教育訓練課程

Map Reduce 介紹

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw

財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

Computing with big datasets is a fundamentally different challenge than doing “big compute” over a small dataset

By the numbers…

- Max data in memory: 32 GB
- Max data per computer: 12 TB
- Data processed by Google every month: 400 PB … in 2007
- Average job size: 180 GB
- Time that would take to read sequentially
- off a single drive: 45 minutes

So …

- We can process data very quickly but we can read/write it very slowly
  - Solution: parallel reads
  - 1 HDD = 75 MB/sec
  - 1000 HDDs = 75 GB/sec

Sharing is Slow

- Grid computing: not new
  - MPI, PVM, Condor…
- Grid focus: distribute the workload
  - NetApp filer or other SAN drives many compute nodes
- Modern focus: distribute the data
  - Reading 100 GB off a single filer would leave nodes starved – just store data locally

Sharing is Tricky

- Exchanging data requires synchronization
  - Deadlock becomes a problem
- Finite bandwidth is available
  - Distributed systems can “drown themselves”
  - Failovers can cause cascading failure
- Temporal dependencies are complicated
  - Difficult to reason about partial restarts

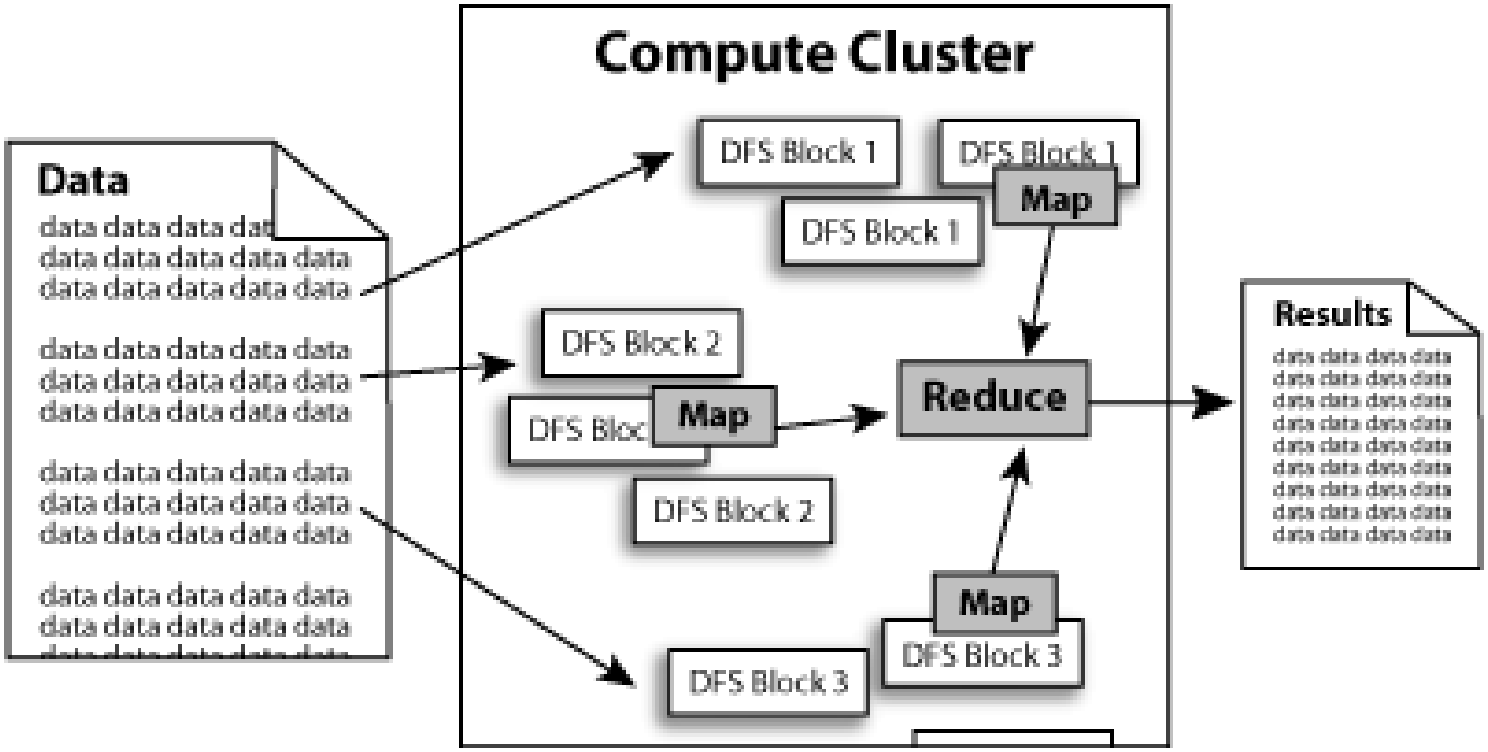
Motivations for MapReduce

- Data processing: > 1 TB
- Massively parallel
  - hundreds or thousands of CPUs
- Must be easy to use
  - High-level applications written in MapReduce
  - Programmers don't worry about socket(), etc.

How MapReduce is Structured

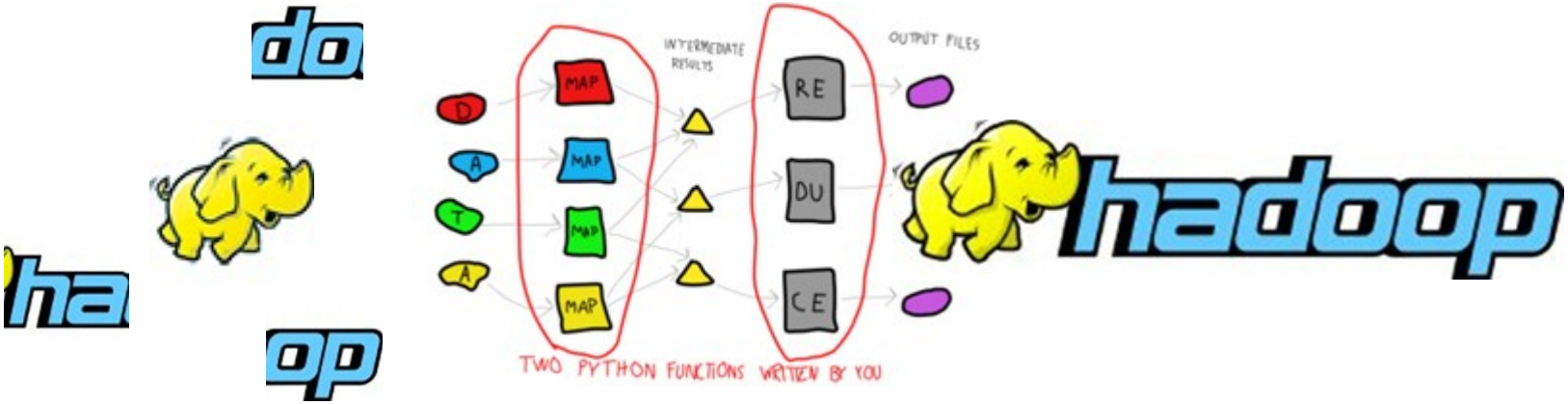
- Functional programming meets distributed computing
- A batch data processing system
- Factors out many reliability concerns from application logic

By Using Map / Reduce



Map-Reduce is a framework for computing certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster.

一句話



Hadoop MapReduce 是一套儲存並處理 petabytes 等級資訊的雲端運算技術

MapReduce Provides

- Automatic parallelization & distribution
- Fault-tolerance
- Status and monitoring tools
- A clean abstraction for programmers

Hadoop 適用於 ..

- 大規模資料集
- 可拆解
  - Text tokenization
  - Indexing and Search
  - Data mining
  - machine learning
  - …

- http://www.dbms2.com/2008/08/26/known-applications-of-mapreduce/
- http://wiki.apache.org/hadoop/PoweredBy

Hadoop Applications (1)

- Adobe
  - use Hadoop and HBase in several areas from social services to structured data storage and processing for internal use.
- Adknowledge - Ad network
  - used to build the recommender system for behavioral targeting, plus other clickstream analytics
- Alibaba
  - processing sorts of business data dumped out of database and joining them together. These data will then be fed into iSearch, our vertical search engine.
- AOL
  - We use hadoop for variety of things ranging from ETL style processing and statistics generation to running advanced algorithms for doing behavioral analysis

Hadoop Applications (2)

- Baidu - the leading Chinese language search engine
  - Hadoop used to analyze the log of search and do some mining work on web page database
- Contextweb - ADSDAQ Ad Excange
  - use Hadoop to store ad serving log and use it as a source for Ad optimizations/Analytics/reporting/machine learning.
- Detikcom - Indonesia's largest news portal
  - use hadoop, pig and hbase to analyze search log, generate Most View News,
  - generate top wordcloud, and analyze all of our logs

Hadoop Applications (3)

- DropFire
  - generate Pig Latin scripts that describe structural and semantic conversions between data contexts
  - use Hadoop to execute these scripts for production-level deployments
- Facebook
  - use Hadoop to store copies of internal log and dimension data sources
  - use it as a source for reporting/analytics and machine learning.
- Freestylers - Image retrieval engine
  - use Hadoop 影像處理
- Hosting Habitat
  - 取得所有 clients 的軟體資訊
  - 分析並告知 clients 未安裝或未更新的軟體

Hadoop Applications (4)

- IBM
  - Blue Cloud Computing Clusters
- ICCS
  - 用 Hadoop and Nutch to crawl Blog posts 並分析之
- IIIT, Hyderabad
  - We use hadoop 資訊檢索與提取
- Journey Dynamics
  - 用 Hadoop MapReduce 分析 billions of lines of GPS data 並產生交通路線資訊 .
- Krugle
  - 用 Hadoop and Nutch 建構 原始碼搜尋引擎

Hadoop Applications (5)

- SEDNS - Security Enhanced DNS Group
  - 收集全世界的 DNS 以探索網路分散式內容 .
- Technical analysis and Stock Research
  - 分析股票資訊
- University of Maryland
  - 用 Hadoop 執行 machine translation, language modeling, bioinformatics, email analysis, and image processing 相關研究
- University of Nebraska Lincoln, Research Computing Facility
  - 用 Hadoop 跑約 200TB 的 CMS 經驗分析
  - **緊湊渺子線圈**（ **CMS** ， **C**ompact **M**uon **S**olenoid ）為瑞士歐洲核子研究組織 CERN 的大型強子對撞器計劃的兩大通用型粒子偵測器中的一個。

Hadoop Applications (6)

- PARC
  - Used Hadoop to analyze Wikipedia conflicts
- Search Wikia
  - A project to help develop open source social search tools
- Yahoo!
  - Used to support research for Ad Systems and Web Search
  - 使用 Hadoop 平台來發現發送垃圾郵件的殭屍網絡
- 趨勢科技
  - 過濾像是釣魚網站或惡意連結的網頁內容

MapReduce Conclusions

- MapReduce has proven to be a useful abstraction in many areas
- Greatly simplifies large-scale computations
- Functional programming paradigm can be applied to large-scale applications
- You focus on the “real” problem, library deals with messy details

Map Reduce
原理說明

王耀聰　陳威宇

Jazz@nchc.org.tw

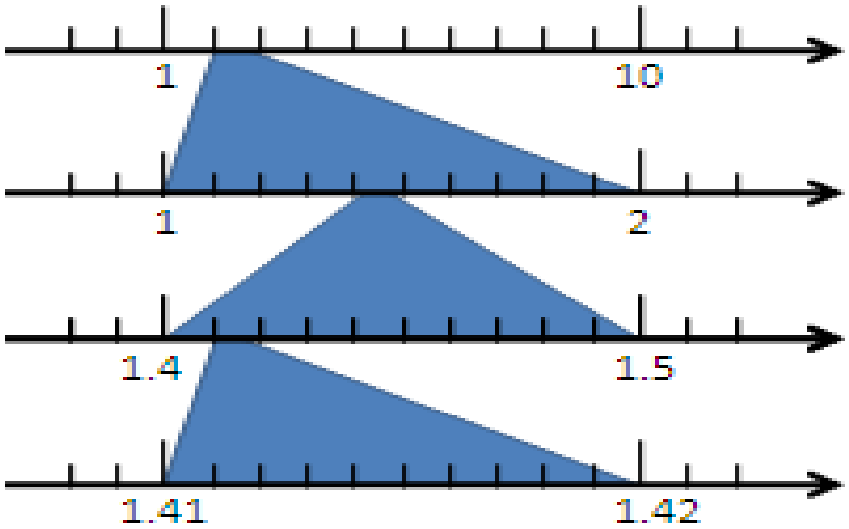
waue@nchc.org.tw

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

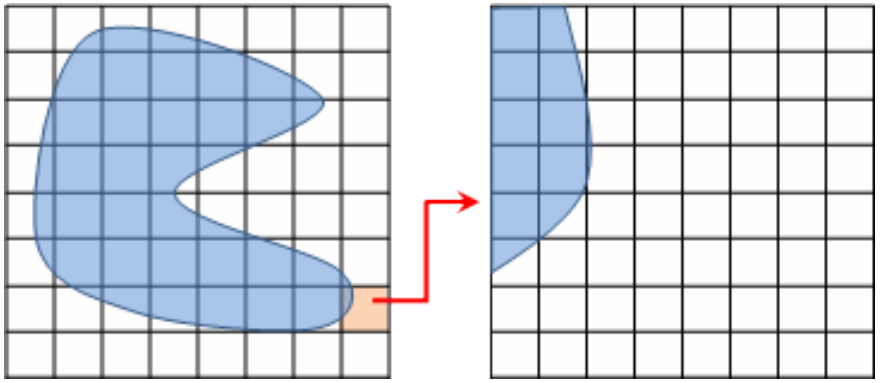
Algorithms

- Functional Programming : Map Reduce
  - map(...) :
    - [1,2,3,4] – (*2) -> [2,4,6,8]
  - reduce(...):
    - [1,2,3,4] - (sum) -> 10
  - 對應演算法中的 Divide and conquer
  - 將問題分解成很多個小問題之後，再做總和

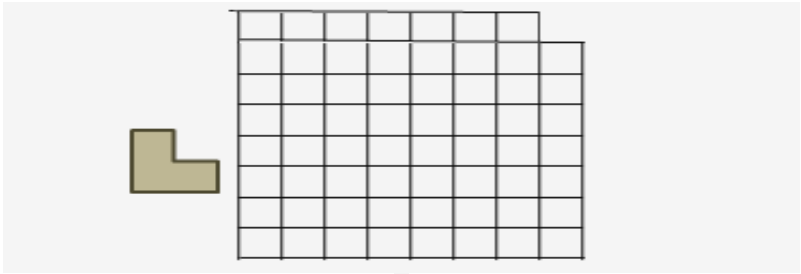
Divide and Conquer



sqrt(2)



範例三：鋪滿　L　形磁磚



範例四：
眼前有五階樓梯，每次可踏上一階或踏上兩階，那麼爬完五階共有幾種踏法？
Ex ： (1,1,1,1,1) or (1,2,1,1)

Programming Model

- Users implement interface of two functions:
  - map (in_key, in_value) → (out_key, intermediate_value) list
  - reduce (out_key, intermediate_value list) → out_value list

Map

- One-to-one Mapper

```
let map(k, v) =
Emit(k.toUpper(),
v.toUpper())
```

("Foo", "other") → ("FOO", "OTHER")
("key2", "data") → ("KEY2", "DATA")

- Explode Mapper

```
let map(k, v) =
foreach char c in
v:
  emit(k, c)
```

("A", "cats") → ("A", "c"), ("A", "a"),

("A", "t"), ("A", "s")

- Filter Mapper

```
let map(k, v) =
    if
    (isPrime(v))
```

("foo", 7) → ("foo", 7)
("test", 10) → (nothing)

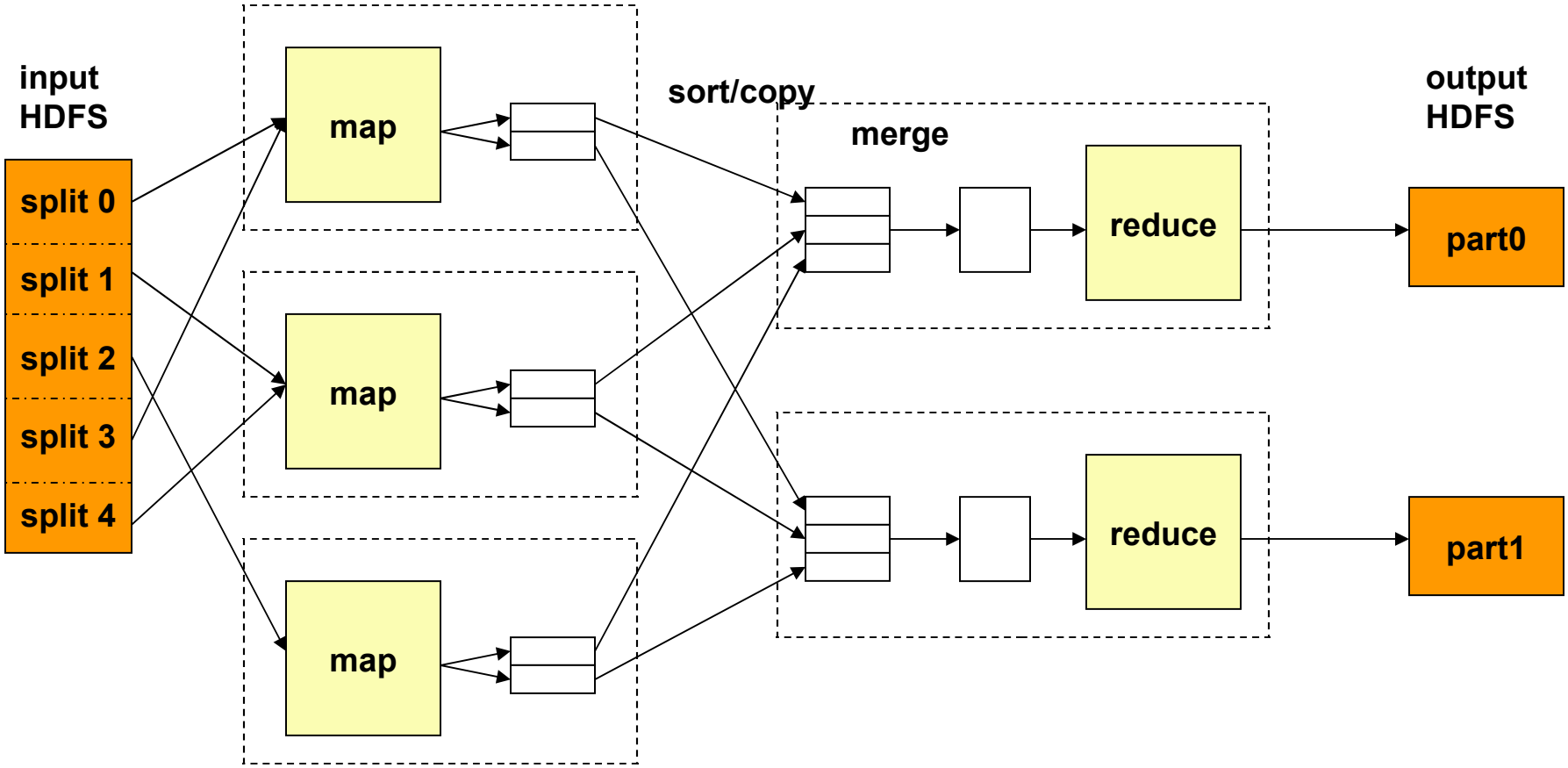
Reduce

Example: Sum Reducer

```
let reduce(k, vals) =
    sum = 0
    foreach int v in vals:
        sum += v
    emit(k, sum)
```

("A", [42, 100, 312]) → ("A", 454)
("B", [12, 6, -2]) → ("B", 16)

MapReduce 運作流程



JobTracker 跟 NameNode 取得需要運算的 blocks

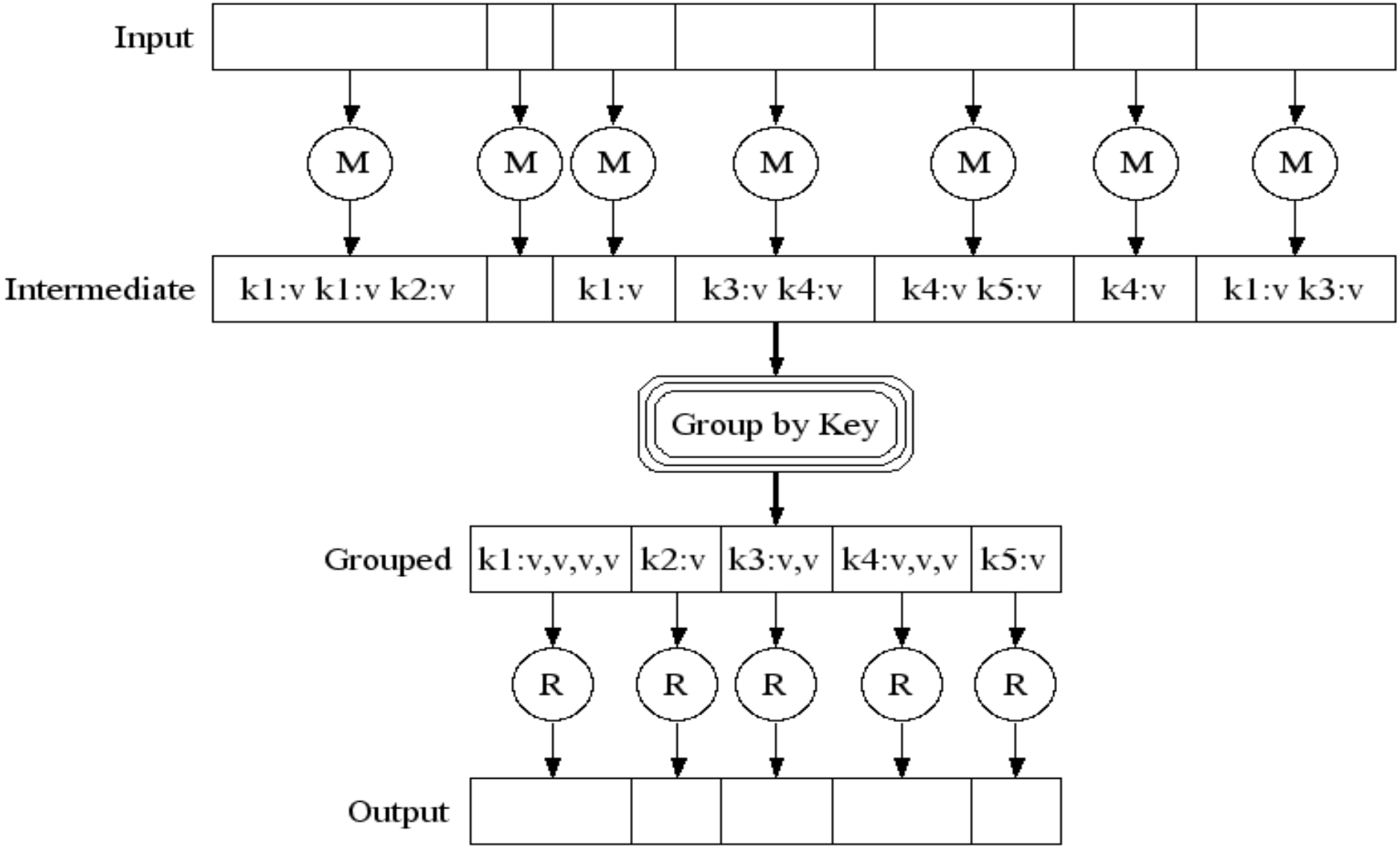
JobTracker 選數個 TaskTracker 來作 Map 運算，產生些中間檔案

JobTracker 將中間檔案整合排序後，複製到需要的 TaskTracker 去

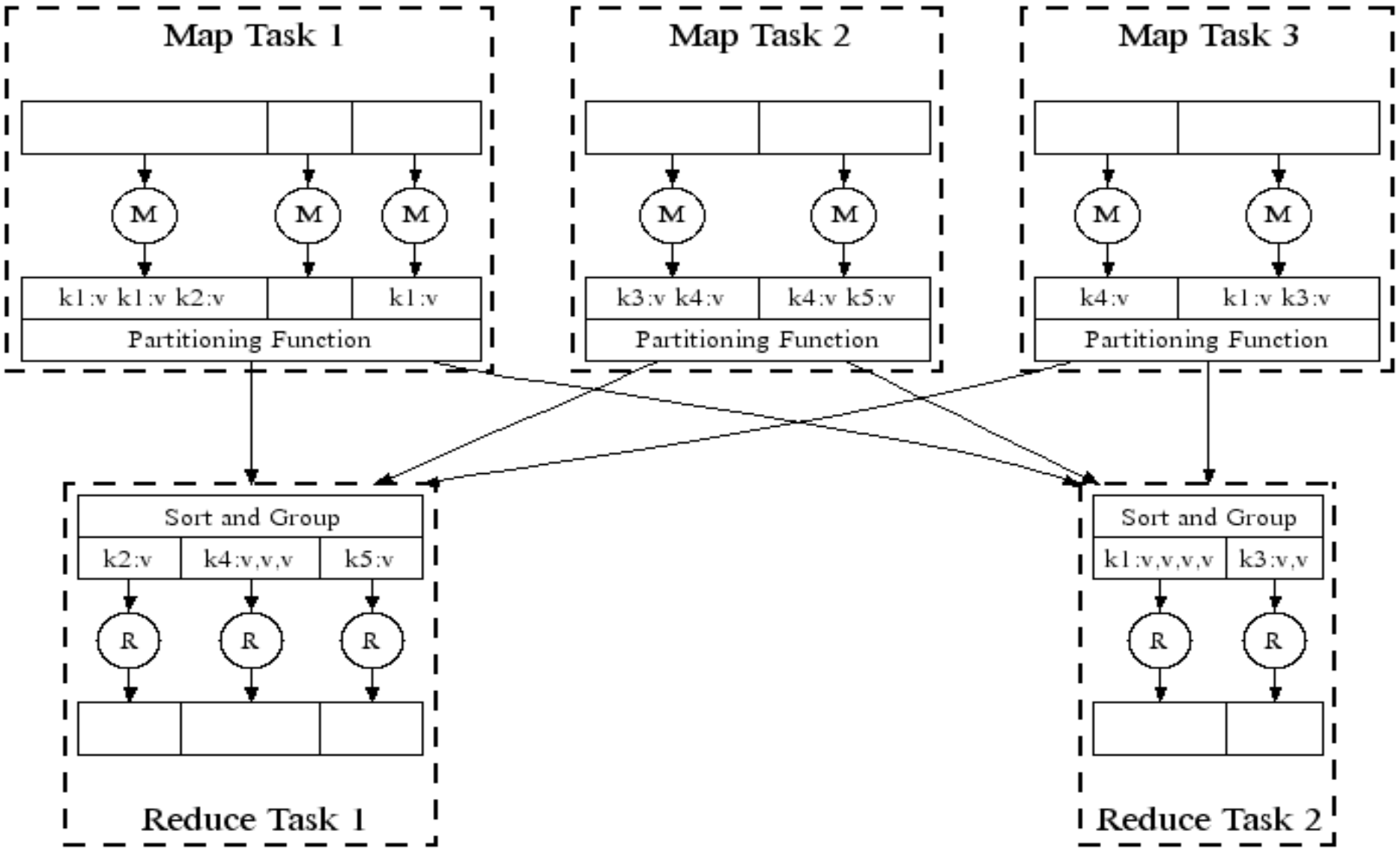
JobTracker 派遣 TaskTracker 作 reduce

reduce 完後通知 JobTracker 與 Namenode 以產生 output

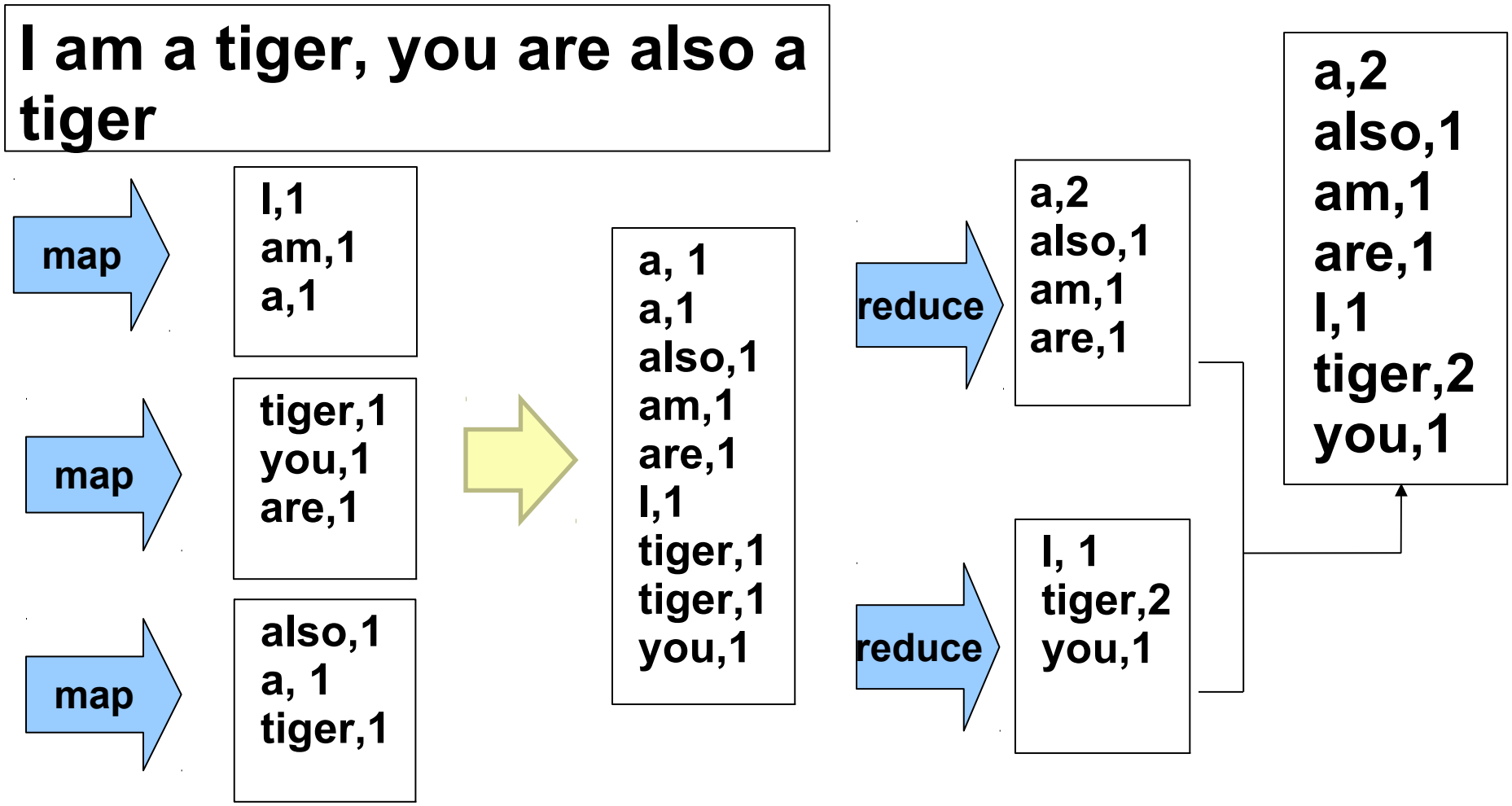
MapReduce 圖解



MapReduce in Parallel



範例



JobTracker 先選了三個 Tracker 做 map

Map 結束後， hadoop 進行中間資料的整理與排序

JobTracker 再選兩個 TaskTracker 作 reduce

Console 端 編譯與執行

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

Java 之編譯與執行

1. 編譯
   - javac △ -classpath △ hadoop-*-core.jar △ -d △ MyJava △ MyCode.java
1. 封裝
   - jar △ -cvf △ MyJar.jar △ -C △ MyJava △ .
1. 執行
   - bin/hadoop △ jar △ MyJar.jar △ MyCode △ HDFS_Input/ △ HDFS_Output/

- 所在的執行目錄為 Hadoop_Home
- ./MyJava = 編譯後程式碼目錄
- Myjar.jar = 封裝後的編譯檔
- 先放些文件檔到 HDFS 上的 input 目錄
- ./input; ./ouput = hdfs 的輸入、輸出目錄

WordCount1 練習 (I)

1. cd $HADOOP_HOME
2. bin/hadoop dfs -mkdir input
3. echo "I like NCHC Cloud Course." > inputwc/input1
4. echo "I like nchc Cloud Course, and we enjoy this crouse." > inputwc/input2
5. bin/hadoop dfs -put inputwc inputwc
6. bin/hadoop dfs -ls input

```
waue@vPro:/opt/hadoop$ bin/hadoop dfs -ls input
Found 2 items
-rw-r--r--   1 waue supergroup         26 2009-03-22 12:15 /user/waue/input/input1
-rw-r--r--   1 waue supergroup         52 2009-03-22 12:15 /user/waue/input/input2
waue@vPro:/opt/hadoop$
```

WordCount1 練習 (II)

1. 編輯 WordCount.java
   http://trac.nchc.org.tw/cloud/attachment/wiki/jazz/Hadoop_Lab6/WordCount.java?format=raw
2. mkdir MyJava
3. javac -classpath hadoop-*-core.jar -d MyJava WordCount.java
4. jar -cvf wordcount.jar -C MyJava .
5. bin/hadoop jar wordcount.jar WordCount input/ output/

- 所在的執行目錄為 **Hadoop_Home**（因為 hadoop-*-core.jar ）
- **javac** 編譯時需要 **classpath,** 但 **hadoop jar** 時不用
- **wordcount.jar =** 封裝後的編譯檔，但執行時需告知 **class name**
- **Hadoop** 進行運算時，只有 **input** 檔要放到 **hdfs** 上，以便 **hadoop** 分析運算；執行檔（ **wordcount.jar** ）不需上傳，也不需每個 **node** 都放，程式的載入交由 **java** 處理

WordCount1 練習 (III)

```
waue@vPro:/opt/hadoop$ mkdir MyJava
waue@vPro:/opt/hadoop$ javac -classpath hadoop-*-core.jar -d MyJava WordCount.java
waue@vPro:/opt/hadoop$ jar -cvf wordcount.jar -C MyJava .
新增 manifest
新增：WordCount.class (讀=1516)(寫=740)(壓縮 51%)
新增：WordCount$Reduce.class (讀=1591)(寫=642)(壓縮 59%)
新增：WordCount$Map.class (讀=1918)(寫=795)(壓縮 58%)
waue@vPro:/opt/hadoop$ bin/hadoop jar wordcount.jar WordCount input/ output/
09/03/22 11:39:01 WARN mapred.JobClient: Use GenericOptionsParser for parsing the argu
ments. Applications should implement Tool for the same.
09/03/22 11:39:01 INFO mapred.FileInputFormat: Total input paths to process : 1
09/03/22 11:39:01 INFO mapred.FileInputFormat: Total input paths to process : 1
09/03/22 11:39:02 INFO mapred.JobClient: Running job: job_200903201526_0007
09/03/22 11:39:03 INFO mapred.JobClient:  map 0% reduce 0%
09/03/22 11:39:08 INFO mapred.JobClient:  map 100% reduce 0%
09/03/22 11:39:15 INFO mapred.JobClient: Job complete: job_200903201526_0007
09/03/22 11:39:15 INFO mapred.JobClient: Counters: 16
09/03/22 11:39:15 INFO mapred.JobClient:   File Systems
09/03/22 11:39:15 INFO mapred.JobClient:     HDFS bytes read=320950
09/03/22 11:39:15 INFO mapred.JobClient:     HDFS bytes written=130568
09/03/22 11:39:15 INFO mapred.JobClient:     Local bytes read=168448
09/03/22 11:39:15 INFO mapred.JobClient:     Local bytes written=336932
09/03/22 11:39:15 INFO mapred.JobClient:   Job Counters
09/03/22 11:39:15 INFO mapred.JobClient:     Launched reduce tasks=1
```

WordCount1 練習 (IV)

```
waue@vPro:/opt/hadoop$ bin/hadoop dfs -cat output/part-00000
Cloud   2
Course, 1
Course. 1
I       2
NCHC    1
and     1
course. 1
enjoy   1
like    2
nchc    1
this    1
we      1
```

BTW …

- 雖然 Hadoop 框架是用 Java 實作，但 Map/Reduce 應用程序則不一定要用 Java 來寫
- Hadoop Streaming ：
  - 執行作業的工具，使用者可以用其他語言（如： PHP ）套用到 Hadoop 的 mapper 和 reducer
- Hadoop Pipes ： C++ API

透過 Eclipse 開發

王耀聰　陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw

財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

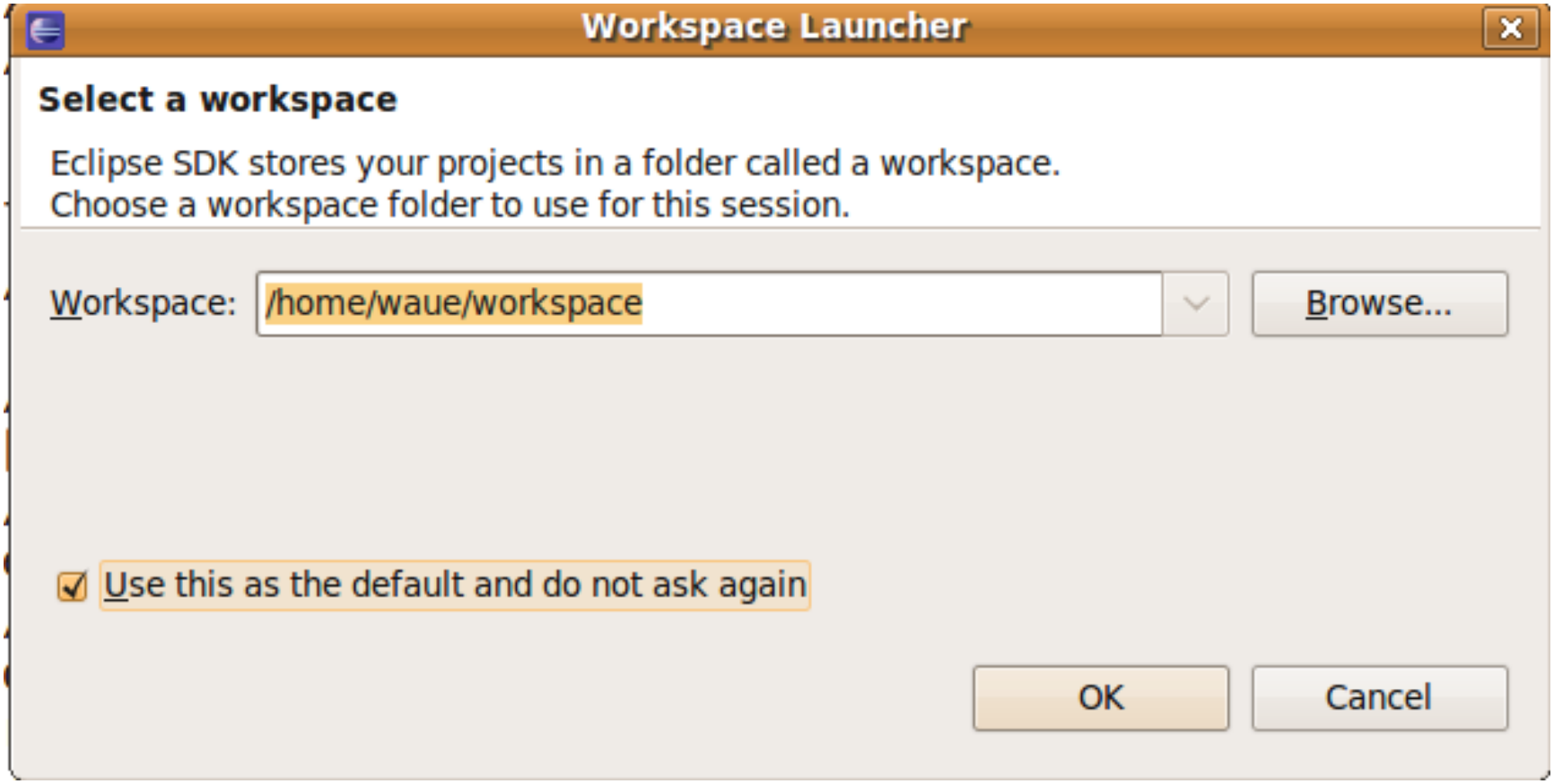
Requirements

- Hadoop 0.20.0 up
- Java 1.6
- Eclipse 3.3 up

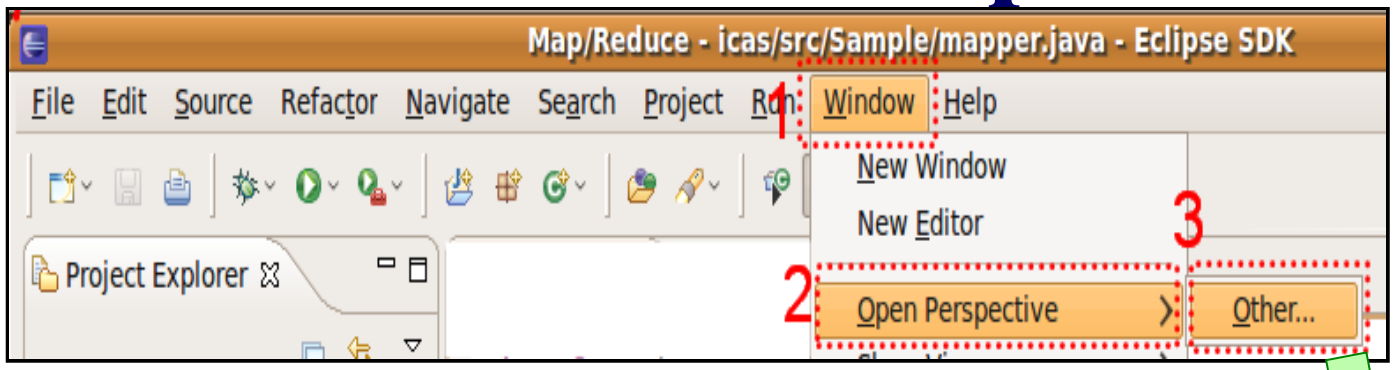
安装 Hadoop Eclipse Plugin

- Hadoop Eclipse Plugin 0.20.0
  - From $Hadoop_0.20.0_home/contrib/eclipse-plugin/hadoop-0.20.0-eclipse-plugin.jar
- Hadoop Eclipse Plugin 0.20.1
  - Compiler needed
  - Or download from http://hadoop-eclipse-plugin.googlecode.com/files/hadoop-0.20.1-eclipse-plugin.jar
- copy to $Eclipse_home/plugins/

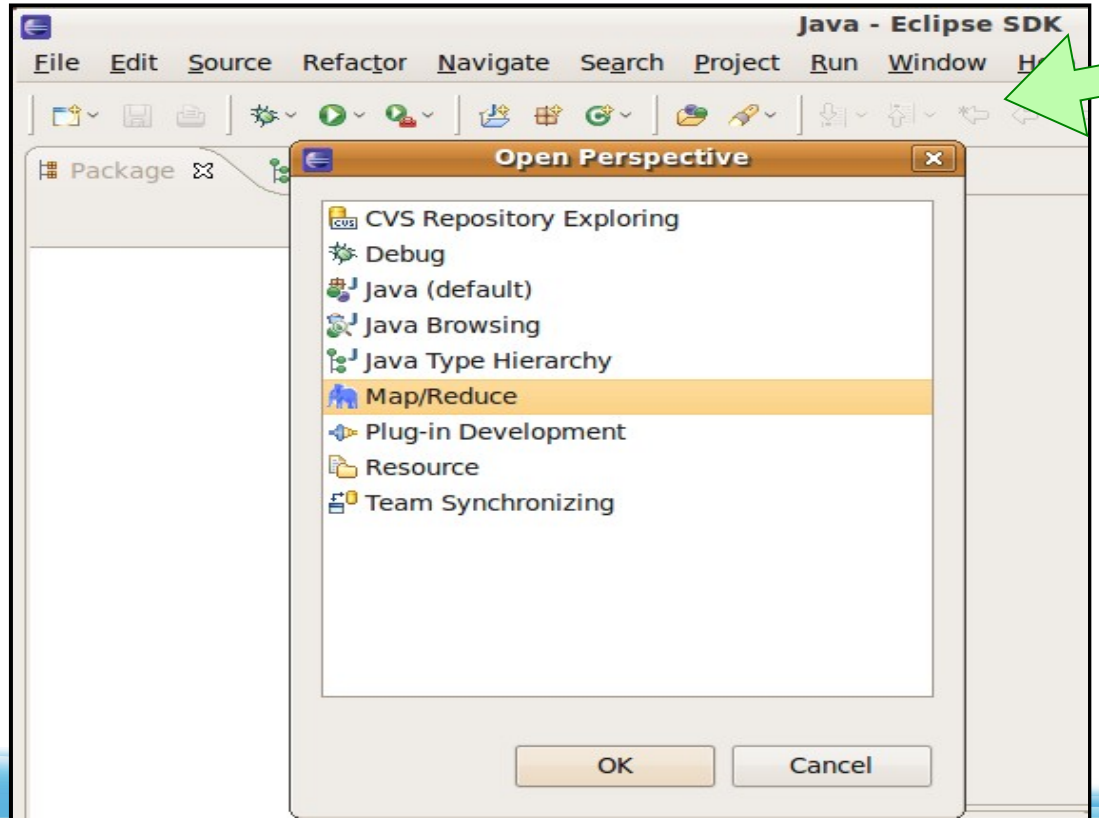
1 打開 Eclipse，設定專案目錄



2. 使用 Hadoop mode 視野

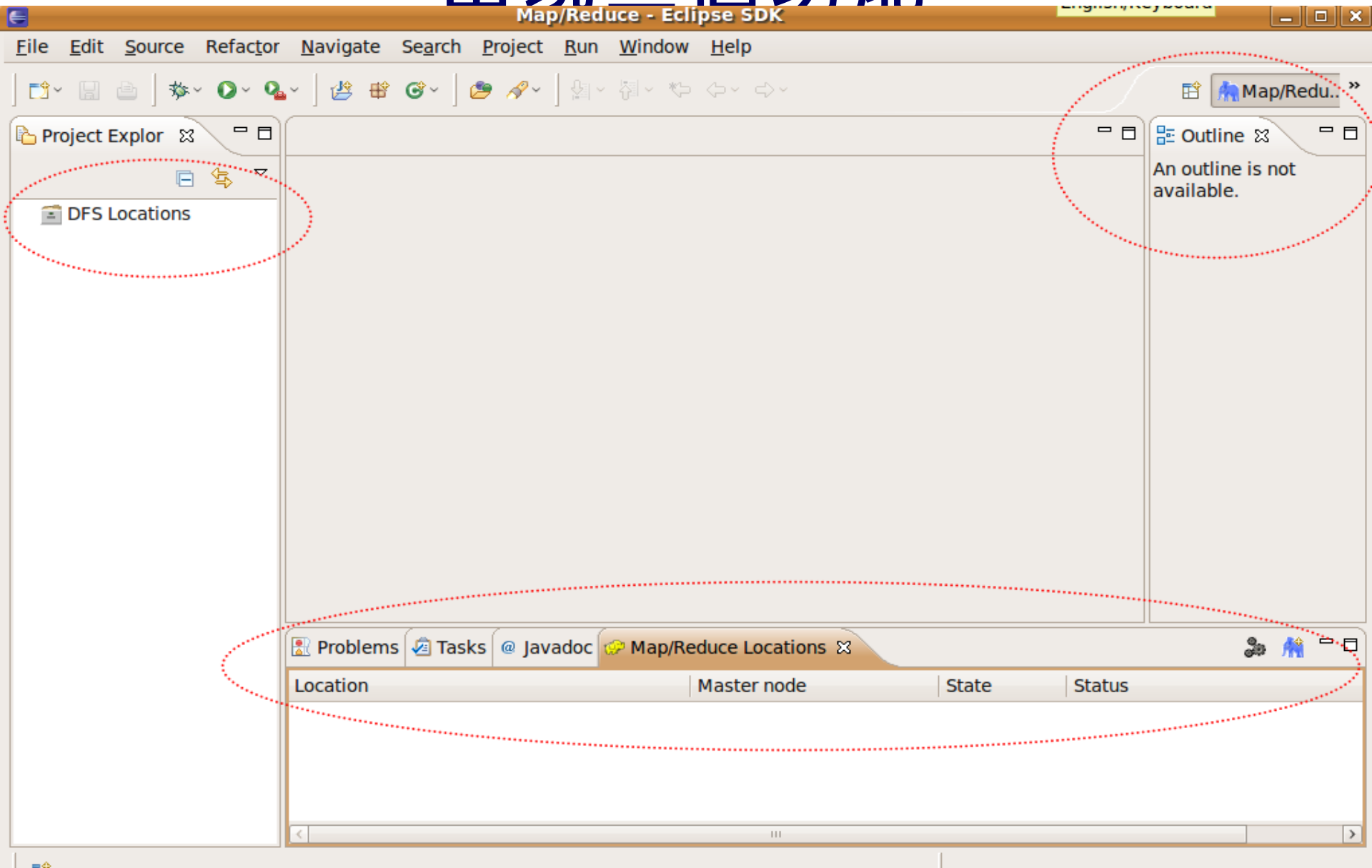


Window → Open Perspective → Other

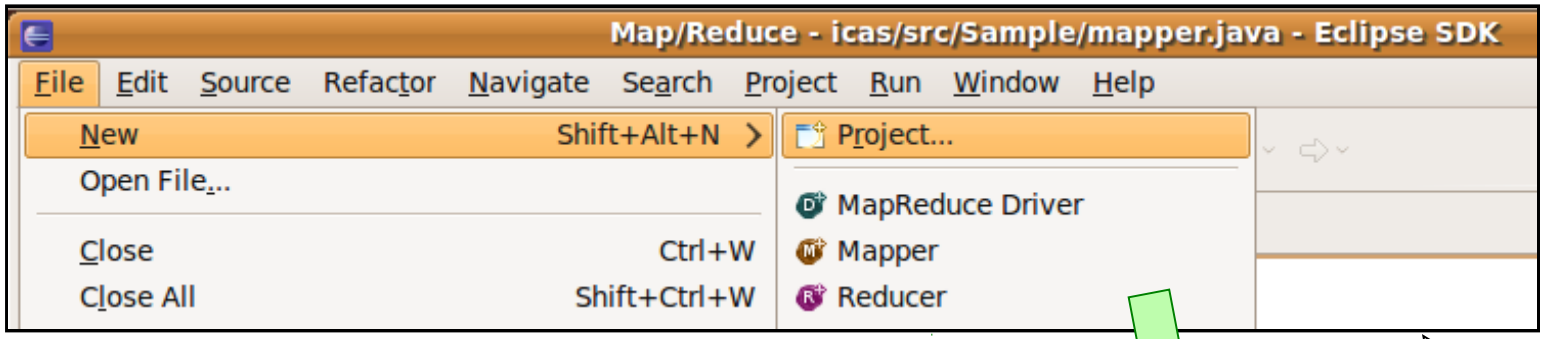


若有看到 MapReduce 的大象圖示代表 Hadoop Eclipse plugin 有安裝成功， 若沒有請檢查是否有安之裝正確

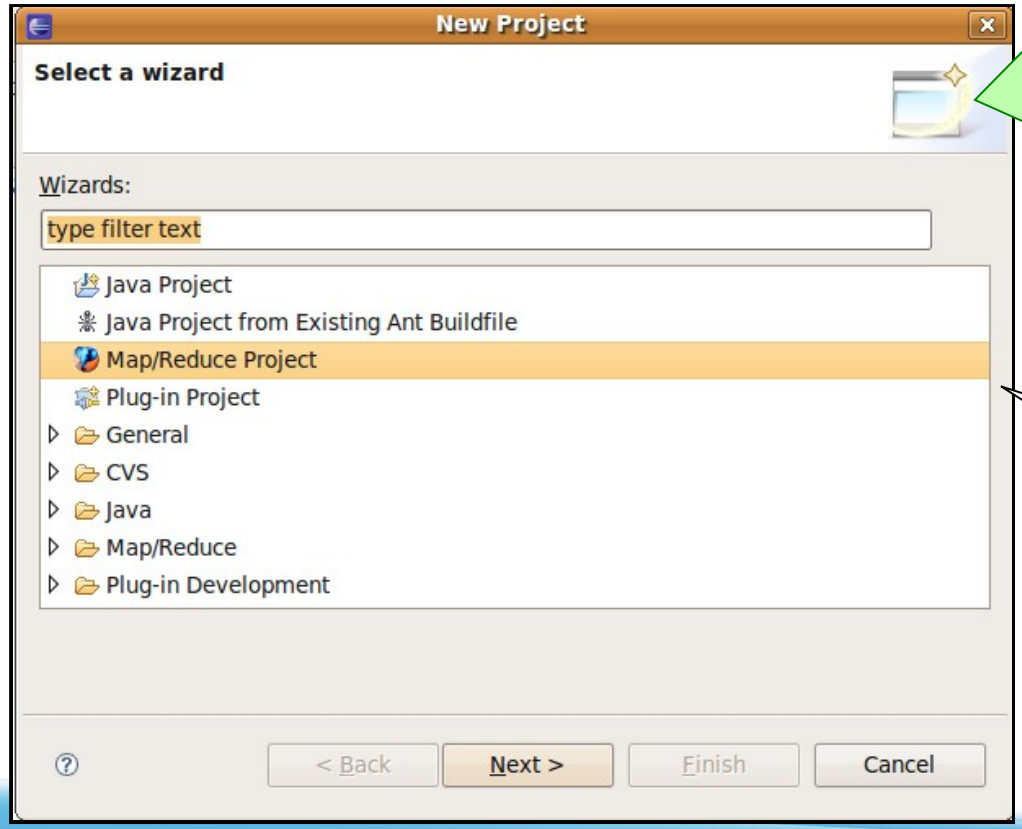
3. 使用 Hadoop 視野，主畫面將出現三個功能



4. 建立一個 Hadoop 專案

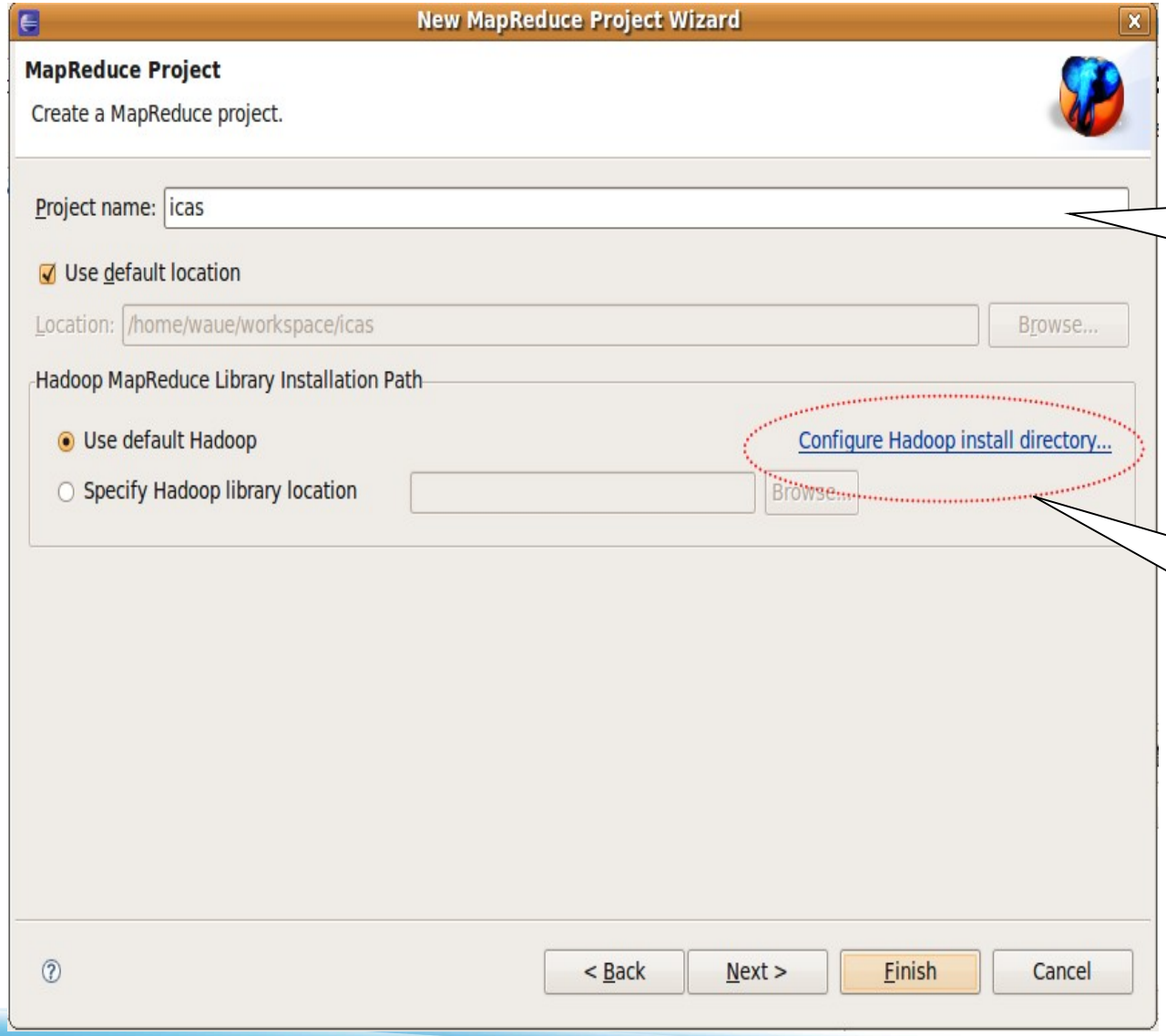


開出新專案



選擇 Map/Reduce 專案

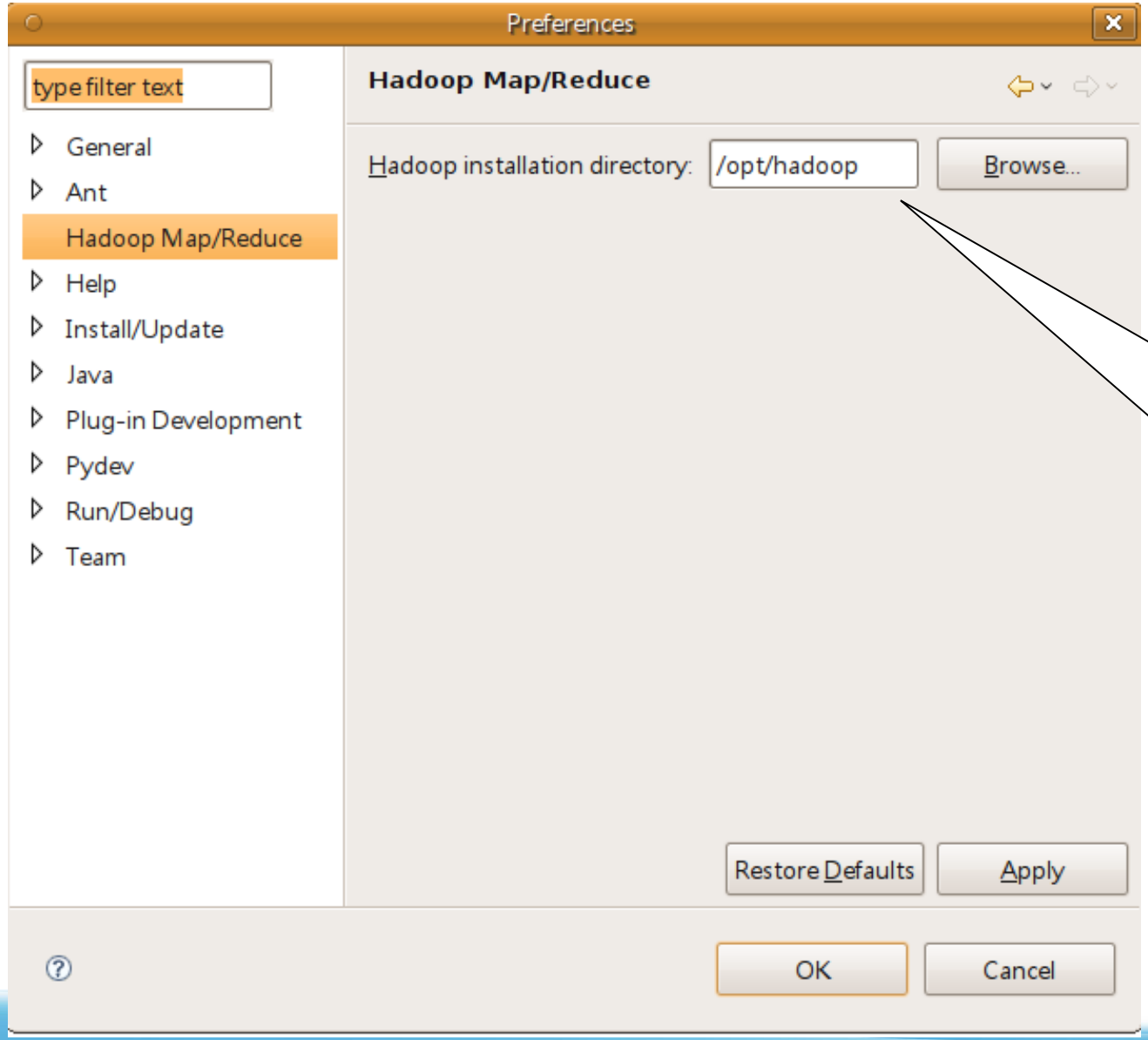
4-1. 輸入專案名稱並點選設定 Hadoop 安裝路徑



由此設定專案名稱

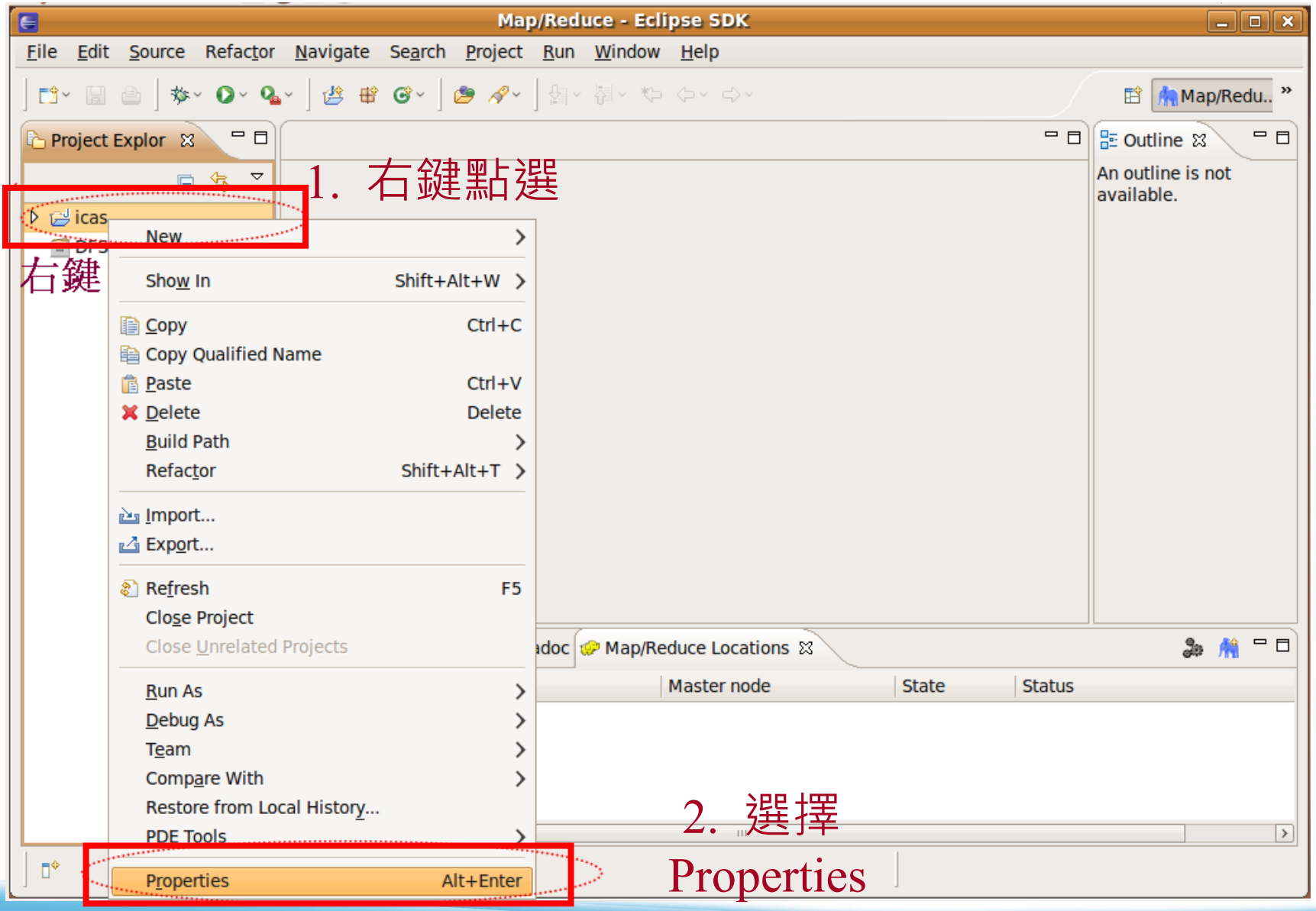
由此設定 Hadoop 的安裝路徑

4-1-1. 填入 Hadoop 安裝路徑



於此輸入您 Hadoop 的安裝路徑，之後選擇 ok

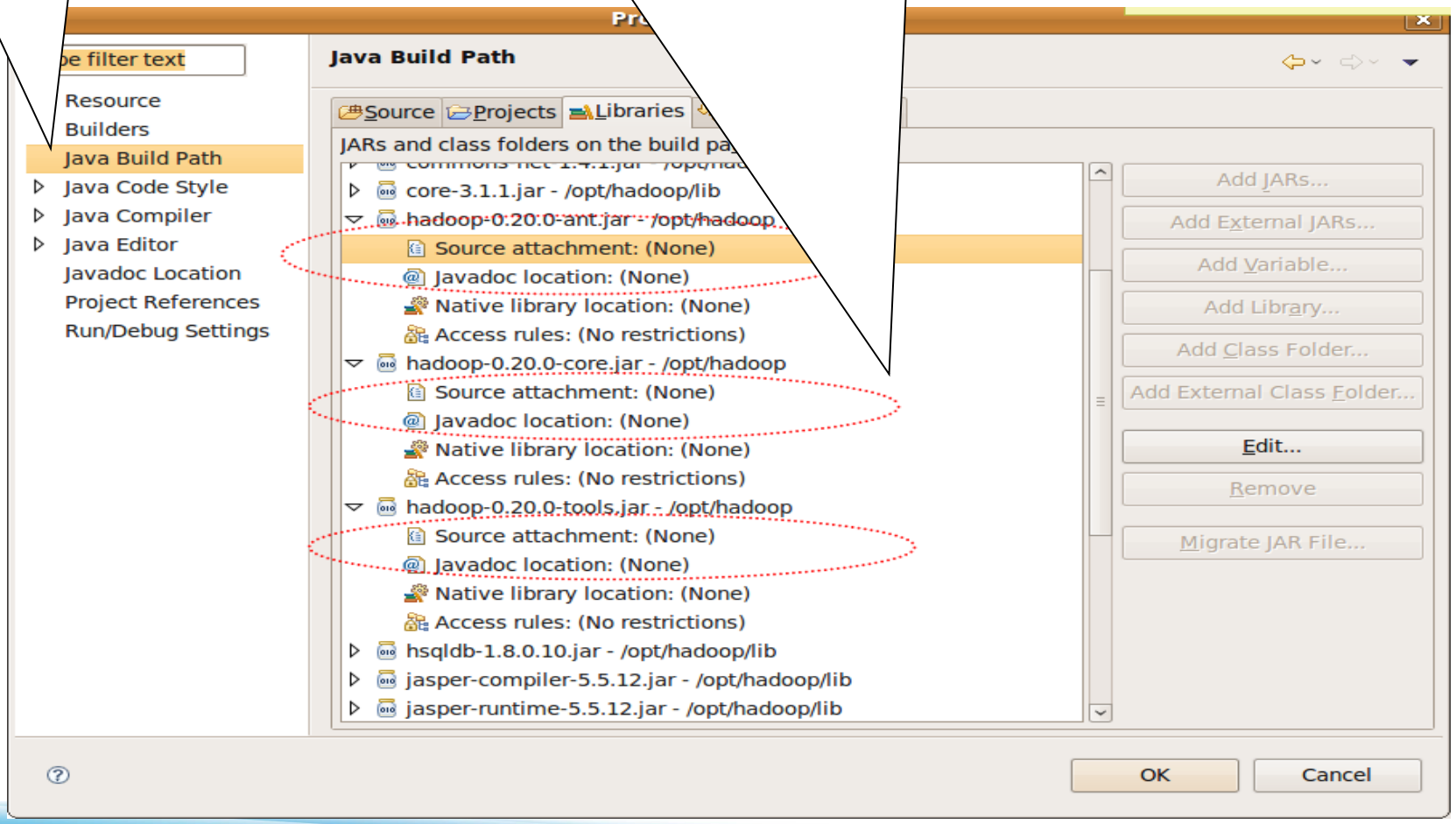
5. 設定 Hadoop 專案細節



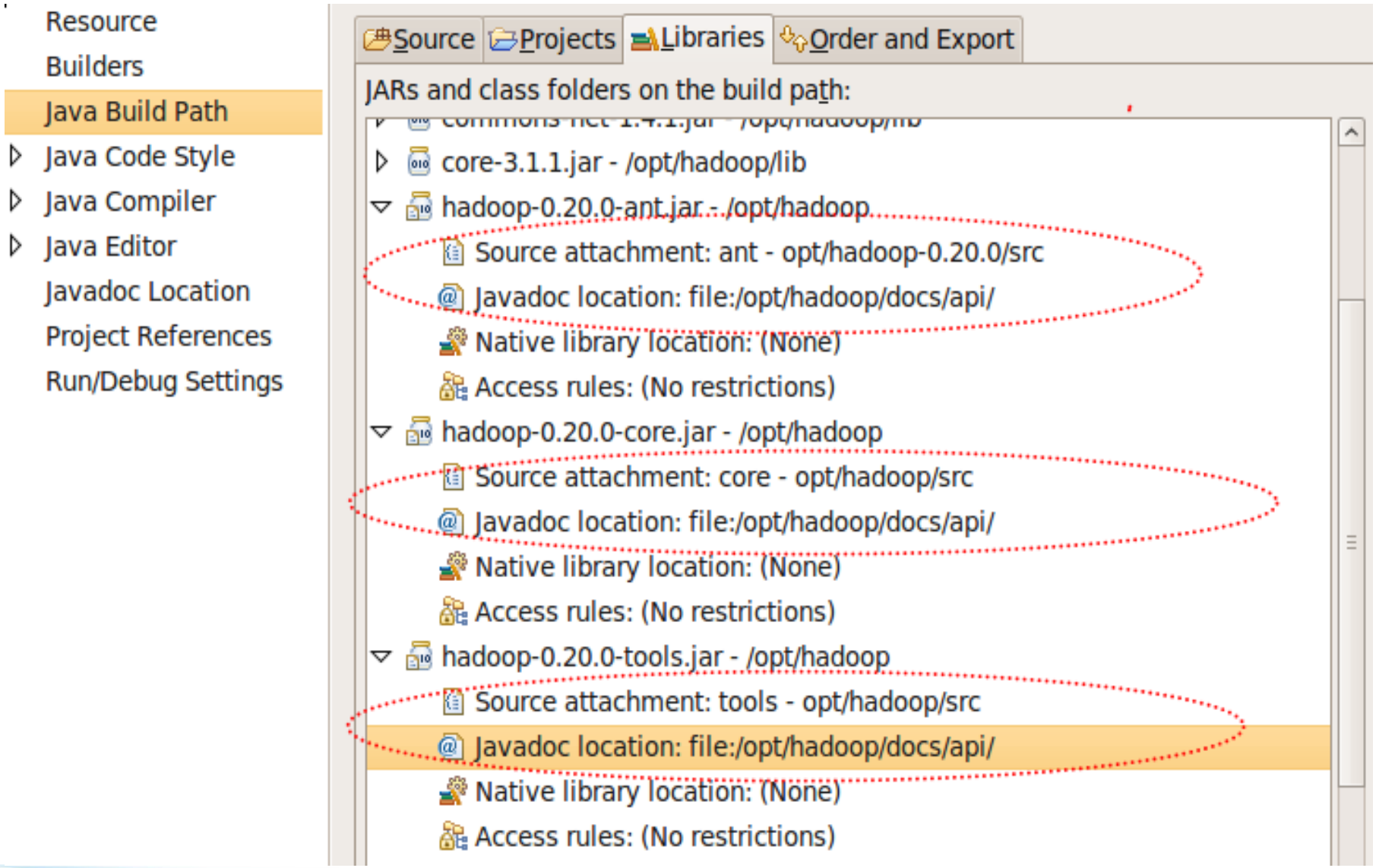
5-1. 設定原始碼與文件路徑

選擇　Java Build Path

以下請輸入正確的 Hadoop 原始碼與 API 文件檔路徑，如
source　：/opt/hadoop/src/core/
javadoc　：file:/opt/hadoop/docs/api/

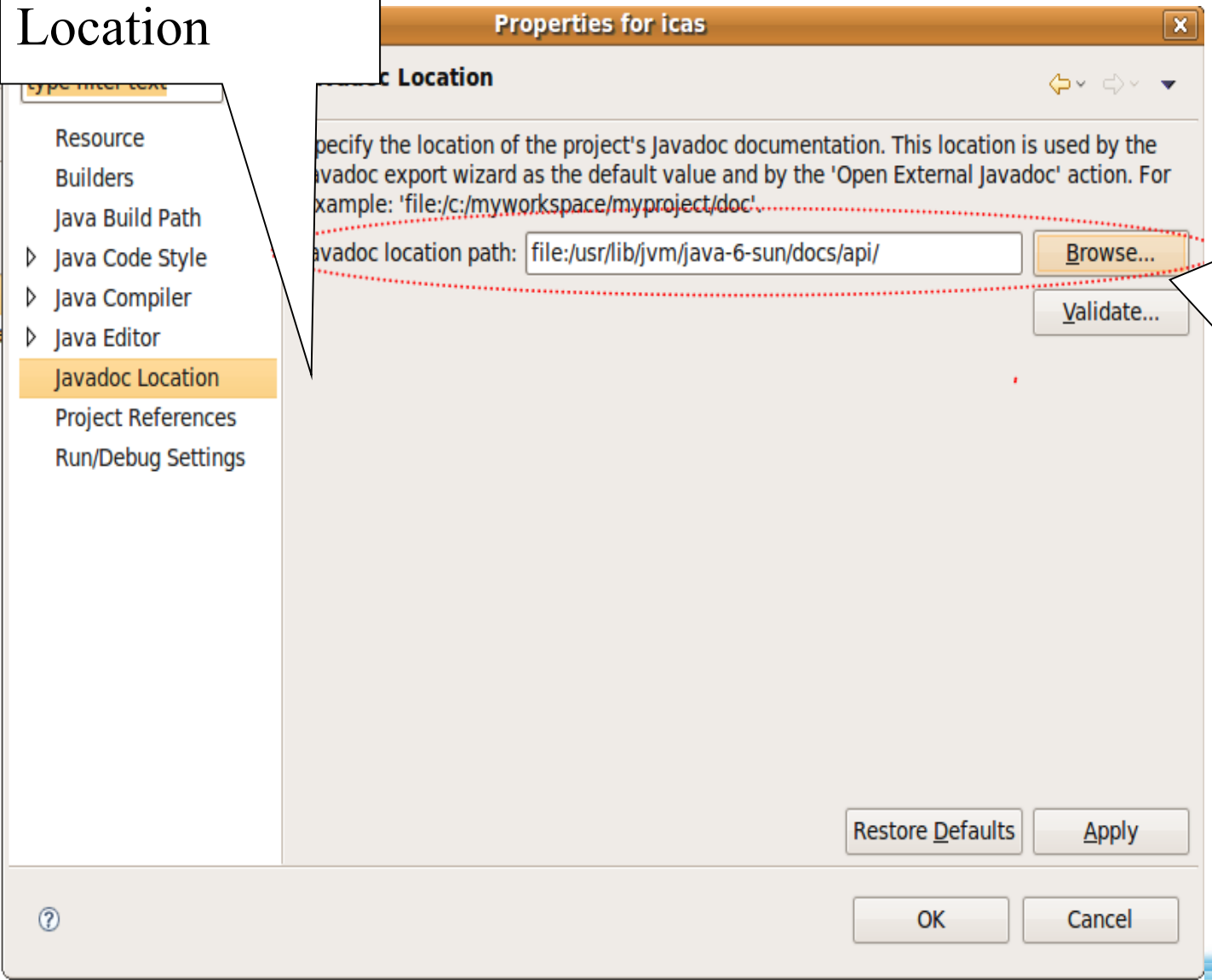


5-1-1. 完成圖


Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Run/Debug Settings
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
core-3.1.1.jar - /opt/hadoop/lib
hadoop-0.20.0-ant.jar - /opt/hadoop
Source attachment: ant - opt/hadoop-0.20.0/src
Javadoc location: file:/opt/hadoop/docs/api/
Native library location: (None)
Access rules: (No restrictions)
hadoop-0.20.0-core.jar - /opt/hadoop
Source attachment: core - opt/hadoop/src
Javadoc location: file:/opt/hadoop/docs/api/
Native library location: (None)
Access rules: (No restrictions)
hadoop-0.20.0-tools.jar - /opt/hadoop
Source attachment: tools - opt/hadoop/src
Javadoc location: file:/opt/hadoop/docs/api/
Native library location: (None)
Access rules: (No restrictions)

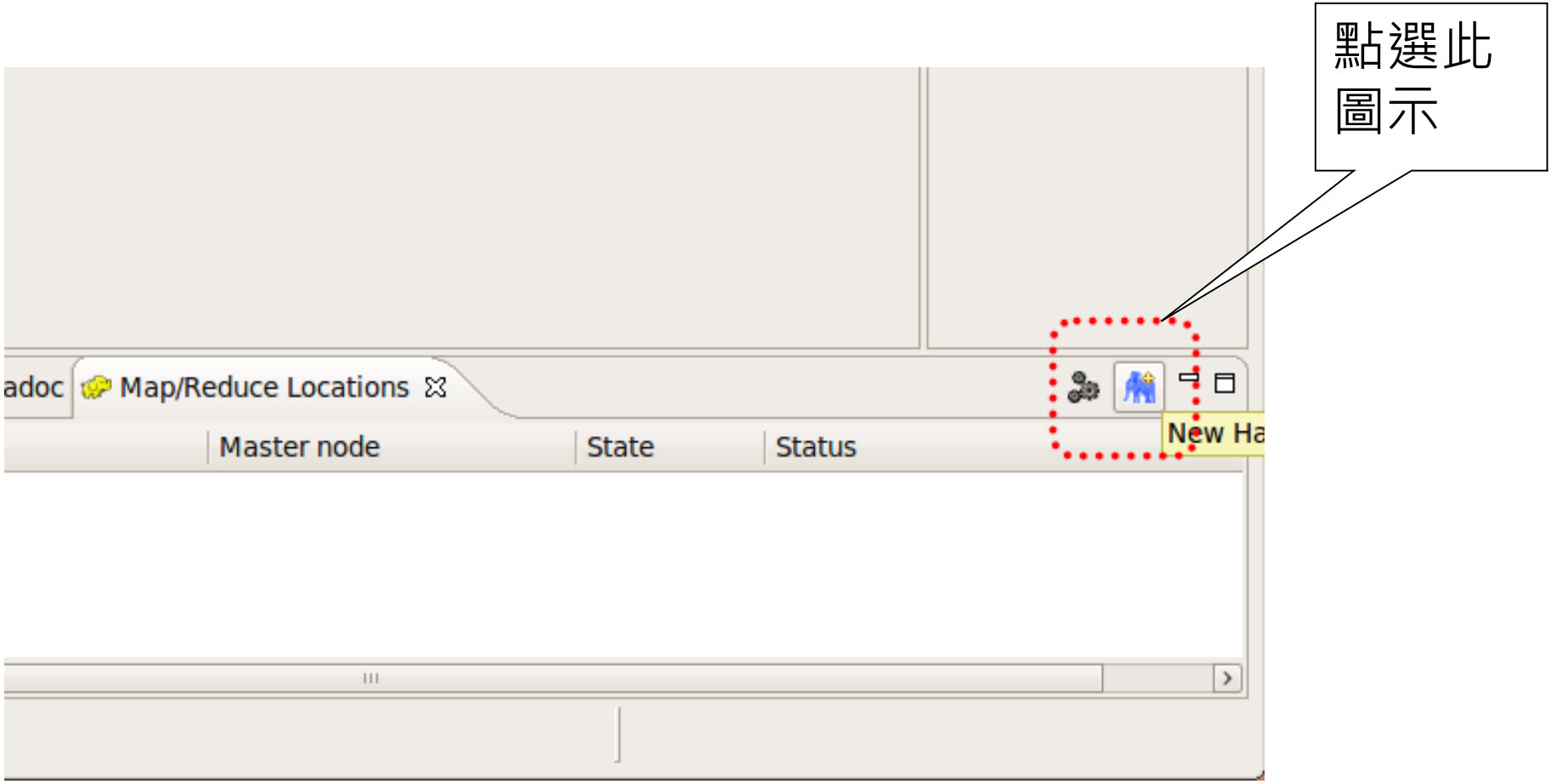
5-2. 設定 java doc 的完整路徑

選擇 Javadoc Location



輸入 java 6 的 API 正確路徑，輸入完後可選擇 validate 以驗證是否正確

6. 連結 Hadoop Server 與 Eclipse

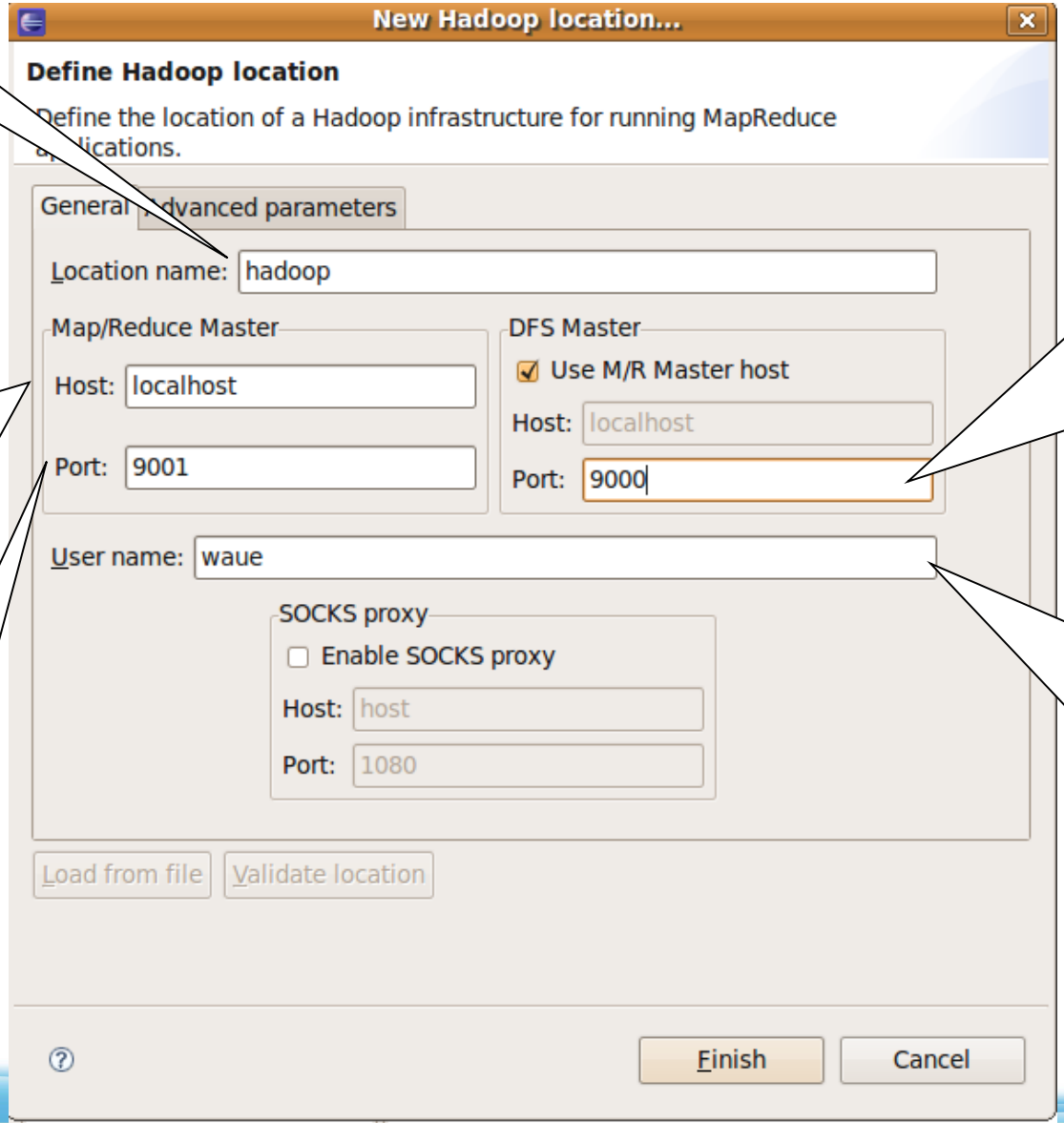


6-1．設定你要連接的 Hadoop 主機

任意填一個名稱

輸入主機位址或 domain name

MapReduce 監聽的 Port (設定於 mapred-site.xml)



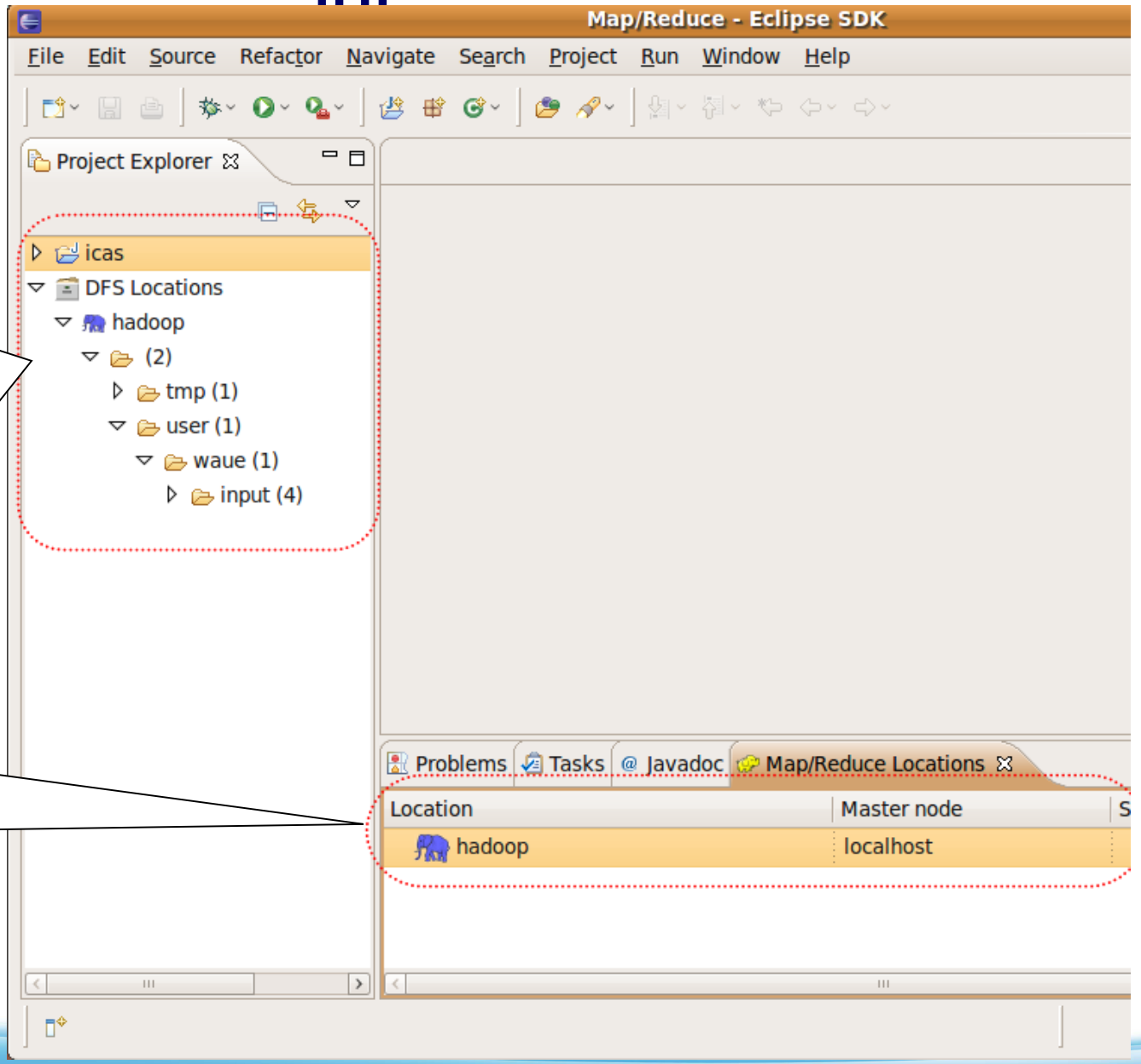
HDFS 監聽的 Port (設定於 core-site.xml)

你在此 Hadoop Server 上的 Username

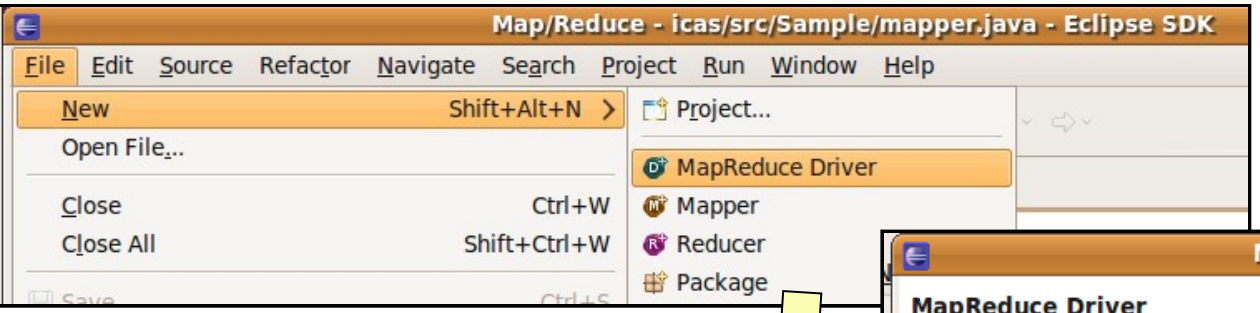
6-2 若正確設定則可得到以下畫面

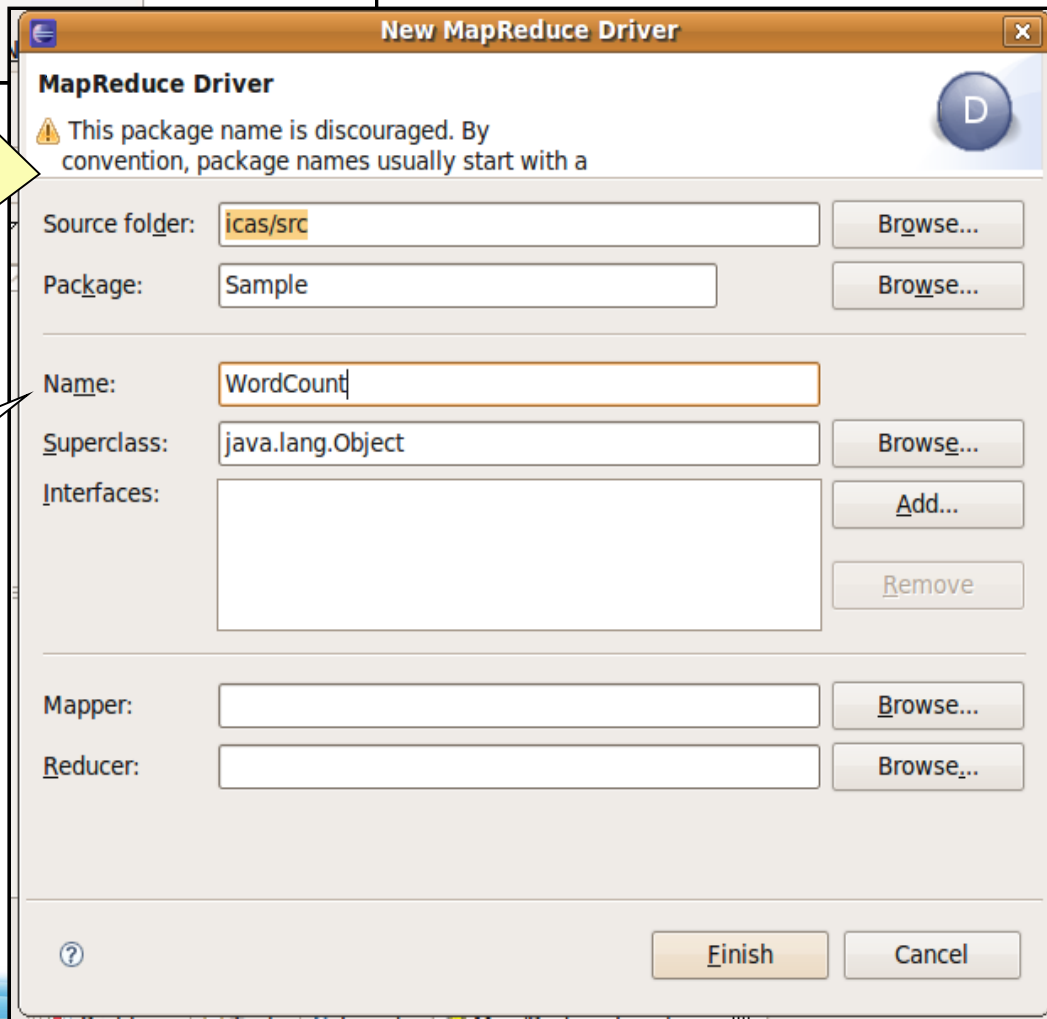
HDFS 的資訊，可直接於此操作檢視、新增、上傳、刪除等命令

若有 Job 運作，可於此視窗檢視



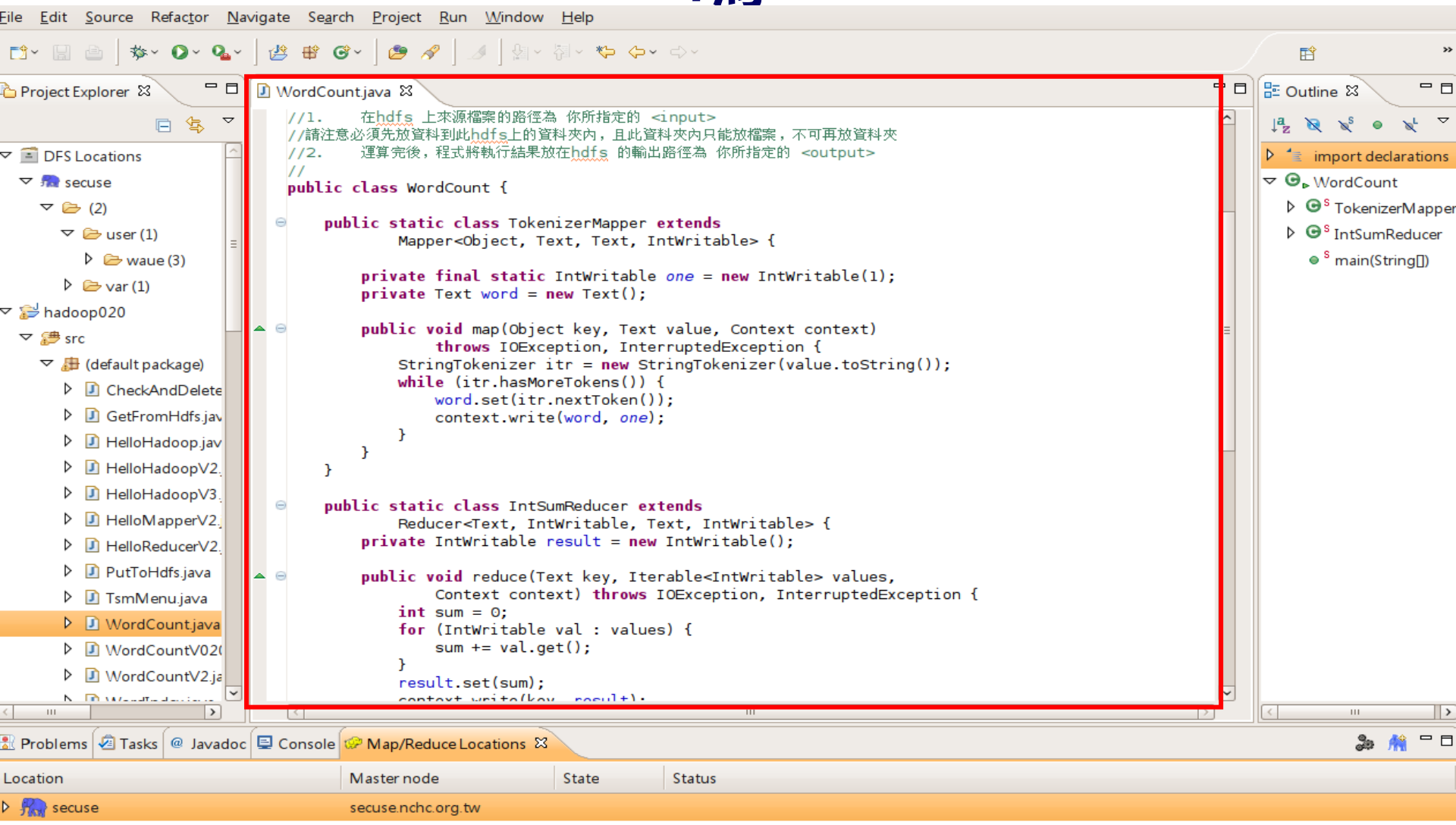
7. 新增一個 Hadoop 程式



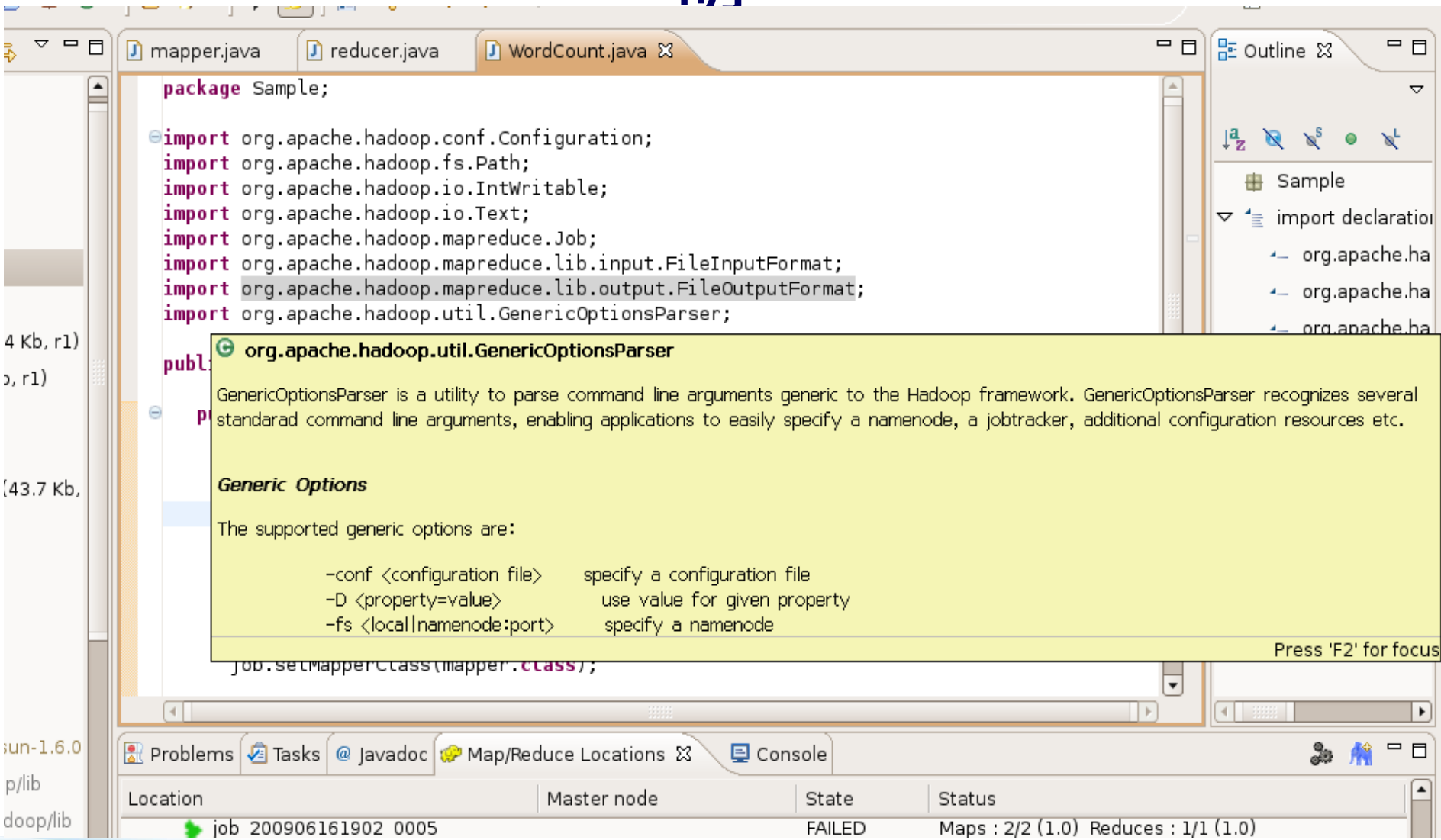


首先先建立一個 WordCount 程式，其他欄位任意

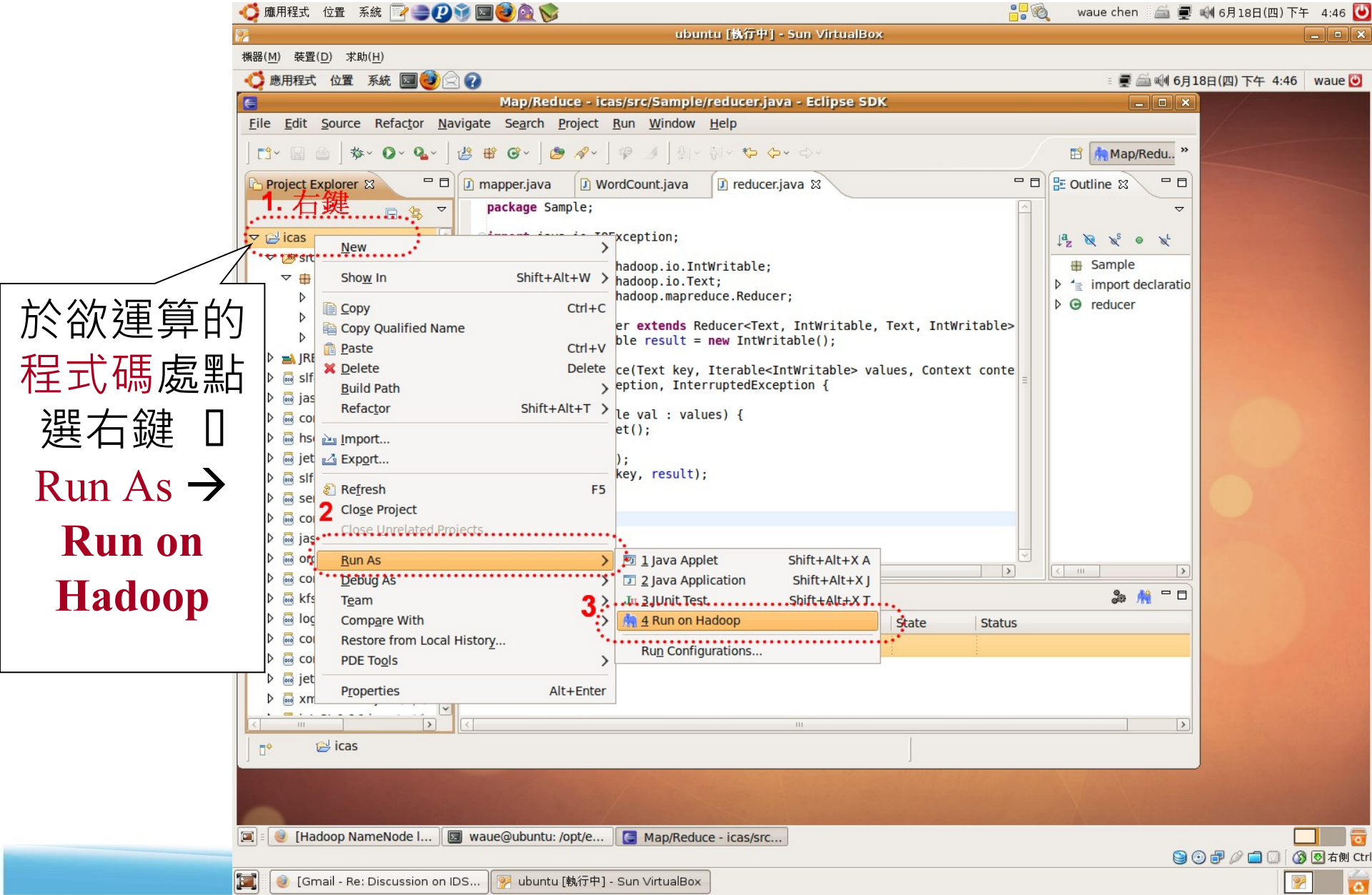
7.1 於程式窗格內輸入程式碼



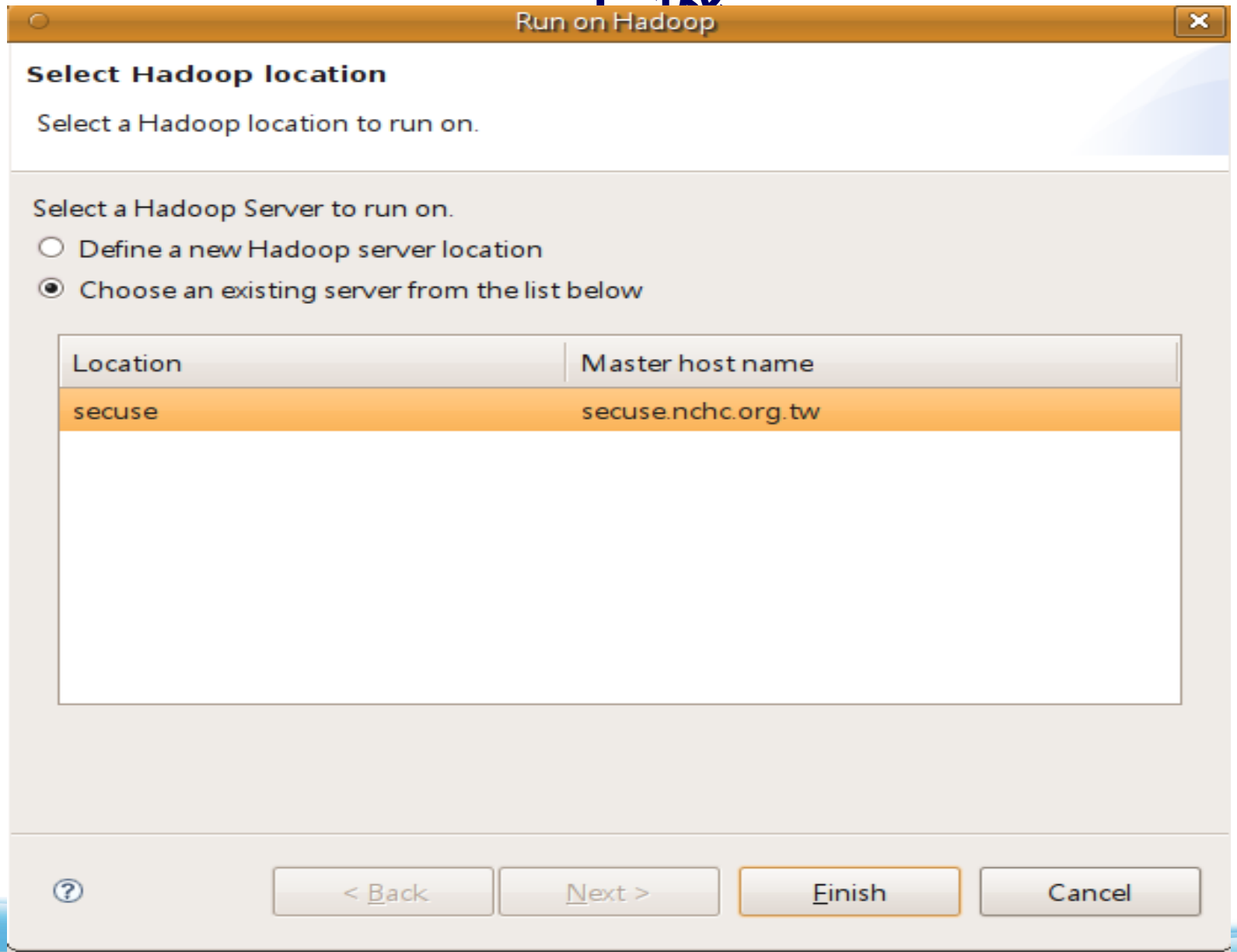
7.2 補充：若之前 doc 部份設定正確，則滑鼠移至程式碼可取得 API 完整說明



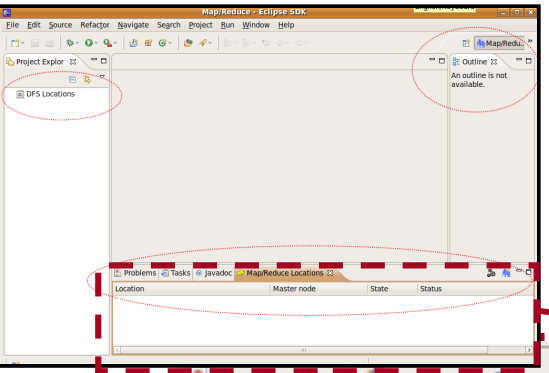
8. 運作


於欲運算的
程式碼處點
選右鍵
Run As →
Run on
Hadoop
1. 右鍵
2
3
New
Show In Shift+Alt+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Refactor Shift+Alt+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Run As
Debug As
Team
Compare With
Restore from Local History...
PDE Tools
Properties Alt+Enter
1 Java Applet Shift+Alt+X A
2 Java Application Shift+Alt+X J
3 JUnit Test Shift+Alt+X T
4 Run on Hadoop
Run Configurations...
Map/Reduce - icas/src/Sample/reducer.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
mapper.java
WordCount.java
reducer.java
Outline
Sample
import declaratio
reducer
State
Status
icas
ubuntu [執行中] - Sun VirtualBox
[Hadoop NameNode I...
waue@ubuntu: /opt/e...
Map/Reduce - icas/src...
[Gmail - Re: Discussion on IDS...
ubuntu [執行中] - Sun VirtualBox

8-1 選擇之前設定好所要運算的主機



8.2 運算資訊出現於 Eclipse 右下方的 Console 視窗

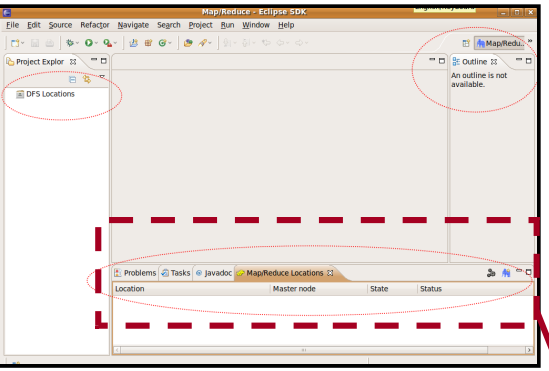


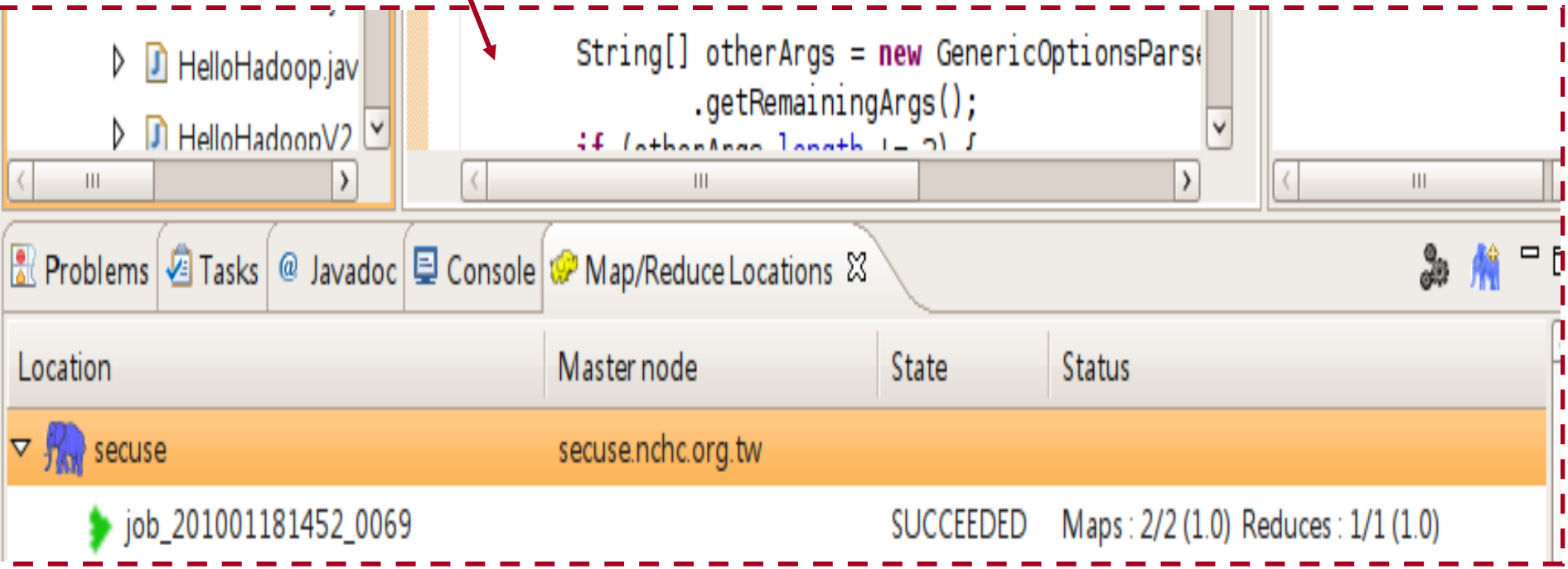
Problems | Tasks | Javadoc | Console | Map/Reduce Locations

<terminated> WordCount (1) [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.16/bin/java (2010/1/20 下午 6:15:07)

```
10/01/20 18:15:08 WARN conf.Configuration: DEPRECATED: hadoop-site.xml found in the classpath. Usage of hadc
10/01/20 18:15:08 WARN mapred.JobConf: The variable mapred.task.maxvmem is no longer used. Instead use mapre
10/01/20 18:15:08 WARN mapred.JobConf: The variable mapred.task.maxvmem is no longer used. Instead use mapre
10/01/20 18:15:08 INFO input.FileInputFormat: Total input paths to process : 2
10/01/20 18:15:08 INFO mapred.JobClient: Running job: job_201001181452_0078
10/01/20 18:15:09 INFO mapred.JobClient:  map 0% reduce 0%
10/01/20 18:15:16 INFO mapred.JobClient:  map 100% reduce 0%
10/01/20 18:15:28 INFO mapred.JobClient:  map 100% reduce 100%
10/01/20 18:15:30 INFO mapred.JobClient: Job complete: job_201001181452_0078
10/01/20 18:15:30 INFO mapred.JobClient: Counters: 17
10/01/20 18:15:30 INFO mapred.JobClient:   Job Counters
10/01/20 18:15:30 INFO mapred.JobClient:     Launched reduce tasks=1
10/01/20 18:15:30 INFO mapred.JobClient:     Launched map tasks=2
10/01/20 18:15:30 INFO mapred.JobClient:     Data-local map tasks=2
10/01/20 18:15:30 INFO mapred.JobClient:   FileSystemCounters
10/01/20 18:15:30 INFO mapred.JobClient:     FILE_BYTES_READ=153
```

8.3 剛剛運算的結果出現如下圖





Conclude

- 優點
  - 快速開發程式
  - 易於除錯
  - 智慧尋找函式庫
  - 自動鍊結 API
  - 直接操控 HDFS 與 JobTracker
  - ...
- 缺點
  - Plugin 並會因 Eclipse 版本而有不同的狀況

Map Reduce 程式架構

王耀聰　陳威宇

Jazz@nchc.org.tw

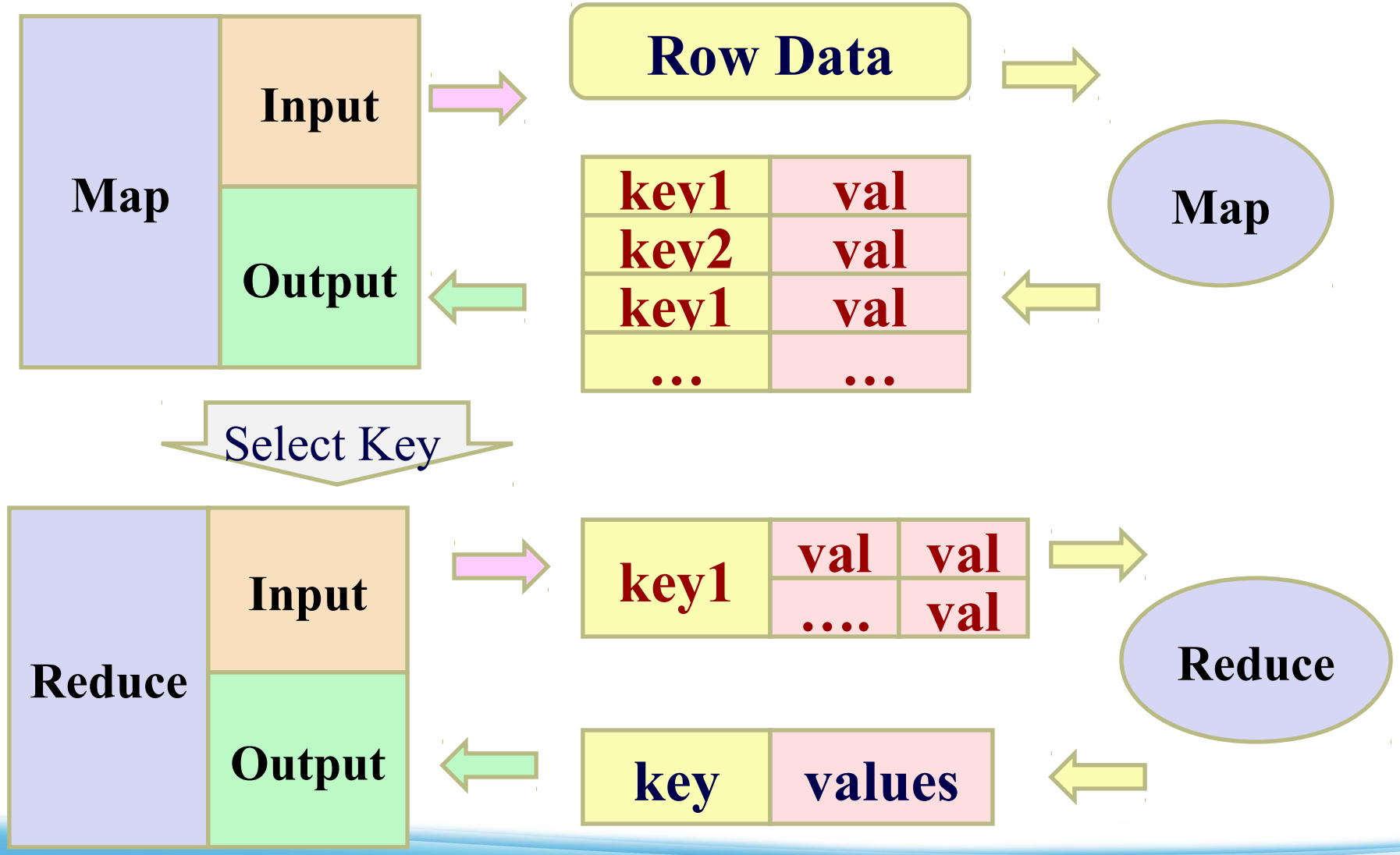
waue@nchc.org.tw

財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

<Key, Value> Pair



Program Prototype (v 0.20)

Map 區

Reduce 區

設定區

```
Class MR{
    static public Class Mapper …{
    }
     static public Class Reducer …{
    }
    main(){
        Configuration conf = new Configuration();
        Job job = new Job(conf, “job name");
        job.setJarByClass(thisMainClass.class);
        job.setMapperClass(Mapper.class);
        job.setReduceClass(Reducer.class);
        FileInputFormat.addInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        其他的設定參數程式碼
        job.waitForCompletion(true);
}}
```

Map 程式碼

Reduce 程式碼

Class Mapper (v 0.20)

```
import org.apache.hadoop.mapreduce.Mapper;
1  class MyMap extends
                Mapper < INPUT KEY Class , INPUT VALUE Class , OUTPUT KEY Class , OUTPUT VALUE Class >
2  {
3  // 全域變數區
4  public void map ( INPUT KEY Class key, INPUT VALUE Class value,
          Context context )throws IOException,InterruptedException
          {
5         // 區域變數與程式邏輯區
6         context.write( NewKey, NewValue);
7         }
8  }
9
```

Class Reducer (v 0.20)

```
import org.apache.hadoop.mapreduce.Reducer;
1  class MyRed extends
                Reducer < INPUT KEY Class , INPUT VALUE Class , OUTPUT KEY Class , OUTPUT VALUE Class >
2  {
3  // 全域變數區
4  public void reduce ( INPUT KEY Class key, Iterable< INPUT VALUE Class > values,
          Context context) throws IOException,InterruptedException

5          {
6          // 區域變數與程式邏輯區
7          context.write( NewKey, NewValue);
8          }
9  }
```

其他常用的設定參數

- 設定　Combiner
  - Job.setCombinerClass (…);
- 設定　output class
  - Job.setMapOutputKeyClass(…);
  - Job.setMapOutputValueClass(…);
  - Job.setOutputKeyClass(…);
  - Job.setOutputValueClass(…);

Class Combiner

- 指定一個 combiner ，它負責對中間過程的輸出進行聚集，這會有助於降低從 Mapper 到 Reducer 數據傳輸量。
- 可不用設定交由 Hadoop 預設
- 也可不實做此程式，引用 Reducer
- 設定
  - JobConf.setCombinerClass(Class)

範例一　(1)

```
public class HelloHadoop {


    static public class HelloMapper extends
    Mapper<LongWritable, Text, LongWritable, Text> {
    public void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException {
    context.write((LongWritable) key, (Text) value);
    }
    }  // HelloReducer end
```

..(待續) …

範例一　(2)

```
static public class HelloReducer extends
Reducer<LongWritable, Text, LongWritable, Text> {
public void reduce(LongWritable key, Iterable<Text> values,
Context context) throws IOException, InterruptedException {
Text val = new Text();
for (Text str : values) {
val.set(str.toString());
}
context.write(key, val);
}
} // HelloReducer end
```

..(待續) …

範例一　(3)

```
    public static void main(String[] args) throws IOException,
    InterruptedException, ClassNotFoundException {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "Hadoop Hello World");
    job.setJarByClass(HelloHadoop.class);
    FileInputFormat.setInputPaths(job, "input");
    FileOutputFormat.setOutputPath(job, new Path("output-hh1"));
    job.setMapperClass(HelloMapper.class);
    job.setReducerClass(HelloReducer.class);
    job.waitForCompletion(true);

    } // main end
} // wordcount class end
//  完
```

補充

Program Prototype (v 0.18)

Map 區

Reduce 區

設定區

```
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);
}}
```

Map 程式碼

Reduce 程式碼

其他的設定參數程式碼

補充

Class Mapper (v0.18)

```
import org.apache.hadoop.mapred.*;
1  class MyMap extends MapReduceBase
   implements Mapper < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
2  {
3  // 全域變數區
4  public void map ( INPUT KEY key, INPUT VALUE value,
           OutputCollector< OUTPUT KEY , OUTPUT VALUE > output,
           Reporter  reporter) throws IOException
5          {
6          // 區域變數與程式邏輯區
7          output.collect( NewKey, NewValue);
8          }
9  }
```

補充

Class Reducer (v0.18)

```
import org.apache.hadoop.mapred.*;
1  class MyRed extends MapReduceBase
   implements Reducer < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
2  {
3  // 全域變數區
4  public void reduce ( INPUT KEY key, Iterator< INPUT VALUE > values,
           OutputCollector< OUTPUT KEY , OUTPUT VALUE > output,
           Reporter  reporter) throws IOException
5          {
6          // 區域變數與程式邏輯區
7          output.collect( NewKey, NewValue);
8          }
9  }
```

程式設計一

HDFS 操作篇

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

傳送檔案至 HDFS

```
// 將檔案從 local 上傳到 hdfs , src 為 local 的來源 , dst 為 hdfs 的目的端

public class PutToHdfs {
  static boolean putToHdfs(String src, String dst, Configuration conf) {
    Path dstPath = new Path(dst);
    try {
      // 產生操作 hdfs 的物件
      FileSystem hdfs = dstPath.getFileSystem(conf);
      // 上傳
      hdfs.copyFromLocalFile(false, new Path(src),new Path(dst));
    } catch (IOException e) {
      e.printStackTrace();
      return false;
    }
    return true;
  }
```

從 HDFS 取回檔案

```
// 將檔案從 hdfs 下載回 local, src 為 hdfs 的來源, dst 為 local 的目的端
public class GetFromHdfs {
static boolean getFromHdfs(String src,String dst, Configuration conf) {
    Path dstPath = new Path(src);
    try {
      // 產生操作 hdfs 的物件
      FileSystem hdfs = dstPath.getFileSystem(conf);
      // 下載
      hdfs.copyToLocalFile(false, new Path(src),new Path(dst));
    } catch (IOException e) {
      e.printStackTrace();
      return false;
    }
    return true;
  }
```

檢查與刪除檔案

// checkAndDelete 函式，檢查是否存在該資料夾，若有則刪除之

```
public class CheckAndDelete {
  static boolean checkAndDelete(final String path, Configuration conf) {
    Path dst_path = new Path(path);
    try {
      // 產生操作 hdfs 的物件
      FileSystem hdfs = dst_path.getFileSystem(conf);
      // 檢查是否存在
      if (hdfs.exists(dst_path)) {
        // 有則刪除
        hdfs.delete(dst_path, true);
      } } catch (IOException e) {
      e.printStackTrace();
      return false;
    }     return true;    }
```

程式設計二

範例程式

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

範例二 (1)

HelloHadoopV2

說明：

此程式碼比 HelloHadoop 增加

* 檢查輸出資料夾是否存在並刪除
* input 資料夾內的資料若大於兩個，則資料不會被覆蓋
* map 與 reduce 拆開以利程式再利用

測試方法：

將此程式運作在 hadoop 0.20 平台上，執行：

---

hadoop jar HelloHadoopV2.jar

注意：

1. 在 hdfs 上來源檔案的路徑為 "/user/$YOUR_NAME/input"

請注意必須先放資料到此 hdfs 上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾

2. 運算完後，程式將執行結果放在 hdfs 的輸出路徑為 "/user/$YOUR_NAME/output-hh2"

範例二 (2)

```
public class HelloHadoopV2 {


 public static void main(String[] args) throws IOException,
    InterruptedException, ClassNotFoundException {

  Configuration conf = new Configuration();
  Job job = new Job(conf, "Hadoop Hello World 2");
  job.setJarByClass(HelloHadoopV2.class);
  // 設定  map and reduce 以及  Combiner class
  job.setMapperClass(HelloMapperV2.class);
  job.setCombinerClass(HelloReducerV2.class);
  job.setReducerClass(HelloReducerV2.class);

  // 設定 map 的輸出型態
  job.setMapOutputKeyClass(Text.class);
  job.setMapOutputValueClass(Text.class);
  // 設定 reduce 的輸出型態
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(Text.class);

  FileInputFormat.addInputPath(job, new Path("input"));

  FileOutputFormat.setOutputPath(job, new Path("output-
    hh2"));

  // 呼叫 checkAndDelete 函式，檢查是否存在該資料
    夾，若有則刪除之
  CheckAndDelete.checkAndDelete("output-hh2", conf);

  boolean status = job.waitForCompletion(true);

  if (status) {
    System.err.println("Integrate Alert Job Finished !");

  } else {
    System.err.println("Integrate Alert Job Failed !");
    System.exit(1);
  }
 }
}
```

範例二 (3)

```
public class HelloMapperV2 extends
    Mapper<LongWritable, Text, Text,
    Text> {


  public void map(LongWritable key,
    Text value, Context context)
    throws IOException,
    InterruptedException {
   context.write(new
    Text(key.toString()), value);
  }


}
```

```
public class HelloReducerV2 extends Reducer<Text,
    Text, Text, Text> {
  public void reduce(Text key, Iterable<Text> values,
    Context context)
    throws IOException, InterruptedException {

   String str = new String("");
   Text final_key = new Text();
   Text final_value = new Text();
   // 將 key 值相同的 values ，透過  && 符號分隔
    之
   for (Text tmp : values) {
    str += tmp.toString() + " &&";
   }

   final_key.set(key);
   final_value.set(str);

   context.write(final_key, final_value);
  }
}
```

範例三 (1)

HelloHadoopV2

說明：

此程式碼比 HelloHadoop 增加

* 檢查輸出資料夾是否存在並刪除
* input 資料夾內的資料若大於兩個，則資料不會被覆蓋
* map 與 reduce 拆開以利程式再利用

測試方法：

將此程式運作在 hadoop 0.20 平台上，執行：

---

hadoop jar HelloHadoopV2.jar

注意：

1. 在 hdfs 上來源檔案的路徑為 "/user/$YOUR_NAME/input"

請注意必須先放資料到此 hdfs 上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾

2. 運算完後，程式將執行結果放在 hdfs 的輸出路徑為 "/user/$YOUR_NAME/output-hh2"

範例三 (2)

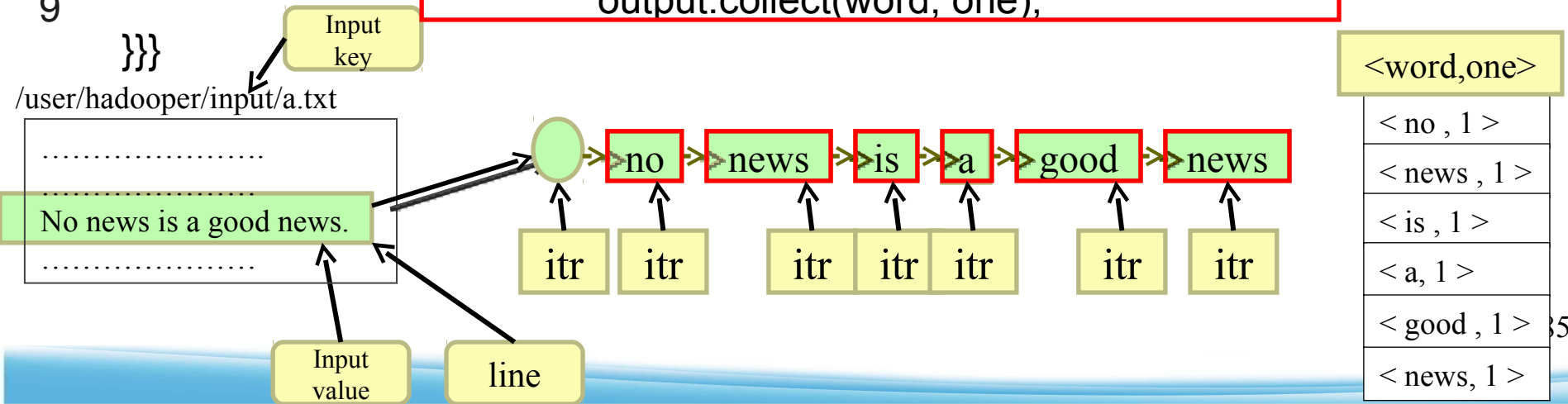
```
public class HelloHadoopV3 {
  public static void main(String[] args) throws IOException,
      InterruptedException, ClassNotFoundException {
String hdfs_input = "HH3_input";
    String hdfs_output = "HH3_output";
    Configuration conf = new Configuration();
    // 宣告取得參數
    String[] otherArgs = new GenericOptionsParser(conf, args)
        .getRemainingArgs();
    // 如果參數數量不為 2 則印出提示訊息
    if (otherArgs.length != 2) {
      System.err
          .println("Usage: hadoop jar HelloHadoopV3.jar <local_input>
        <local_output>");
      System.exit(2);
    }
    Job job = new Job(conf, "Hadoop Hello World");
    job.setJarByClass(HelloHadoopV3.class);
    // set map and reduce class
    job.setMapperClass(HelloMapperV2.class);
    job.setCombinerClass(HelloReducerV2.class);
    job.setReducerClass(HelloReducerV2.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    // 用     checkAndDelete  函式防止 overhead 的錯誤
    CheckAndDelete.checkAndDelete(hdfs_input, conf);
    CheckAndDelete.checkAndDelete(hdfs_output, conf);
  // 放檔案到 hdfs
    PutToHdfs.putToHdfs(args[0], hdfs_input, conf);
  // 設定 hdfs  的輸入輸出來源路定
    FileInputFormat.addInputPath(job, new Path(hdfs_input));
    FileOutputFormat.setOutputPath(job, new Path(hdfs_output));
    long start = System.nanoTime();
    job.waitForCompletion(true);
    // 把 hdfs 的結果取下
    GetFromHdfs.getFromHdfs(hdfs_output, args[1], conf);
    boolean status = job.waitForCompletion(true);
    // 計算時間
    if (status) {
      System.err.println("Integrate Alert Job Finished !");
      long time = System.nanoTime() - start;
      System.err.println(time * (1E-9) + " secs.");
    } else {
      System.err.println("Integrate Alert Job Failed !");
      System.exit(1);
  } } }
```

範例四　0.18 (1)

```
Class WordCount{
 main()
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("wordcount");
        // set path
        FileInputFormat.setInputPaths(new Path(args[0]));
        FileOutputFormat.setOutputPath(new Path(args[1]));
        // set map reduce
        conf.setMapperClass(MapClass.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(ReduceClass.class);
        // run
        JobClient.runJob(conf);
}}
```

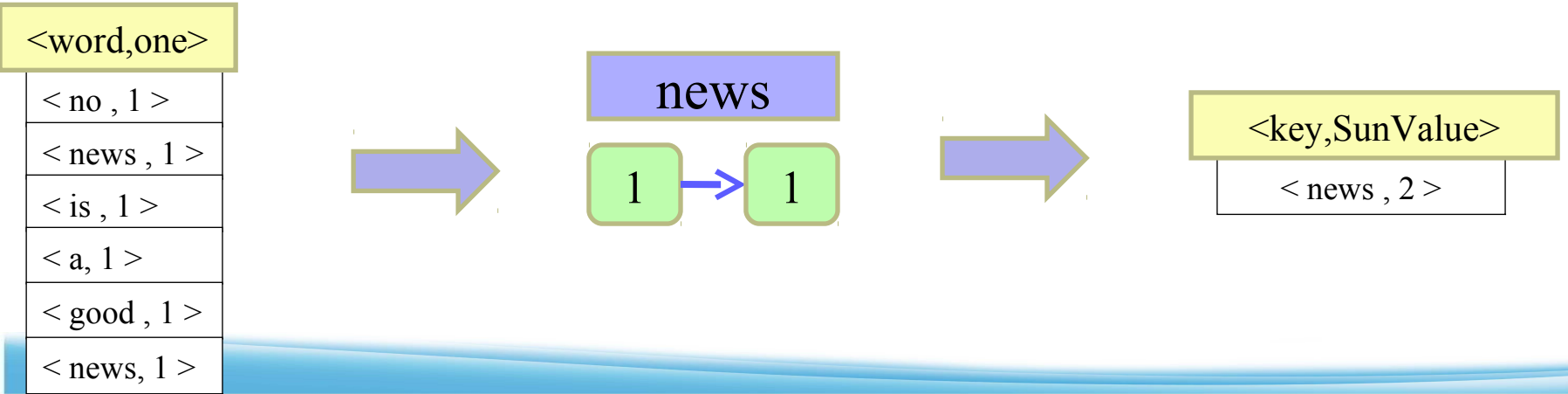
範例四　0.18 (2)

```
1   class MapClass extends MapReduceBase implements
    Mapper<LongWritable, Text, Text, IntWritable> {
2           private final static IntWritable one = new IntWritable(1);
3           private Text word = new Text();
4           public void map( LongWritable key, Text value,
              OutputCollector<Text, IntWritable> output, Reporter
    reporter) throws IOException {
5               String line = ((Text) value).toString();
6               StringTokenizer itr = new StringTokenizer(line);
7               while (itr.hasMoreTokens()) {
8                       word.set(itr.nextToken());
9                       output.collect(word, one);
    }}}
```



範例四　0.18 (3)

```
1   class ReduceClass extends MapReduceBase implements Reducer< Text,
    IntWritable, Text, IntWritable> {
2           IntWritable SumValue = new IntWritable();
3           public void reduce( Text key, Iterator<IntWritable> values,
            OutputCollector<Text, IntWritable> output, Reporter reporter)
            throws IOException {
4                   int sum = 0;
5                   while (values.hasNext())
6                           sum += values.next().get();
7                   SumValue.set(sum);
8                   output.collect(key, SumValue);
    }}
```



範例五 (1)

WordCountV2

說明：

用於字數統計，並且增加略過大小寫辨識、符號篩除等功能

測試方法：

將此程式運作在 hadoop 0.20 平台上，執行：

```
-------------------------
hadoop jar WordCountV2.jar -Dwordcount.case.sensitive=false \
  <input> <output> -skip patterns/patterns.txt
-------------------------
```

注意：

1. 在 hdfs 上來源檔案的路徑為 你所指定的 <input>
   請注意必須先放資料到此 hdfs 上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾
2. 運算完後，程式將執行結果放在 hdfs 的輸出路徑為 你所指定的 <output>
3. 請建立一個資料夾 pattern 並在裡面放置 pattern.txt ，內容如下 (一行一個，前置提示符號 \ )
   \.
   \,
   \!

範例五 (2)

```
public class WordCountV2 extends Configured implements Tool {

 public static class Map extends MapReduceBase implements
   Mapper<LongWritable, Text, Text, IntWritable> {

  static enum Counters {
   INPUT_WORDS
  }

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  private boolean caseSensitive = true;
  private Set<String> patternsToSkip = new HashSet<String>();

  private long numRecords = 0;
  private String inputFile;

  public void configure(JobConf job) {
   caseSensitive = job.getBoolean("wordcount.case.sensitive", true);
   inputFile = job.get("map.input.file");

   if (job.getBoolean("wordcount.skip.patterns", false)) {
    Path[] patternsFiles = new Path[0];
    try {
     patternsFiles = DistributedCache.getLocalCacheFiles(job);
    } catch (IOException ioe) {
     System.err
       .println("Caught exception while getting cached files: "
         + StringUtils.stringifyException(ioe));
    }
    for (Path patternsFile : patternsFiles) {
     parseSkipFile(patternsFile);
    }
   }
  }
```

```
 private void parseSkipFile(Path patternsFile) {
    try {
     BufferedReader fis = new BufferedReader(new FileReader(
       patternsFile.toString()));
     String pattern = null;
     while ((pattern = fis.readLine()) != null) {
      patternsToSkip.add(pattern);
     }
    } catch (IOException ioe) {
     System.err
       .println("Caught exception while parsing the cached file '"
         + patternsFile
         + "' : "
         + StringUtils.stringifyException(ioe));
    }   }
  public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output, Reporter reporter)
    throws IOException {
   String line = (caseSensitive) ? value.toString() : value.toString()
     .toLowerCase();

   for (String pattern : patternsToSkip) {
    line = line.replaceAll(pattern, "");
   }
   StringTokenizer tokenizer = new StringTokenizer(line);
   while (tokenizer.hasMoreTokens()) {
    word.set(tokenizer.nextToken());
    output.collect(word, one);
    reporter.incrCounter(Counters.INPUT_WORDS, 1);
   }
```

範例五 (3)

```
if ((++numRecords % 100) == 0) {
      reporter.setStatus("Finished processing " + numRecords
          + " records " + "from the input file: " + inputFile);
    }
  }
}
public static class Reduce extends MapReduceBase implements
    Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterator<IntWritable> values,
      OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
public int run(String[] args) throws Exception {

  JobConf conf = new JobConf(getConf(), WordCount.class);
  conf.setJobName("wordcount");
  String[] otherArgs = new GenericOptionsParser(conf, args)
      .getRemainingArgs();
  if (otherArgs.length < 2) {
    System.out.println("WordCountV2 [-Dwordcount.case.sensitive=<false|true>] \\
        ");
    System.out.println("           <inDir> <outDir> [-skip Pattern_file]");
    return 0;
  }
  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);
```

```
  conf.setMapperClass(Map.class);
  conf.setCombinerClass(Reduce.class);
  conf.setReducerClass(Reduce.class);
  conf.setInputFormat(TextInputFormat.class);
  conf.setOutputFormat(TextOutputFormat.class);

  List<String> other_args = new ArrayList<String>();
  for (int i = 0; i < args.length; ++i) {
    if ("-skip".equals(args[i])) {
      DistributedCache
          .addCacheFile(new Path(args[++i]).toUri(), conf);
      conf.setBoolean("wordcount.skip.patterns", true);
    } else {
      other_args.add(args[i]);
    }
  }
  FileInputFormat.setInputPaths(conf, new Path(other_args.get(0)));
  FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));
  CheckAndDelete.checkAndDelete(other_args.get(1), conf);
  JobClient.runJob(conf);
  return 0;
}
public static void main(String[] args) throws Exception {
int res = ToolRunner.run(new Configuration(), new WordCountV2(), args);
  System.exit(res);
}
}
```

範例六 (1)

WordIndex

說明：

將每個字出於哪個檔案，那一行印出來

測試方法：

將此程式運作在 hadoop 0.20 平台上，執行：

---

hadoop jar WordIndex.jar <input> <output>

注意：

1. 在 hdfs 上來源檔案的路徑為 你所指定的 <input>

請注意必須先放資料到此 hdfs 上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾

2. 運算完後，程式將執行結果放在 hdfs 的輸出路徑為 你所指定的 <output>

範例六 (2)

```
public class WordIndex {
  public static class wordindexM extends
    Mapper<LongWritable, Text, Text, Text> {
   public void map(LongWritable key, Text value, Context
      context)
      throws IOException, InterruptedException {

    FileSplit fileSplit = (FileSplit) context.getInputSplit();

    Text map_key = new Text();
    Text map_value = new Text();
    String line = value.toString();
    StringTokenizer st = new
     StringTokenizer(line.toLowerCase());
    while (st.hasMoreTokens()) {
      String word = st.nextToken();
      map_key.set(word);
      map_value.set(fileSplit.getPath().getName() + ":" +
      line);
      context.write(map_key, map_value);
    }
   }
 }
```

```
static public class wordindexR extends Reducer<Text,
    Text, Text, Text> {

 public void reduce(Text key, Iterable<Text> values,
    OutputCollector<Text, Text> output, Reporter
    reporter)
    throws IOException {
  String v = "";
  StringBuilder ret = new StringBuilder("\n");
  for (Text val : values) {
    v += val.toString().trim();
    if (v.length() > 0)
      ret.append(v + "\n");
  }

  output.collect((Text) key, new Text(ret.toString()));
 }
}
```

範例六 (2)

```
public static void main(String[] args) throws
      IOException,
     InterruptedException, ClassNotFoundException
      {
    // debug using
//    String[] argv = { "input", "output-wi" };
//    args = argv;

    Configuration conf = new Configuration();
    String[] otherArgs = new
      GenericOptionsParser(conf, args)
       .getRemainingArgs();
    if (otherArgs.length < 2) {
     System.out.println("hadoop jar WordIndex.jar
      <inDir> <outDir>");
     return;
    }
    Job job = new Job(conf, "word index");
    job.setJobName("word inverted index");
    job.setJarByClass(WordIndex.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setMapperClass(wordindexM.class);
    job.setReducerClass(wordindexR.class);
    job.setCombinerClass(wordindexR.class);
    FileInputFormat.setInputPaths(job, args[0]);
    CheckAndDelete.checkAndDelete(args[1], conf);
    FileOutputFormat.setOutputPath(job, new
      Path(args[1]));
    long start = System.nanoTime();
    job.waitForCompletion(true);
    long time = System.nanoTime() - start;
    System.err.println(time * (1E-9) + " secs.");
  }
}
```

範例七 (1)

TsmMenu

說明：

將之前的功能整合起來

測試方法：

將此程式運作在 hadoop 0.20 平台上，執行：

```
-------------------------
hadoop jar TsmMenu.jar < 功能 >
-------------------------
```

注意：

1. 此程式需與之前的所有範例一起打包成一個 jar 檔

範例七 (2)

```
public class TsmMenu {


  public static void main(String argv[]) {
    int exitCode = -1;
    ProgramDriver pgd = new ProgramDriver();
    if (argv.length < 1) {

      System.out.print("**********************************************\n"
        + " 歡迎使用  TSM 的運算功能  \n" + " 指令：  \n"
        + "  Hadoop jar TSM-example-*.jar < 功能 > \n" + " 功能：\n"
        + "  HelloHadoop:  秀出 Hadoop 的 <Key,Value> 為何  \n"
        + "  HelloHadoopV2:  秀出 Hadoop 的 <Key,Value> 進階版 \n"
        + "  HelloHadoopV3:  秀出 Hadoop 的 <Key,Value> 進化版 \n"
        + "  WordCount:  計算輸入資料夾內分別在每個檔案的字數統計  \n"
        + "  WordCountV2: WordCount 進階版  \n"
        + "  WordIndex:  索引每個字與其所有出現的所在列  \n"
        + "**********************************************\n");
    } else {
```

```
      try {
        pgd.addClass("HelloHadoop", HelloHadoop.class, "
        Hadoop hello world");
        pgd.addClass("HelloHadoopV2",
        HelloHadoopV2.class, " Hadoop hello world V2");
        pgd.addClass("HelloHadoopV3",
        HelloHadoopV3.class, " Hadoop hello world V3");
        pgd.addClass("WordCount", WordCount.class, "
        word count.");
        pgd.addClass("WordCountV2",
        WordCountV2.class, " word count V2.");
        pgd.addClass("WordIndex", WordIndex.class,
        "invert each word in line");
        pgd.driver(argv);
        // Success
        exitCode = 0;
        System.exit(exitCode);
      } catch (Throwable e) {
        e.printStackTrace();
}}}}
```

Conclusions

- 以上範例程式碼包含
  - Hadoop 的 key,value 架構
  - 操作 Hdfs 檔案系統
  - Map Reduce 運算方式
- 執行 hadoop 運算時，程式檔不用上傳至 hadoop 上，但資料需要再 HDFS 內
- 可運用範例七的程式達成連續運算
- Hadoop 0.20 與 Hadoop 0.18 有些 API 有些許差異，盡可能完全改寫

Map Reduce
專案分享

王耀聰　陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw

財團法人國家實驗研究院
國家高速網路與計算中心
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

Overview

- 系統分析
- 前處理
- 運算與二次運算
- 後置動作
- 系統輸入輸出介面