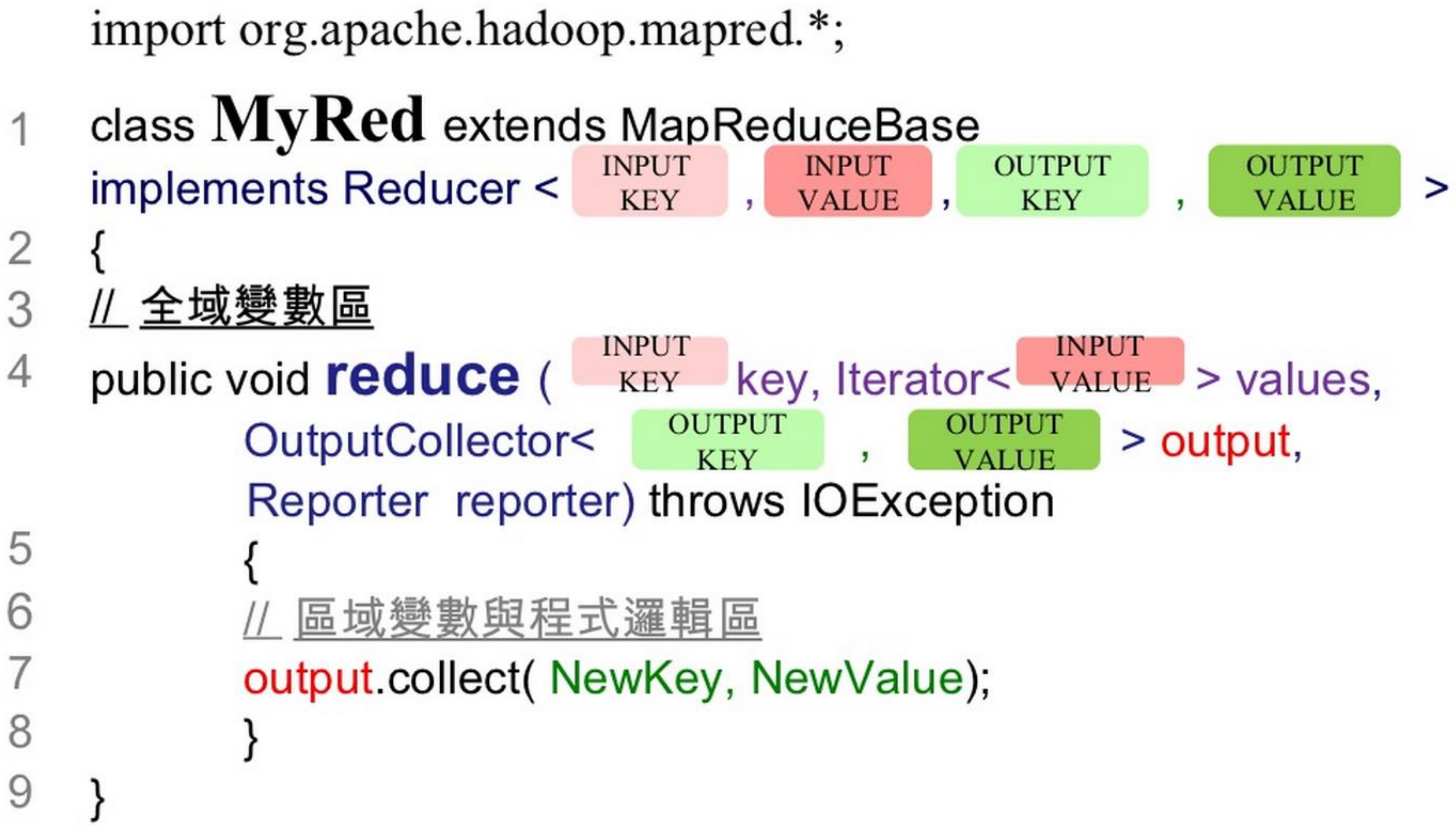
New and Old API



```
Class MR{
    Class Mapper … {
    }
    Class Reducer …
    }
    main(){
        JobConf conf = new JobConf("MR.class");
        conf.setMapperClass(Mapper.class);
        conf.setReduceClass(Reducer.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]))
        FileOutputFormat.setOutputPath(conf, new Path(args[1

        JobClient.runJob(conf);
}}
```



```
import org.apache.hadoop.mapred.*;
class MyMap extends MapReduceBase
implements Mapper < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
{
// 全域變數區
public void map ( INPUT KEY key, INPUT VALUE value,
          OutputCollector< OUTPUT KEY , OUTPUT VALUE > output,
          Reporter  reporter) throws IOException
          {
          // 區域變數與程式邏輯區
          output.collect( NewKey, NewValue);
          }
}
```

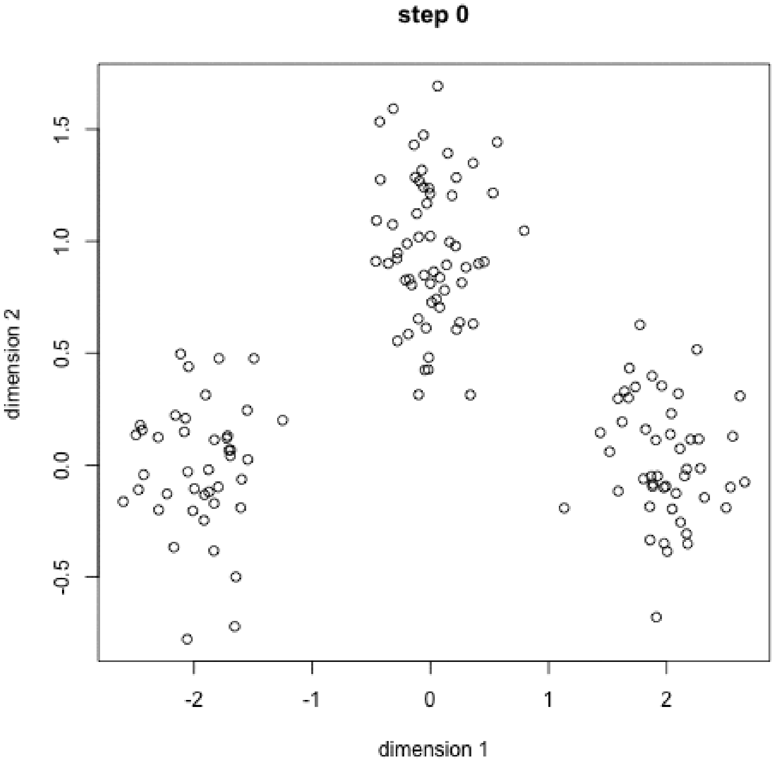


```
import org.apache.hadoop.mapred.*;
class MyRed extends MapReduceBase
implements Reducer < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
{
// 全域變數區
public void reduce ( INPUT KEY key, Iterator< INPUT VALUE > values,
          OutputCollector< OUTPUT KEY , OUTPUT VALUE > output,
          Reporter  reporter) throws IOException
          {
          // 區域變數與程式邏輯區
          output.collect( NewKey, NewValue);
          }
}
```

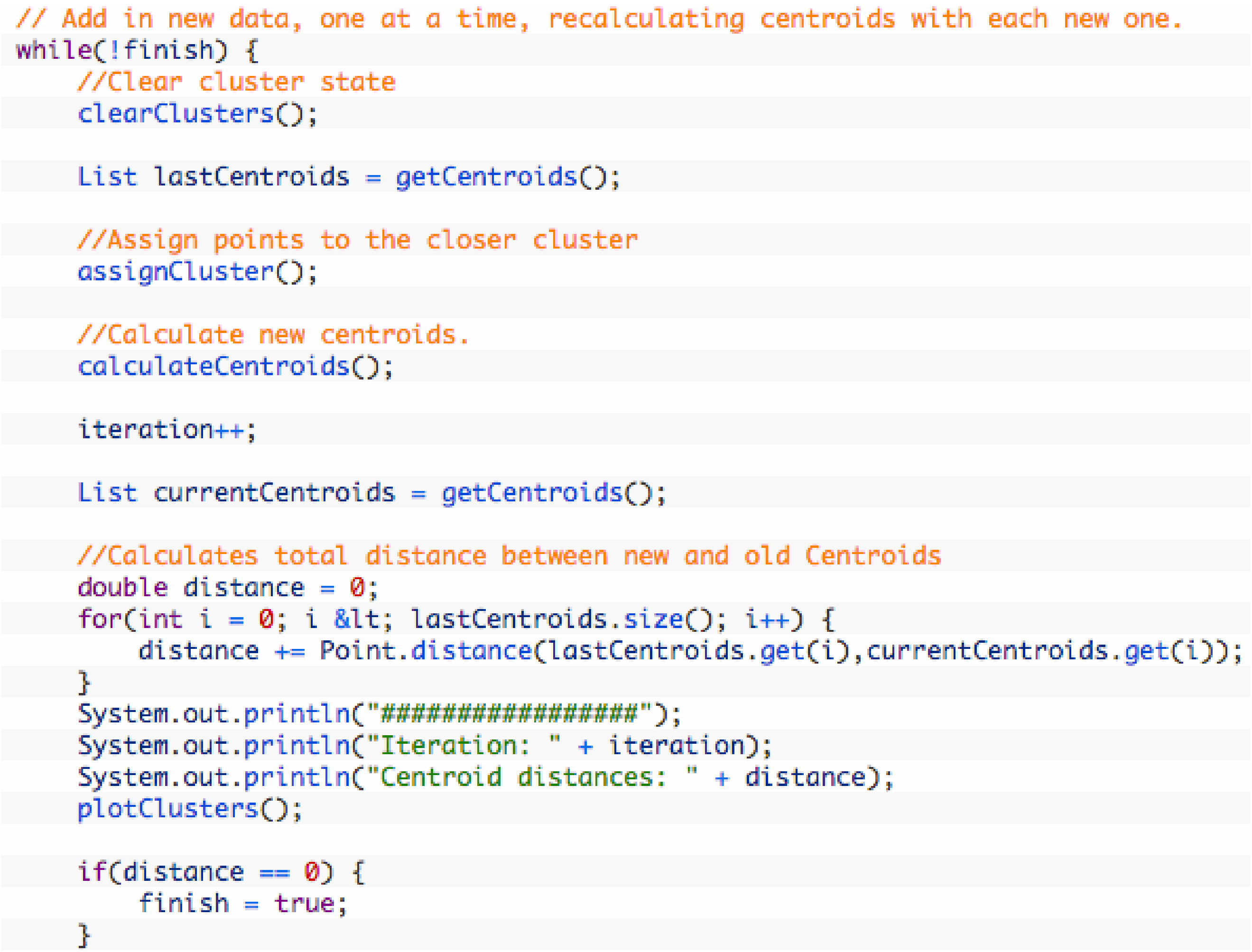
MapReduce Example
- K-means

K-means clustering

- 隨機選取資料組中的k筆資料當作初始群中心u_1~u_k
- 計算每個資料xi 對應到最短距離的群中心 (固定 ui 求解所屬群 Si)
- 利用目前得到的分類重新計算群中心 (固定 Si 求解群中心 ui)
- 重複step 2,3直到收斂 (達到最大疊代次數 or 群心中移動距離很小)



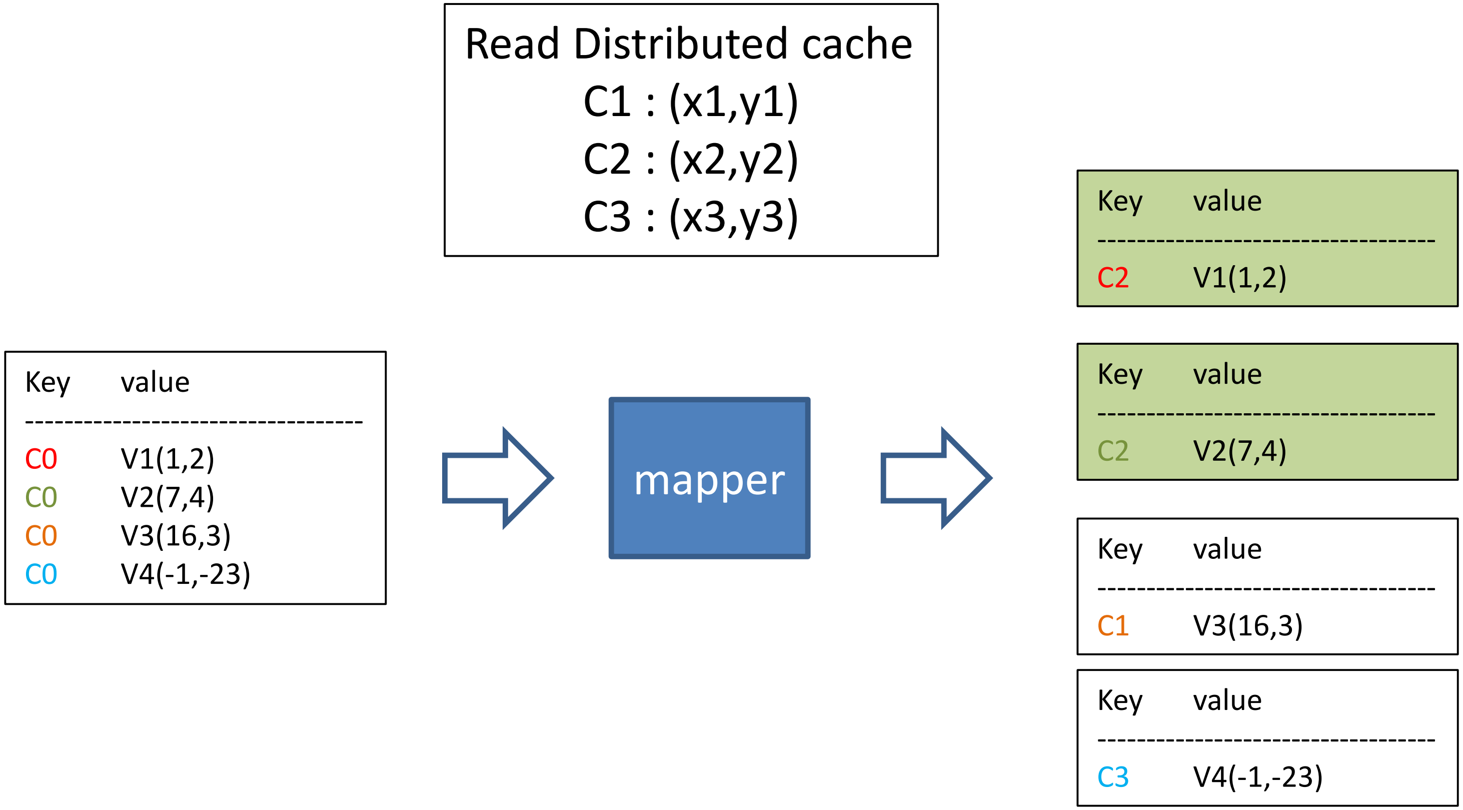
集中式版本程式



```
// Add in new data, one at a time, recalculating centroids with each new one.
while(!finish) {
    //Clear cluster state
    clearClusters();

    List lastCentroids = getCentroids();

    //Assign points to the closer cluster
    assignCluster();

    //Calculate new centroids.
    calculateCentroids();

    iteration++;

    List currentCentroids = getCentroids();

    //Calculates total distance between new and old Centroids
    double distance = 0;
    for(int i = 0; i &lt; lastCentroids.size(); i++) {
        distance += Point.distance(lastCentroids.get(i),currentCentroids.get(i));
    }
    System.out.println("#################");
    System.out.println("Iteration: " + iteration);
    System.out.println("Centroid distances: " + distance);
    plotClusters();

    if(distance == 0) {
        finish = true;
    }
```

Map

輸入為<目前的中心，point>
求point到每個中心的距離
輸出為<所屬的中心，point>



Read Distributed cache
C1 : (x1,y1)
C2 : (x2,y2)
C3 : (x3,y3)

Key	value
C0	V1(1,2)
C0	V2(7,4)
C0	V3(16,3)
C0	V4(-1,-23)

Key	value
C2	V1(1,2)

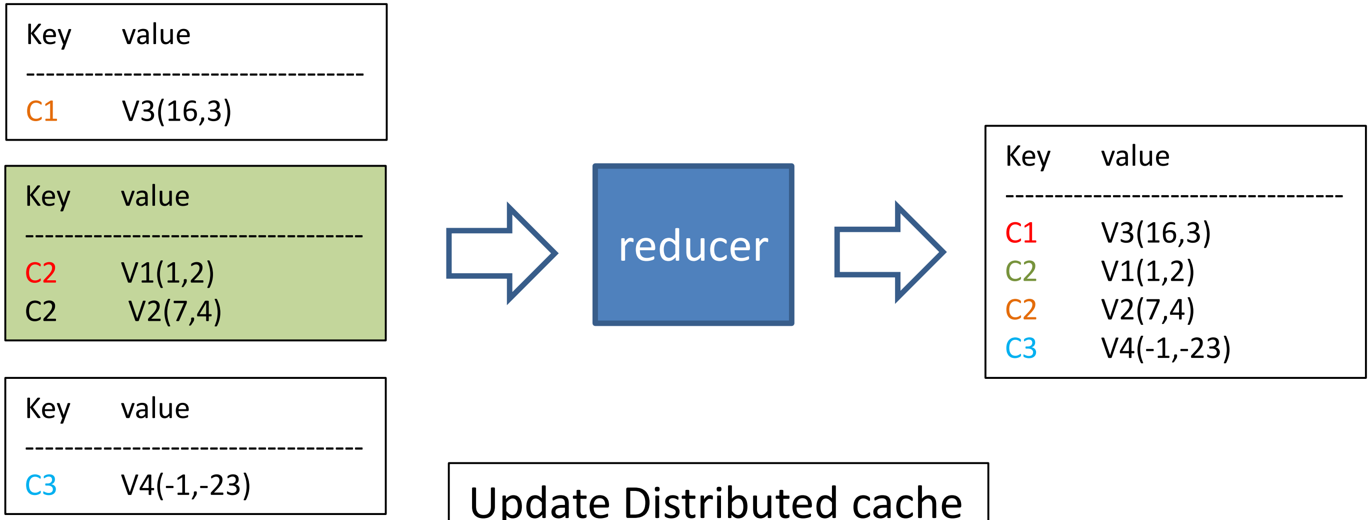
Key	value
C2	V2(7,4)

Key	value
C1	V3(16,3)

Key	value
C3	V4(-1,-23)

Reducer

輸入為<中心，屬於該中心的所有point>
對所有的point計算出新的中心
輸出<新的中心，point>做為下一次疊代



Key	value
C1	V3(16,3)

Key	value
C2	V1(1,2)
C2	V2(7,4)

Key	value
C3	V4(-1,-23)

Key	value
C1	V3(16,3)
C2	V1(1,2)
C2	V2(7,4)
C3	V4(-1,-23)

Update Distributed cache

C1 : (x’1,y’1)
C2 : (x’2,y’2)
C3 : (x’3,y’3)

Labs 2: K-means (MapReduce)

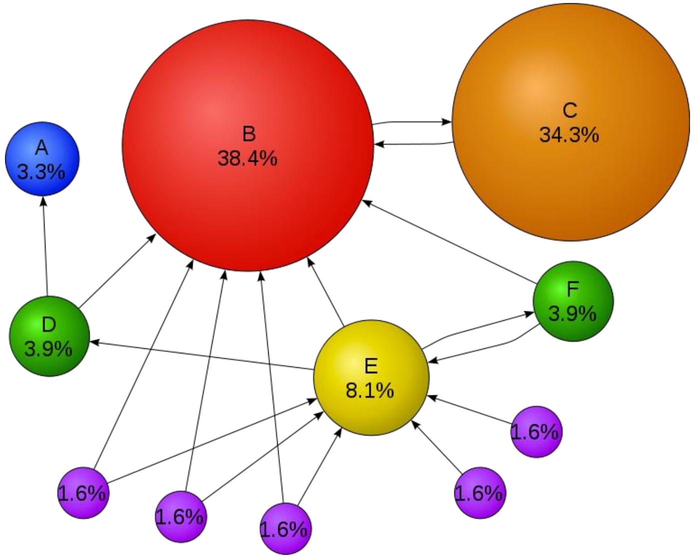
- KMeansMapper
  - Use DistanceMeasurer.measureDistance() to calculate distance between vector and ClusterCenter
  - Find the nearest ClusterCenter and the shortest distance
- KMeansReducer
  - Sum up all Vector value. (Each digital is stored in source[])

    Save result in resultVector[].
  - Calculate mean of each digital in resultVector[]

MapReduce Example
- Page Rank

PageRank

- 評估網頁重要程度的指標



$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}$$

$$= \sum_{p_j} \frac{\mathrm{PageRank}(p_j)}{L(p_j)}$$

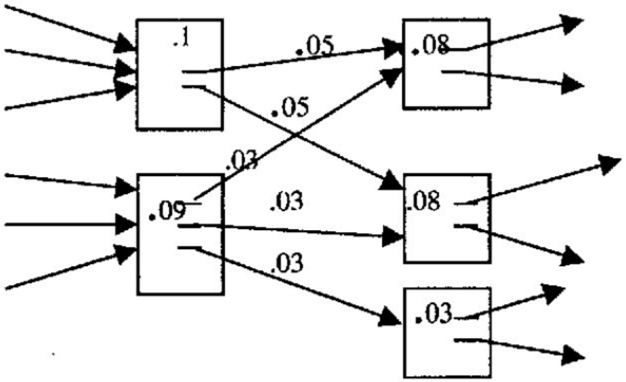
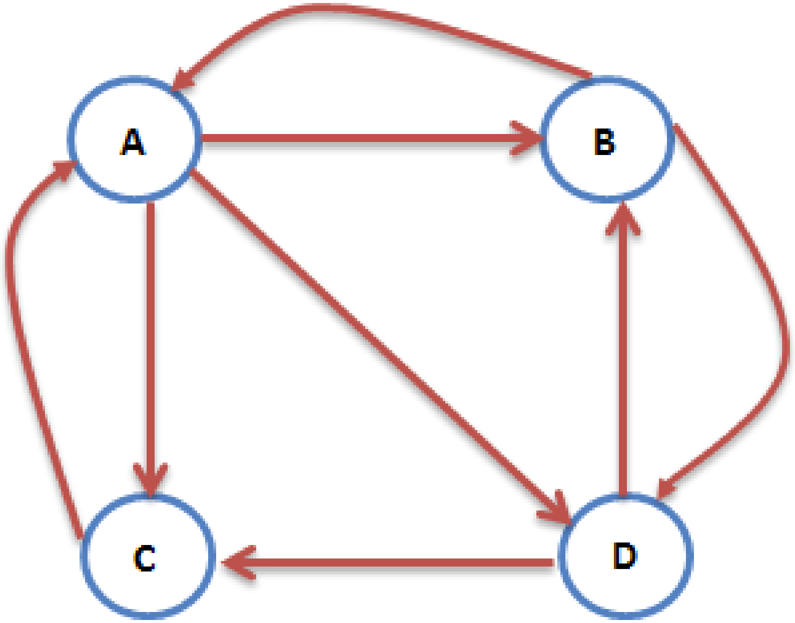


图 1　链接结构中的部分网页及其 PageRank 值



Page link matrix
 = adjacency matrix
M[i][j] = 1 表示由 i 到 j 有一個邊

	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	0
D	0	1	1	0

Page link probability matrix
M[i][j] = p 表示由 i 到 j 的機率為 p

	A	B	C	D
A	0	1/3	1/3	1/3
B	1/2	0	0	1/2
C	1	0	0	
D	0	1/2	1/2	0

Transport Page link probability matrix
M[i][j] = p 表示由 j 到 i 的機率為 p
= 前一頁中的1 / Lj

	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

PR(A) = P(B→A) * PR(B)
+ P(C→A) * PR(C)
+ P(D→A) * PR(D)

PR(B) = P(A→B) * PR(A)
+ P(C→B) * PR(C)
+ P(D→B) * PR(D)

PR(C) = P(A→C) * PR(A)
+ P(B→C) * PR(B)
+ P(D→C) * PR(D)

PR(D) = P(A→D) * PR(A)
+ P(B→D) * PR(B)
+ P(C→D) * PR(C)

$$\sum_{p_j} \frac{\mathrm{PageRank}(p_j)}{L(p_j)}$$

Iteration 1

P	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

X

	PR
A	1/4
B	1/4
C	1/4
D	1/4

=

	PR
A	9/24
B	5/24
C	5/24
D	5/24

Iteration 2

P	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

X

	PR
A	9/24
B	5/24
C	5/24
D	5/24

=

	PR
A	15/48
B	11/48
C	11/48
D	11/48

Iteration N

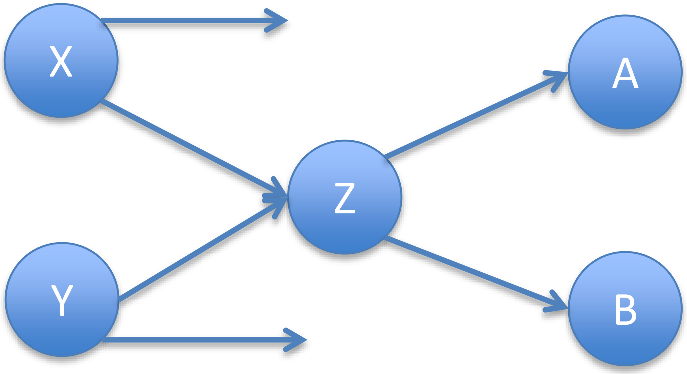
P	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

…

	PR
A	3/9
B	2/9
C	2/9
D	2/9

Reduce Pseudo Code

Z	<PR$_x$, L$_x$>
	<PR$_y$, L$_y$>
	<A,B>



$$\sum_{p_j} \frac{\text{PageRank}(p_j)}{L(p_j)}$$

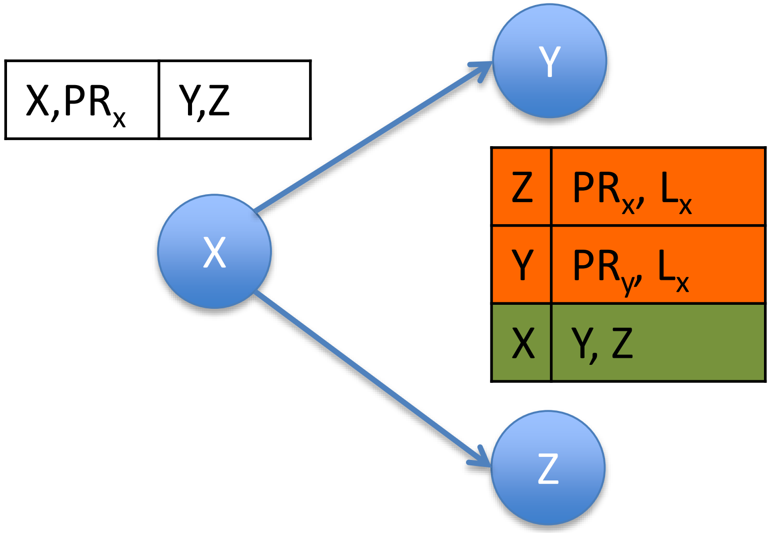
Z, new_PR$_z$	A,B

- Input:
  - Key: <p0>
  - Value: [<p1, p2, …, pn>, <PR, L >, <PR, L> …]
- Output
  - Key: <p0 new_PR>
  - Value: <p1, p2, …, pn>

Map Pseudo Code

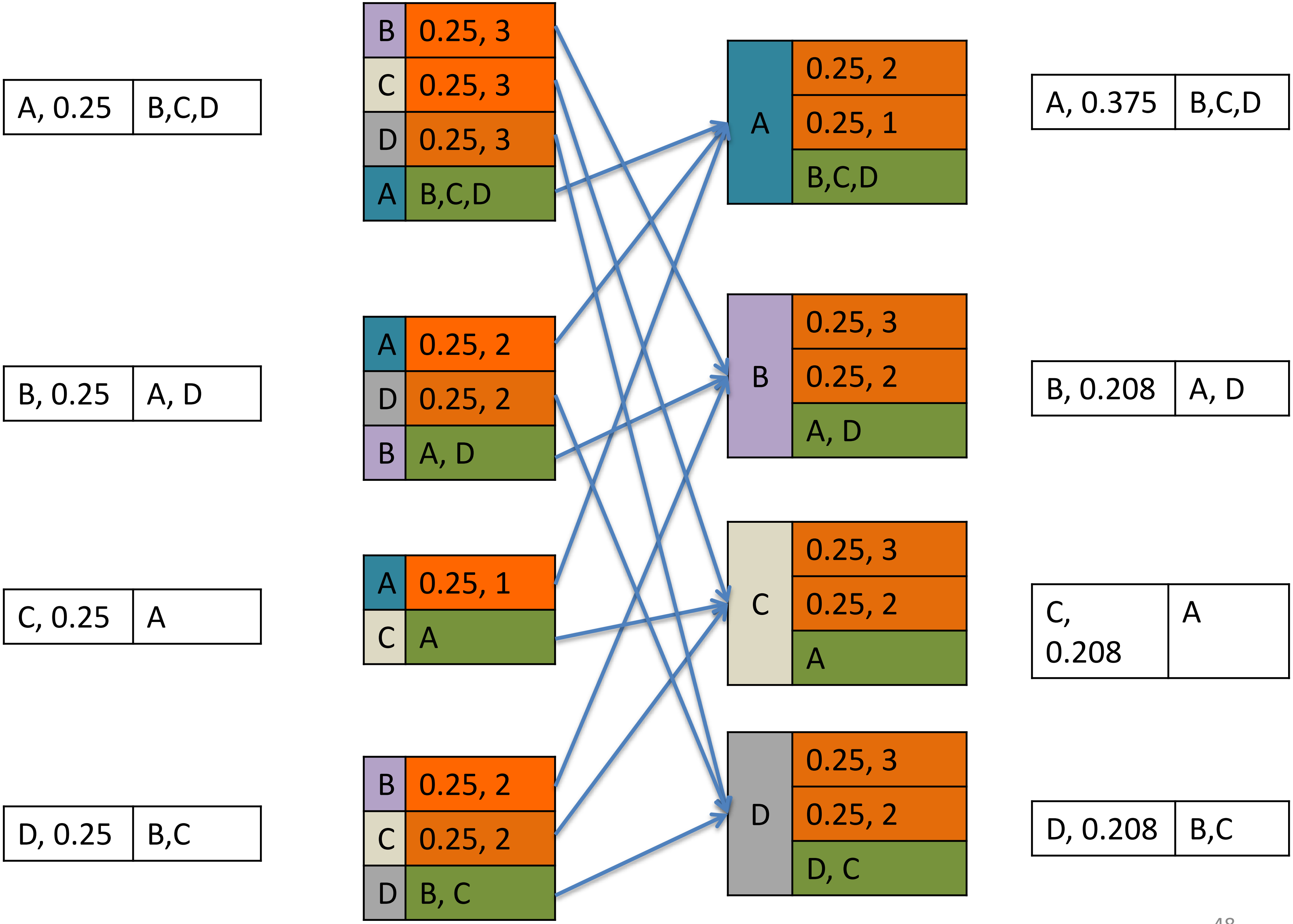
- Input:
  - Key: <p0, PR>
  - Value: <p1, p2, …, pn>

- Output:
  - Type 1
    - Key: <p1> (<p2>, <p3>, …)
    - Value: <PR L>
  - Type 2
    - Key: <p0>
    - Value: <p1, p2, …, pn>



X,PR_x	Y,Z

Z	PR_x, L_x
Y	PR_y, L_x
X	Y, Z



A, 0.25 | B,C,D

B, 0.25 | A, D

C, 0.25 | A

D, 0.25 | B,C

B | 0.25, 3
C | 0.25, 3
D | 0.25, 3
A | B,C,D

A | 0.25, 2
D | 0.25, 2
B | A, D

A | 0.25, 1
C | A

B | 0.25, 2
C | 0.25, 2
D | B, C

A | 0.25, 2
0.25, 1
B,C,D

B | 0.25, 3
0.25, 2
A, D

C | 0.25, 3
0.25, 2
A

D | 0.25, 3
0.25, 2
D, C

A, 0.375 | B,C,D

B, 0.208 | A, D

C, 0.208 | A

D, 0.208 | B,C

Labs 3: Page Rank (MapReduce)

- RankCalculateMapper
  - For each linked to page, store (page, thisPagesRank + TotalLinksNumber)

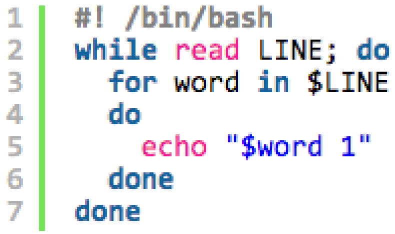
- RankCalculateReducer
  - Calculate fraction pagerank contributed from linked page.
  - Sum up all contributed pagerank.

用其他語言做word count
Hadoop Streaming

- 用其他語言分別撰寫map與reduce
- 限制：
  - mapper和reducer只能從stdin一行一行讀取資料
  - Mapper與reducer的結果都是送至stdout
  - Mapper是透過tab區分KEY/VALUE，若無tab，則整行為key，value為null
  - Java 的reducer的輸入為<key, list of value>, 但Streaming 的redcuer 程式需負責資料分組

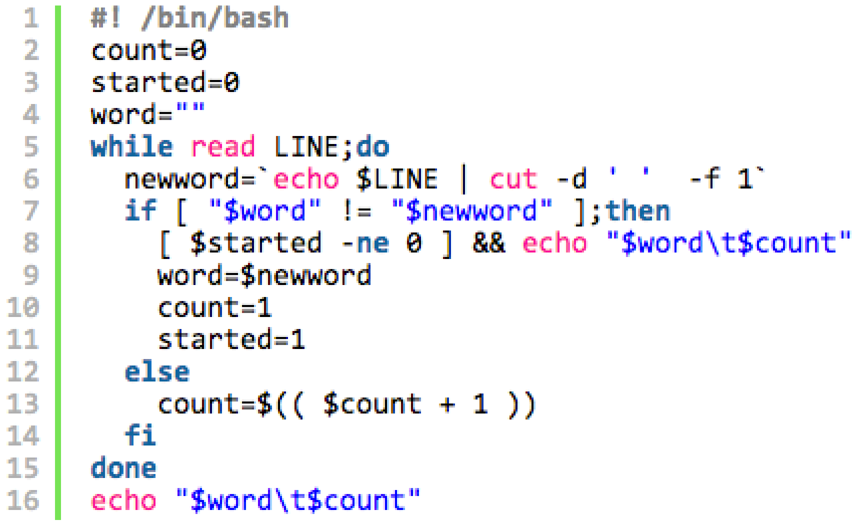
bash範例

详细版，每行可有多个单词（由史江明编写）： **mapper.sh**



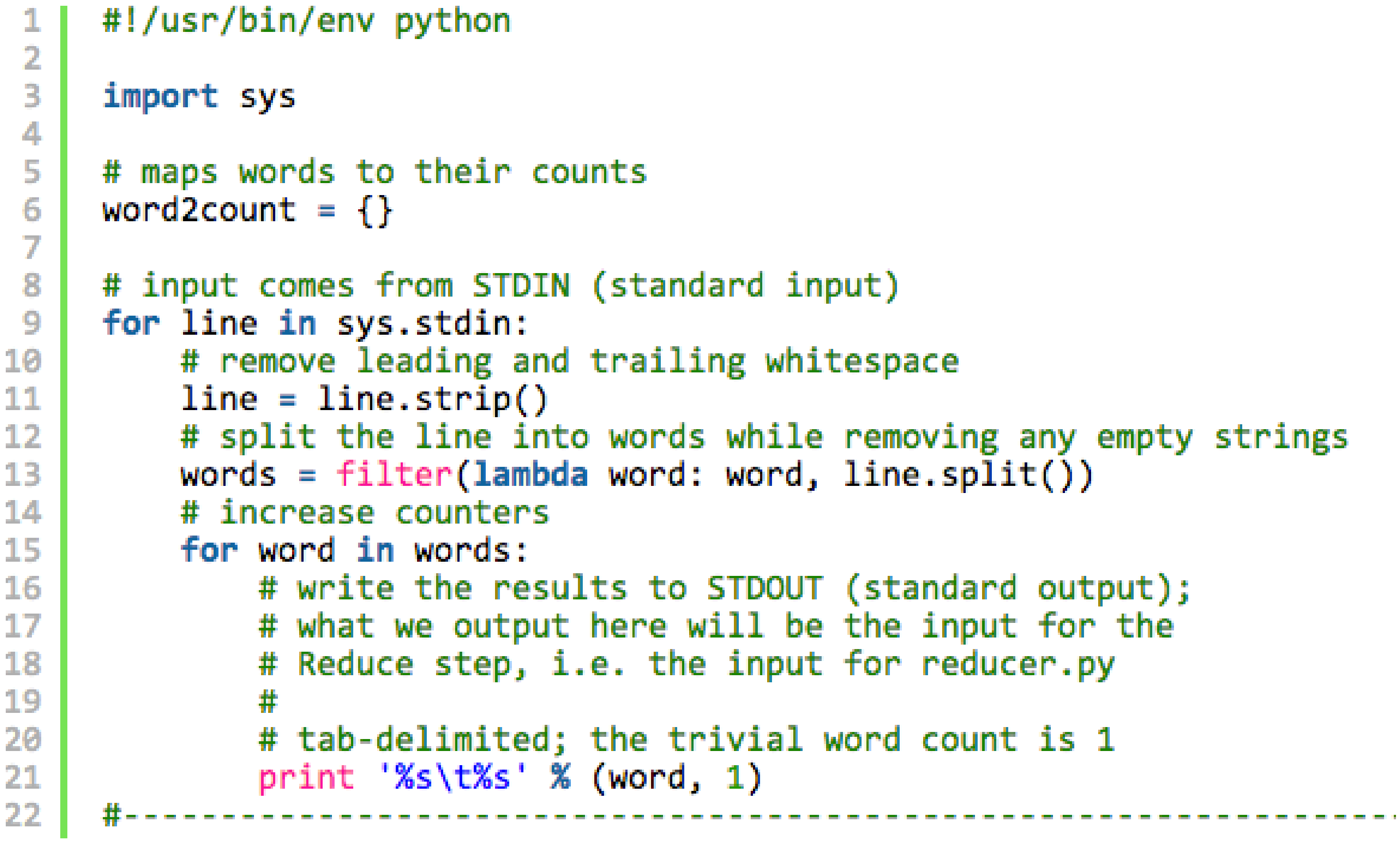
```
#! /bin/bash
while read LINE; do
  for word in $LINE
  do
    echo "$word 1"
  done
done
```

reducer.sh



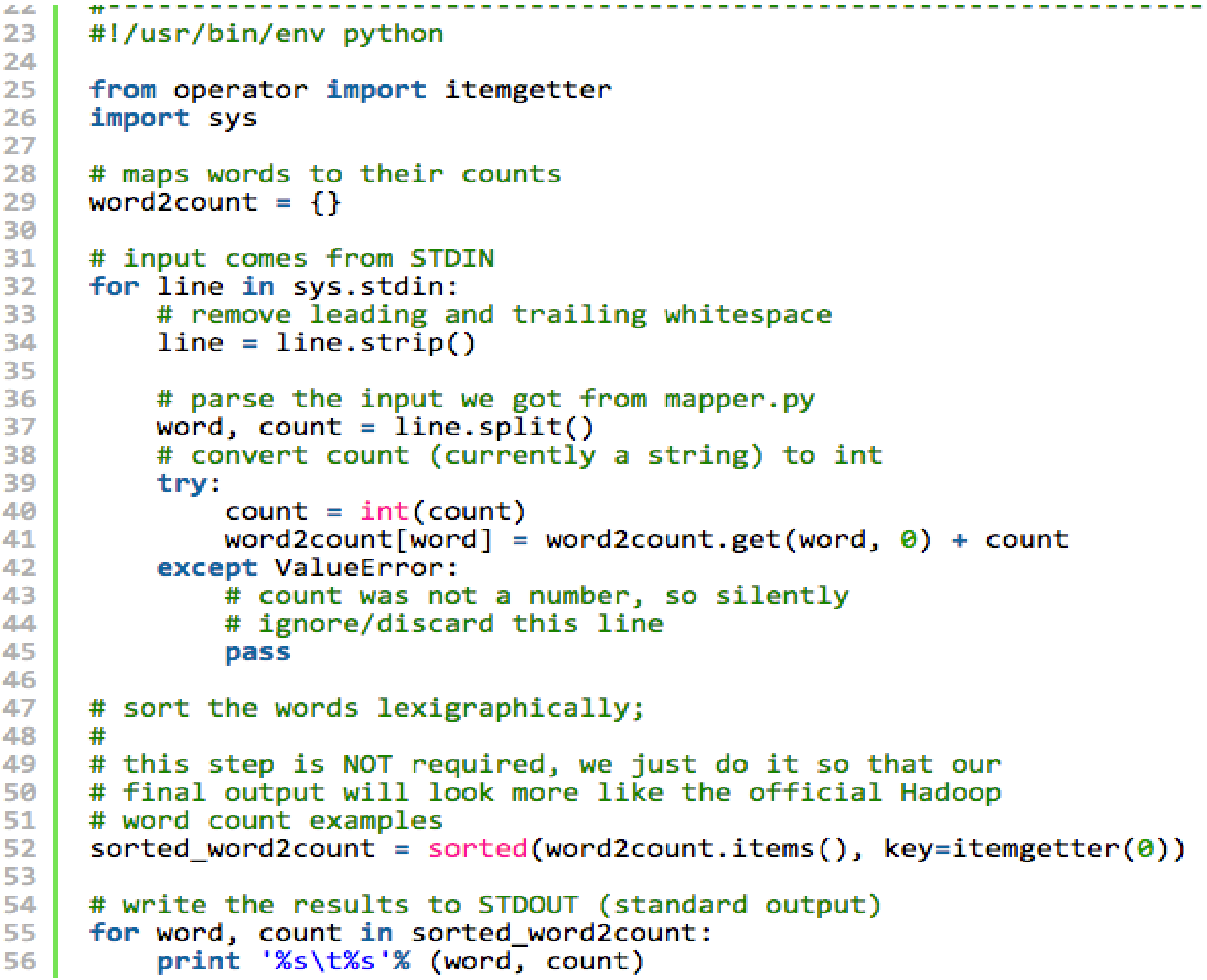
```
#! /bin/bash
count=0
started=0
word=""
while read LINE;do
  newword=`echo $LINE | cut -d ' '  -f 1`
  if [ "$word" != "$newword" ];then
    [ $started -ne 0 ] && echo "$word\t$count"
    word=$newword
    count=1
    started=1
  else
    count=$(( $count + 1 ))
  fi
done
echo "$word\t$count"
```

Python 範例: mapper



```python
#!/usr/bin/env python

import sys

# maps words to their counts
word2count = {}

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words while removing any empty strings
    words = filter(lambda word: word, line.split())
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
#-----------------------------------------------------------------
```

Python 範例: reducer



```
#-----------------------------------------------------------------
#!/usr/bin/env python

from operator import itemgetter
import sys

# maps words to their counts
word2count = {}

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split()
    # convert count (currently a string) to int
    try:
        count = int(count)
        word2count[word] = word2count.get(word, 0) + count
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        pass

# sort the words lexigraphically;
#
# this step is NOT required, we just do it so that our
# final output will look more like the official Hadoop
# word count examples
sorted_word2count = sorted(word2count.items(), key=itemgetter(0))

# write the results to STDOUT (standard output)
for word, count in sorted_word2count:
    print '%s\t%s'% (word, count)
```

Labs 4:

- 本地測試
  - cat wc.in | ./map.sh | sort –k 1 | ./reduce.sh

-
  ```
  $hadoop jar hadoop-streaming-2.5.0-cdh5.3.2.jar \
  -mapper ./mapper.py  \
  -reducer ./reducer.py \
  -file ./mapper.py \
  -file ./reducer.py \
  -input wc.in \
  -output out
  ```