Outline

Module 1：大數據簡介

Module 2：Hadoop Ecosystem介紹

Module 3：Hadoop 平台安裝

Module 4：Hadoop 分散式檔案系統（HDFS）

Module 5：Hadoop MapReduce

Module 6：Apache Hive

Module 7：Sqoop與Flume

Module 8：Apache Spark

Module 9：Spark 平台安裝

Module 10：RDD － Resilient distributed dataset

Module 11：Scala 程式開發基礎

Module 12：Spark SQL 及 DataFrame

Module 13：Spark 機器學習函式庫(MLlib)

MapReduce的由來

- 為處理大巨量資料而來－假設單一電腦最多可同時處理2百萬筆資料，要如何處理2億筆資料？
  - ~~分一百次處理~~
  - 分散至一百台電腦處理 ✓

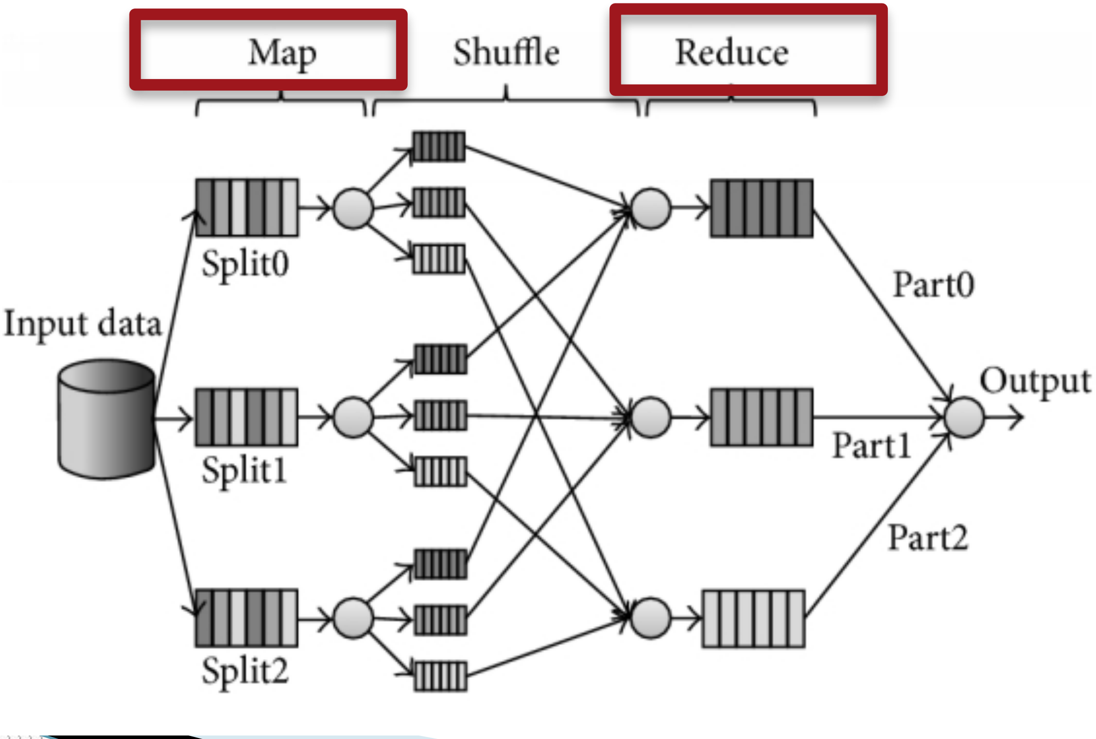
分而治之、各別擊破(Divide and Conquer)的概念

MapReduce的概念

- Google發表GFS及MapReduce相關論文(2003~2004)
- 處理巨量資料的共同課題：
  - 使用多台機器(cluster)共同處理
  - 要處理兩項基本作業：**Map**及**Reduce**
- Google主要受到**函數編程(Functional Programming)**中Map / Reduce的啟發
  - Map－將輸入**轉換為另一個集合**(方便後續處理)
  - Reduce－將Map的結果進行**聚合處理**(加總或計數等)
    - Ex：求[1,2,3,4,5]的平方和
      - Map：[1,2,3,4,5] - (^2) -> [1,4,9,16,25]
      - reduce：[1,4,9,16,25] - (sum) -> 55

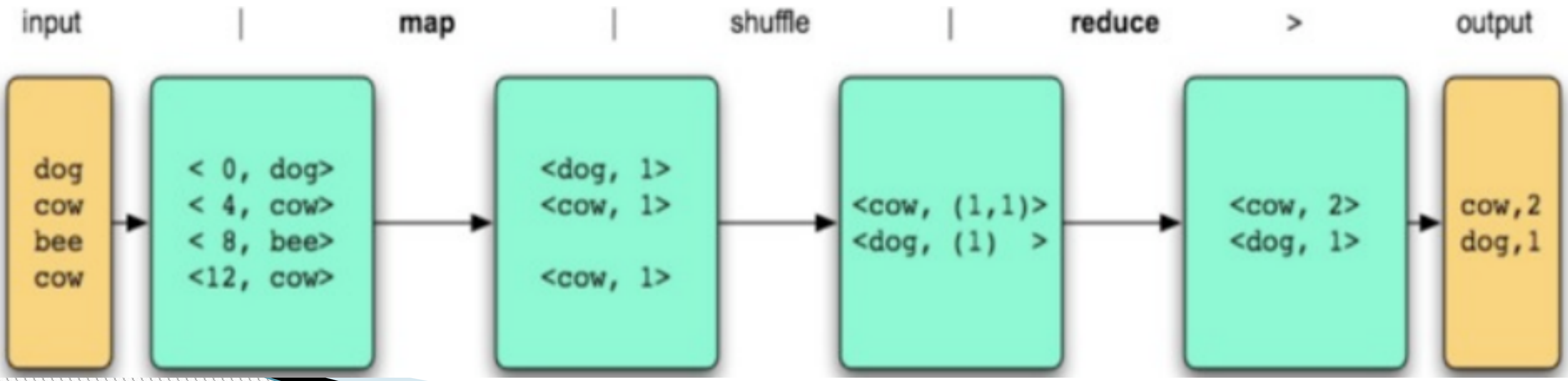
MapReduce的軟體架構

- 由Google提出，在電腦叢集上執行分散式運算的軟體架構
- 開發人員只需專注於定義Map及Reduce的執行內容
  - Map必需實作，Reduce可視情況決定是否實作
- 平行運算的其它細節(如工作及資源分配、輸入的切分、輸出結果的收集等)由MapReduce框架負責協調



MapReduce的程式模型

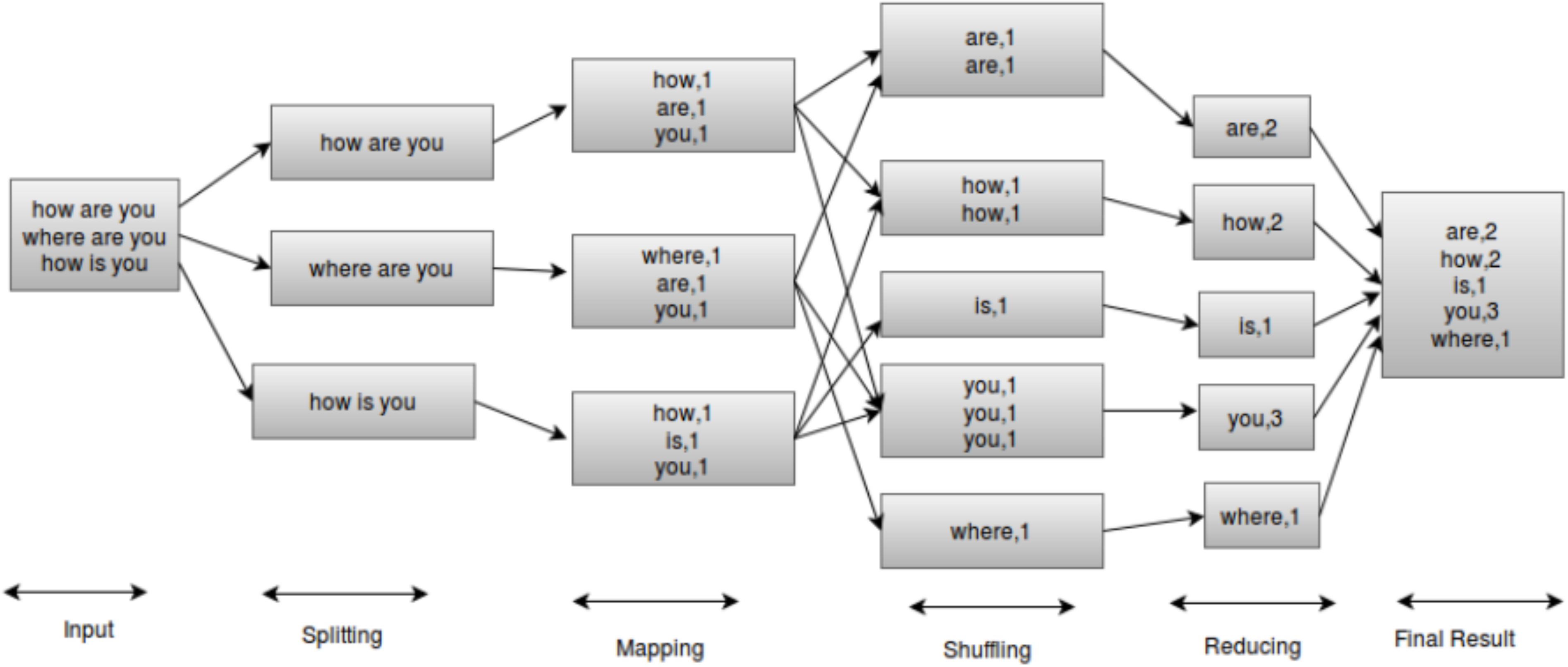
- MapReduce函式示意
  - Map （K1，V1） → list(K2，V2)
  - Reduce （K2，list(V2))→list(K3，V3)
- 邏輯流程範例



MapReduce的程式模型(以WordCount為例)

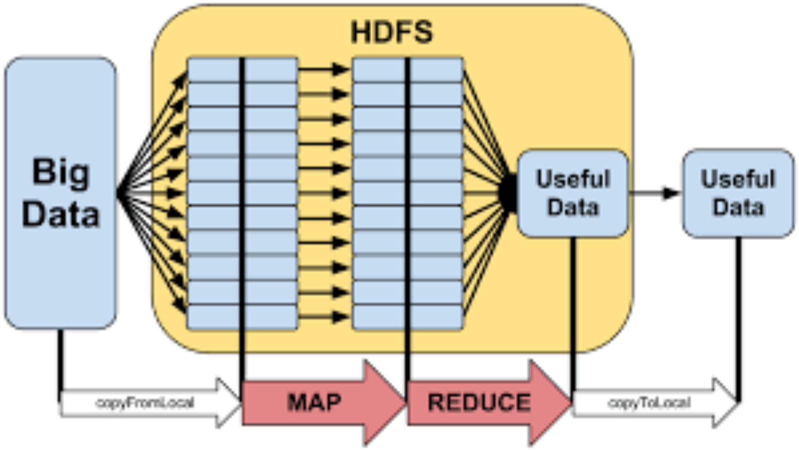
- 試著如何在大量的文件集合中，計算每個字組(Word)的出現次數？
  - Map：(offset, line) -> (split) -> [word1, word2, word1] -> (collect) -> [(word1, 1),(word2, 1), (word1, 1)]
  - Shuffle(done by framework)： [(word1, 1),(word2, 1),(word1, 1)] -> (group) -> (group) -> [(word1, [1,1]),(word2, [1])]
  - Reduce： [(word1, [1,1]),(word2, [1])] -> (sum) -> [(word1, 2),(word2, 1)]

MapReduce的程式模型(以WordCount為例)



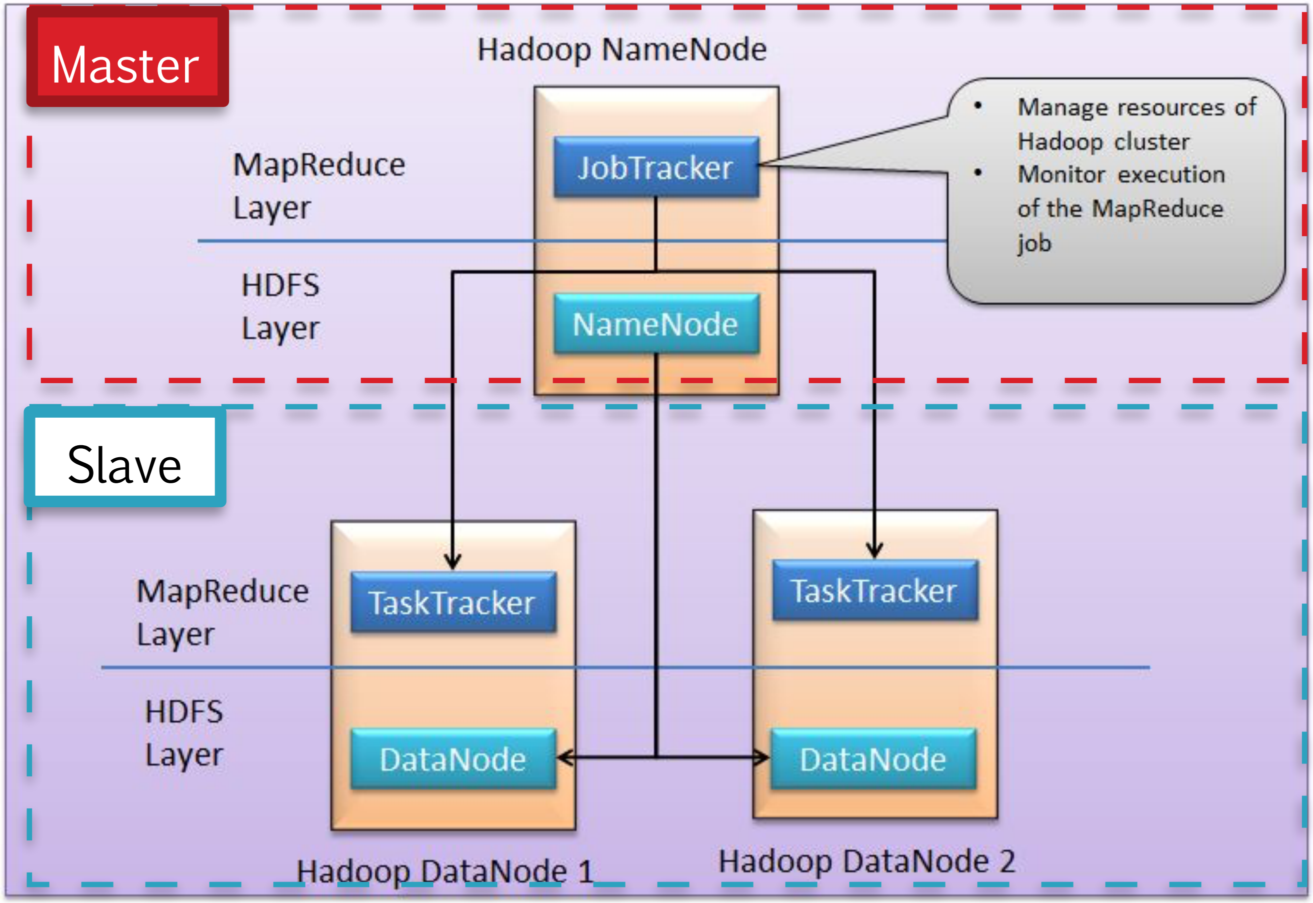
MapReduce In Hadoop

- Apache Hadoop實作Google的MapReduce
  - 提供Open Source的MapReduce框架(free for use!)
  - 以Java作為原生語言
  - 以HDFS作為輸出 / 輸入 / 中繼資料的儲存系統



Hadoop MapReduce(V1)

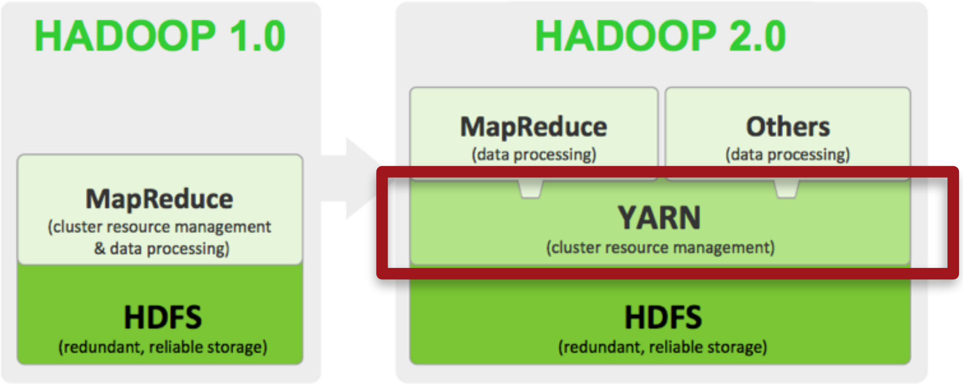
- **Master Daemon**
  - JobTracker
  - 工作調配
- **Slave Daemon**
  - TaskTracker
  - 工作執行
  - MapReduce執行

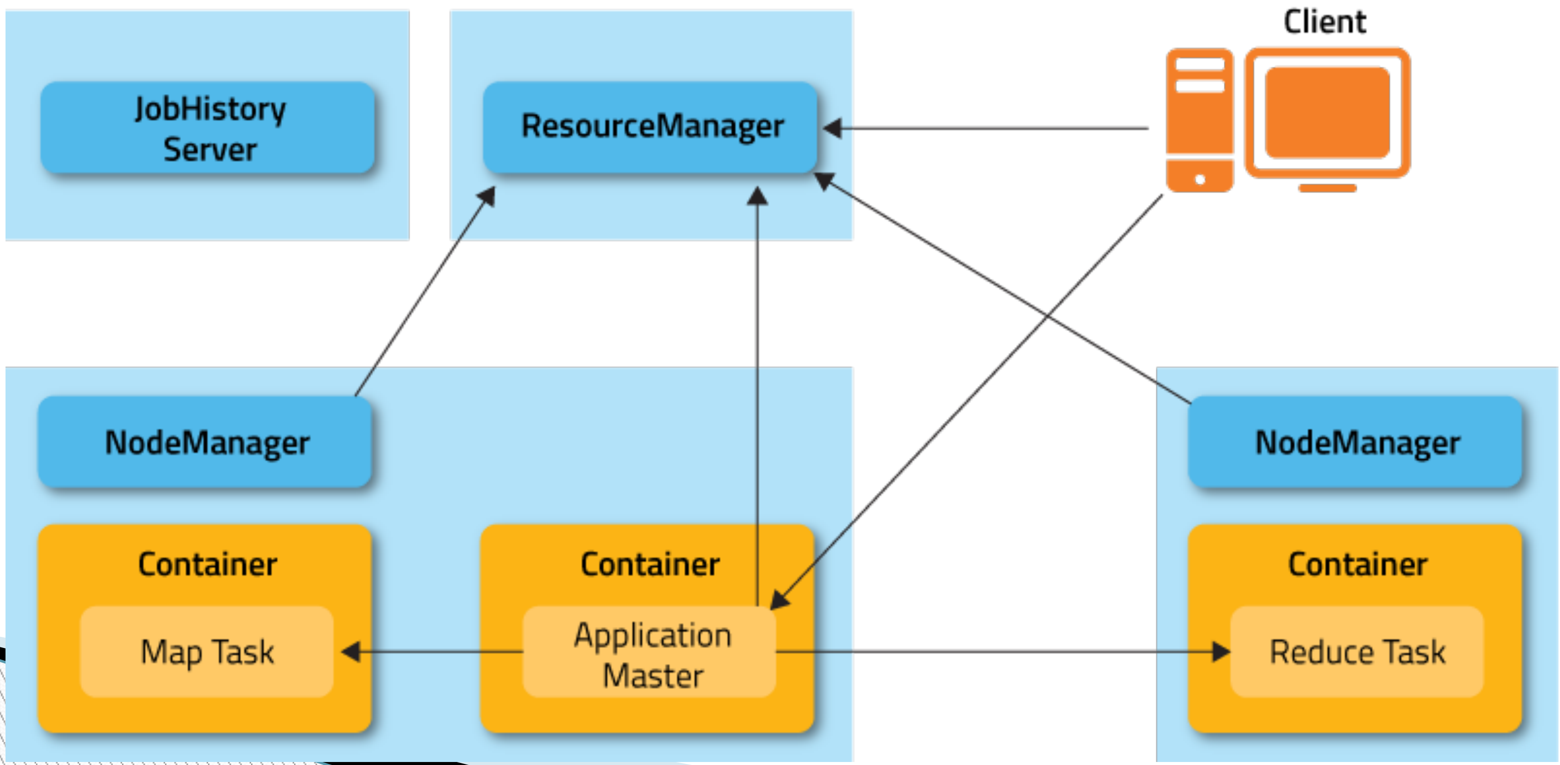


MapReduce V1 的缺點

- **延展性問題 / 效能問題**
  - JobTracker / NameNode只有一個，能夠管理的TaskTracker有限，縱使有大量的DataNode亦無用武之地
- **可及性問題**
  - NameNode有單點失敗問題
- **資源運用問題**
  - 在V1的設計中，Job的分配是考量Node的Task數進行分配，而非依據Node的CPU / Disk使用狀況，易有資源運用問題
  - V1強制配置Map Slot及Reduce Slot，若Job只有Map Task會造Reduce Slot浪費
- **與異質系統的藕合問題**
  - V1的JobTracker僅支援MapReduce的應用程式，非MapReduce的框架(如Spark)無法運作

MapReduce V2 / YARN架構


HADOOP 1.0
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(data processing)
Others
(data processing)
YARN
(cluster resource management)
HDFS
(redundant, reliable storage)



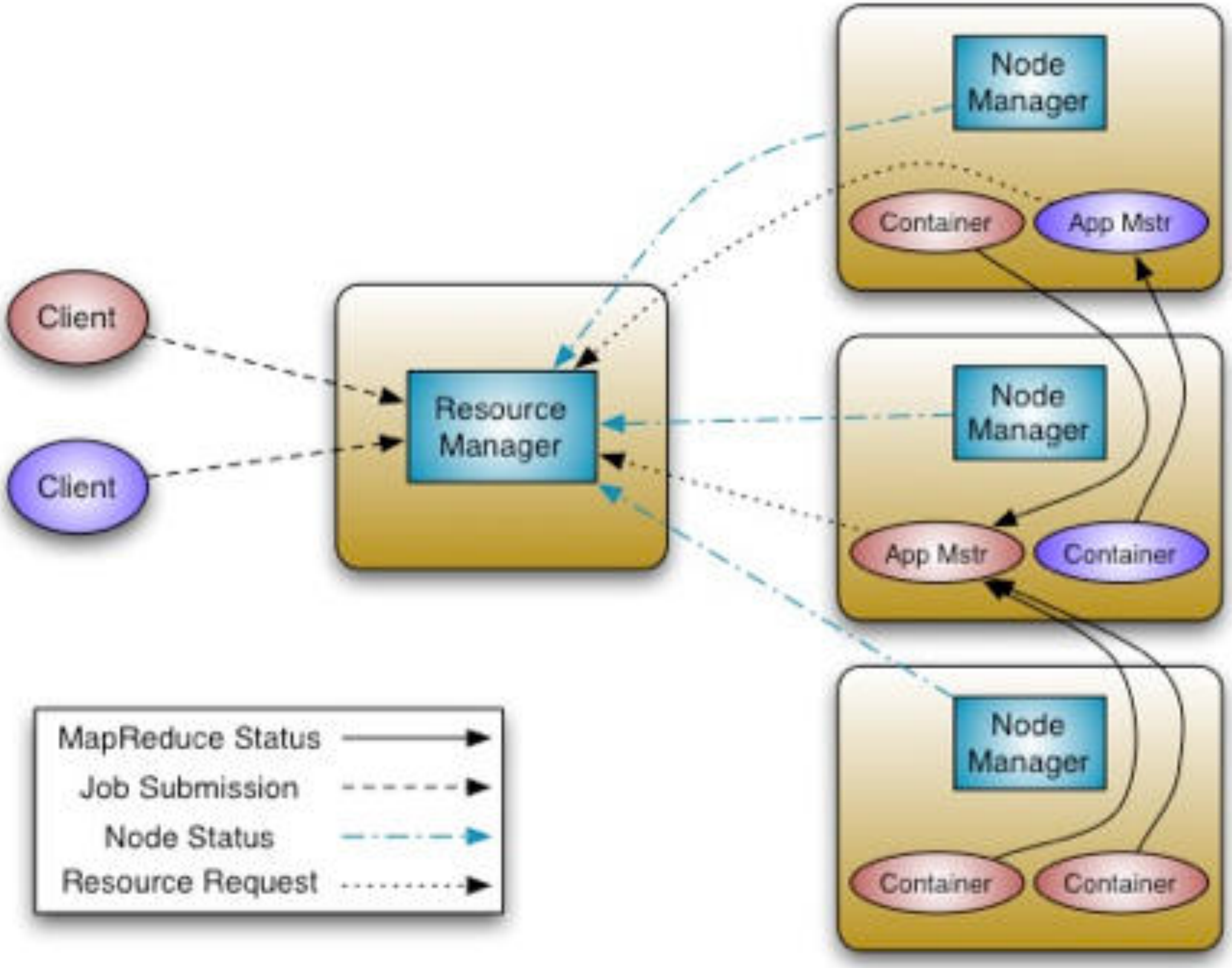
MapReduce V2 / YARN設計

- 改善V1的問題
  - 將Job Tracker的作業控制及資源管理分開
    - Resource Manager：資源管理與調度
    - Application Master：Task執行與控制
    - Node Manager：Slave上的資源與任務管理器
    - Container：依據需求所動態配置的資源
  - 透過YARN支援異質運算框架

MapReduce V2 / YARN運作

- Resource Manager
  - 資源管理與調度(依Application Master的需求，指示Node Manager建立Container及Task)
- Application Master
  - 依運算特性向Resource Manager要求資源
  - 監控Task運作狀況
- Node Manager
  - 依Resource Manager要求配置資源並建立Container及Task

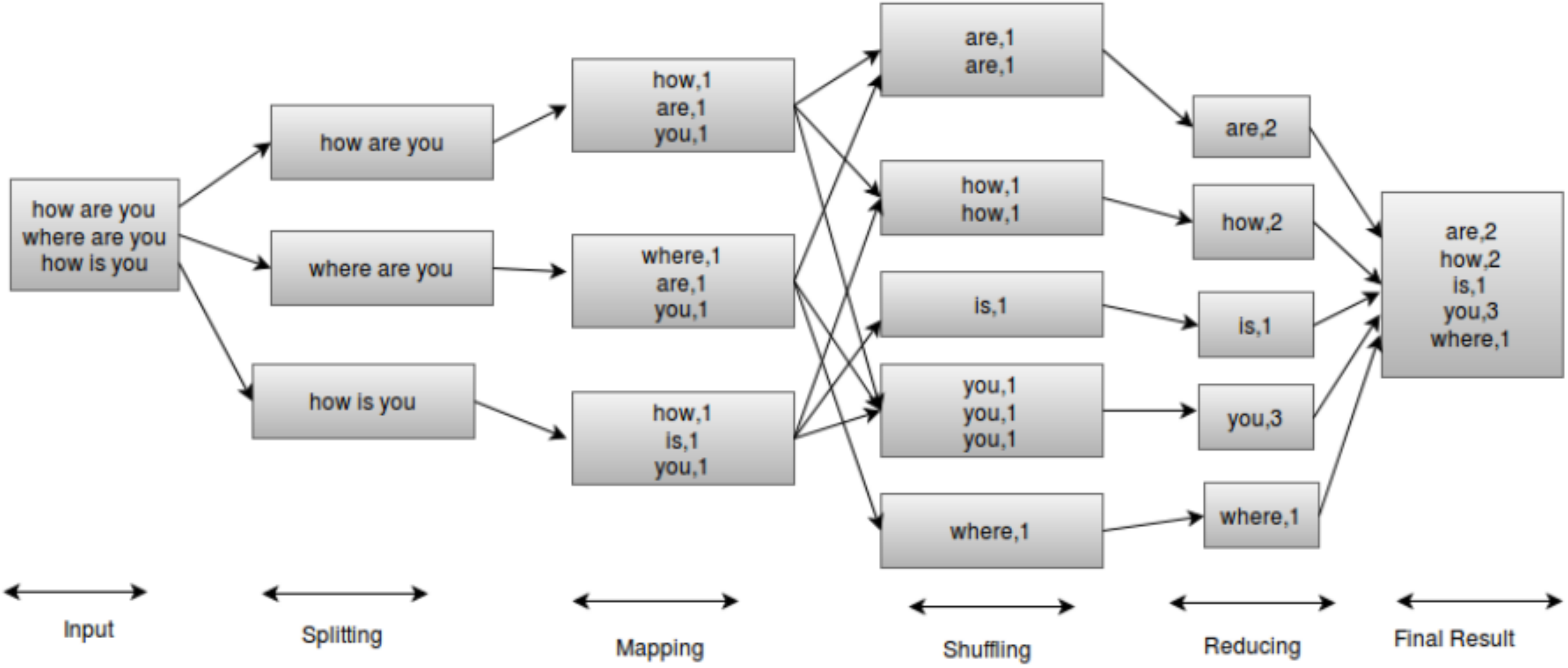
MapReduce V2 / YARN運作



Hadoop MapReduce程式開發

- 以WordCount為例
  - 開發工具 – Scala IDE (http://scala-ide.org)
  - 建立Java Project
  - 設定build Path，引入Hadoop相關library
    - Hadoop jar檔存放於＄HADOOP_HOME / share底下
    - common/hadoop-common.jar
    - common/lib/*.jar
    - hdfs/hadoop-hdfs.jar
    - mapreduce/hadoop-mapreduce-client-*.jar
    - yarn/hadoop-yarn-*.jar

WordCount執行過程示意



Hadoop MapReduce的型態

- Writable介面
  - 在org.apache.hadoop.io套件下
    - IntWritable，LongWritable，Text
    - 對應到Java原生型態IntWritable => Int
  - 主要功能
    - 資料序列化及反序列化，方便資料交換
  - 所有Mapper與Reducer都必需**使用Writable Interface作為輸入參數**

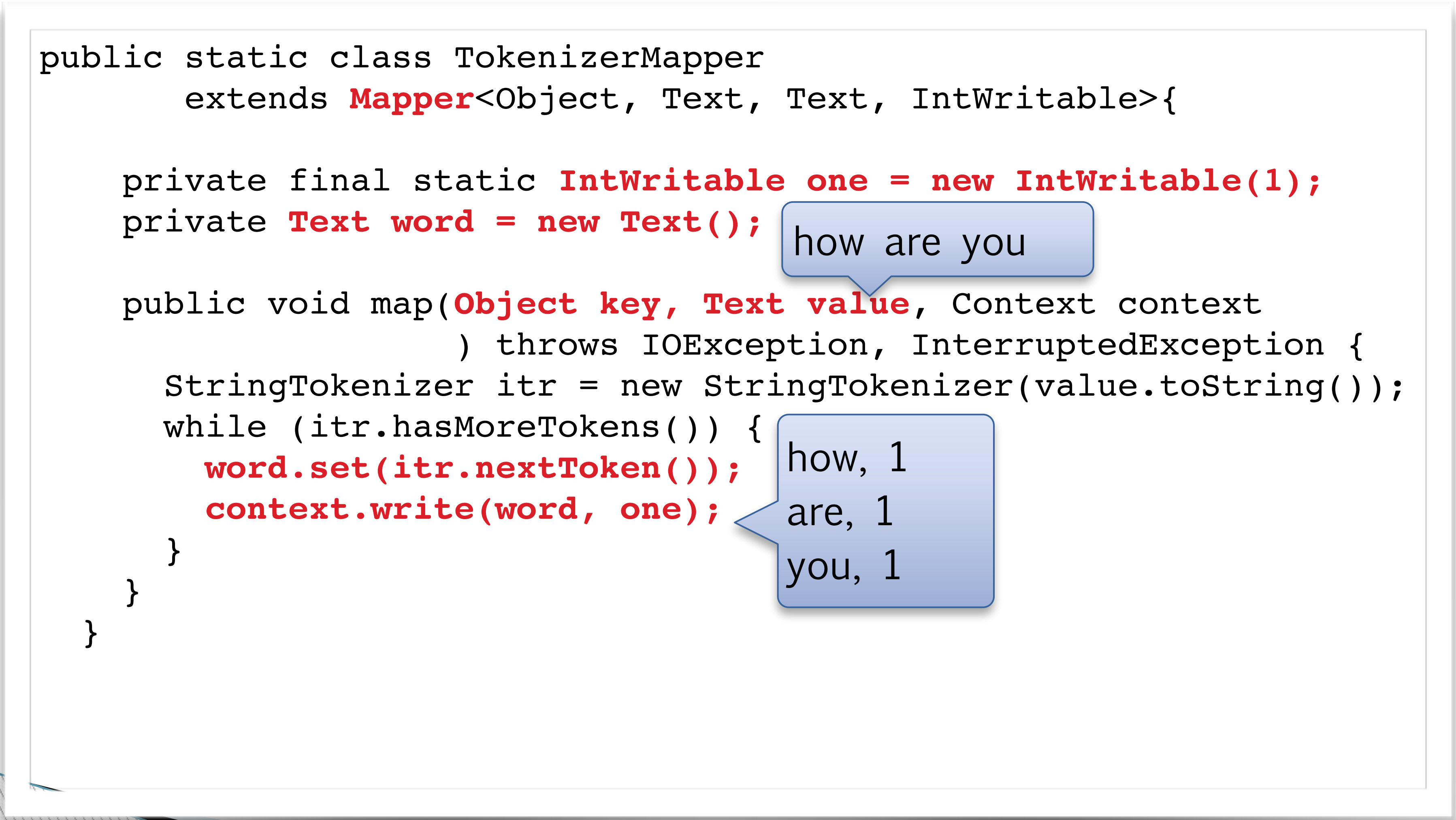
Hadoop MapReduce的型態

- Input
  - 在org.apache.hadoop.mapreduce.lib.input套件下
    - InputFormat - FileInputFormat, TextInputFormat
    - RecordReader - LineRecordReader
  - 主要功能
    - 將資料在Map階段執行分割(Splits)，確定Mask Task個數
    - 產生RecordReader，以從Splits產生一連串key / value
  - InputFormat及RecordReader**是Input及key / value的溝通橋樑**

Hadoop MapReduce的型態

- Output
  - 在org.apache.hadoop.mapreduce.lib.output套件下
    - OutputFormat - FileOutputFormat, TextOutputFormat
    - RecordWriter
  - 主要功能
    - 將產的Key / value輸出至檔案系統中

WordCount程式分析－Mapper



```
public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

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

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

how are you

how, 1
are, 1
you, 1

WordCount程式分析－Reducer

```
public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
```

對應Mapper的輸出

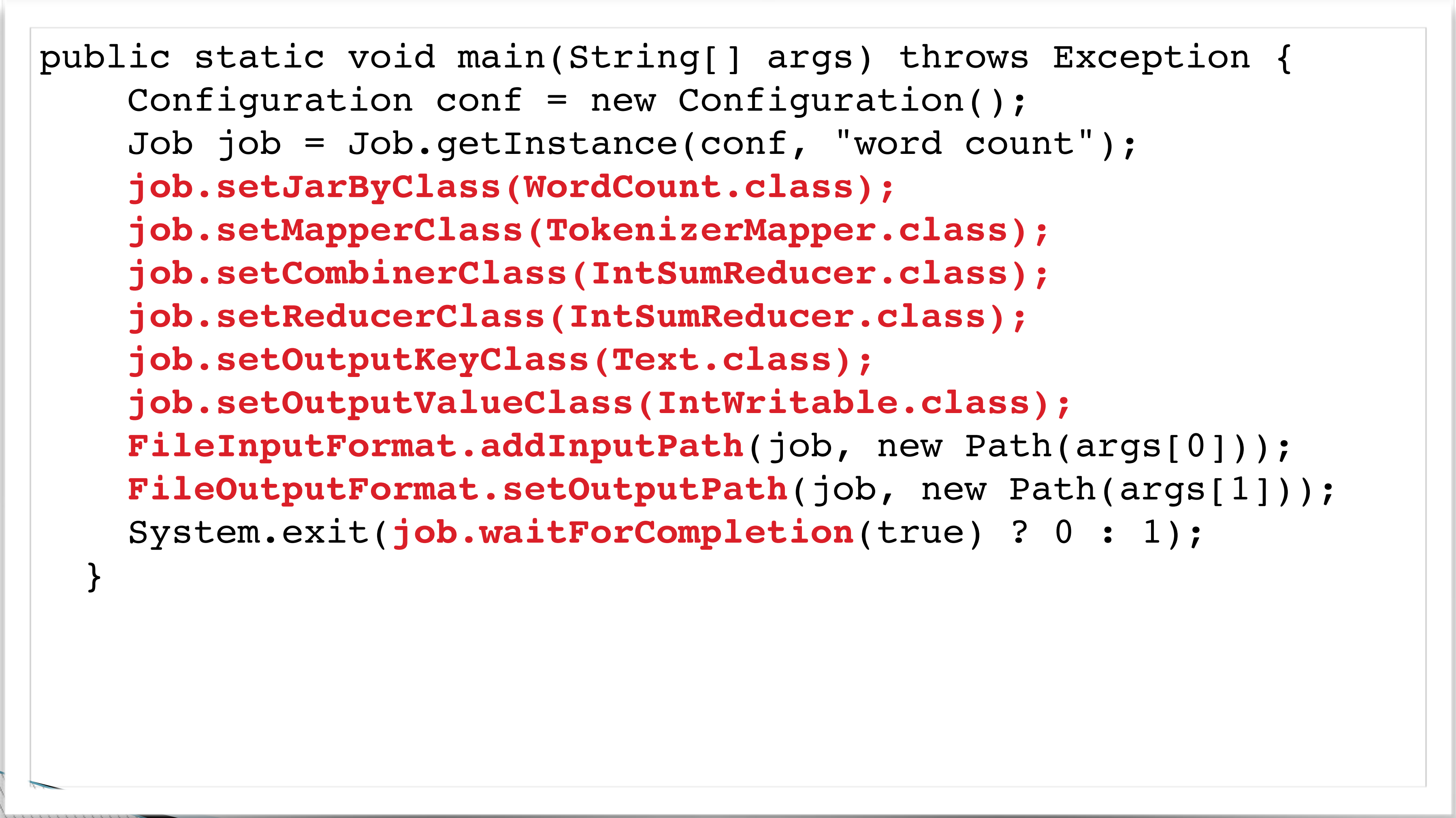
宣告Reducer的輸出

```
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException,         tedException
{
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
```

are, 1
are, 1

are, 2

WordCount程式分析－Driver



```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
```

WordCount執行－前置作業

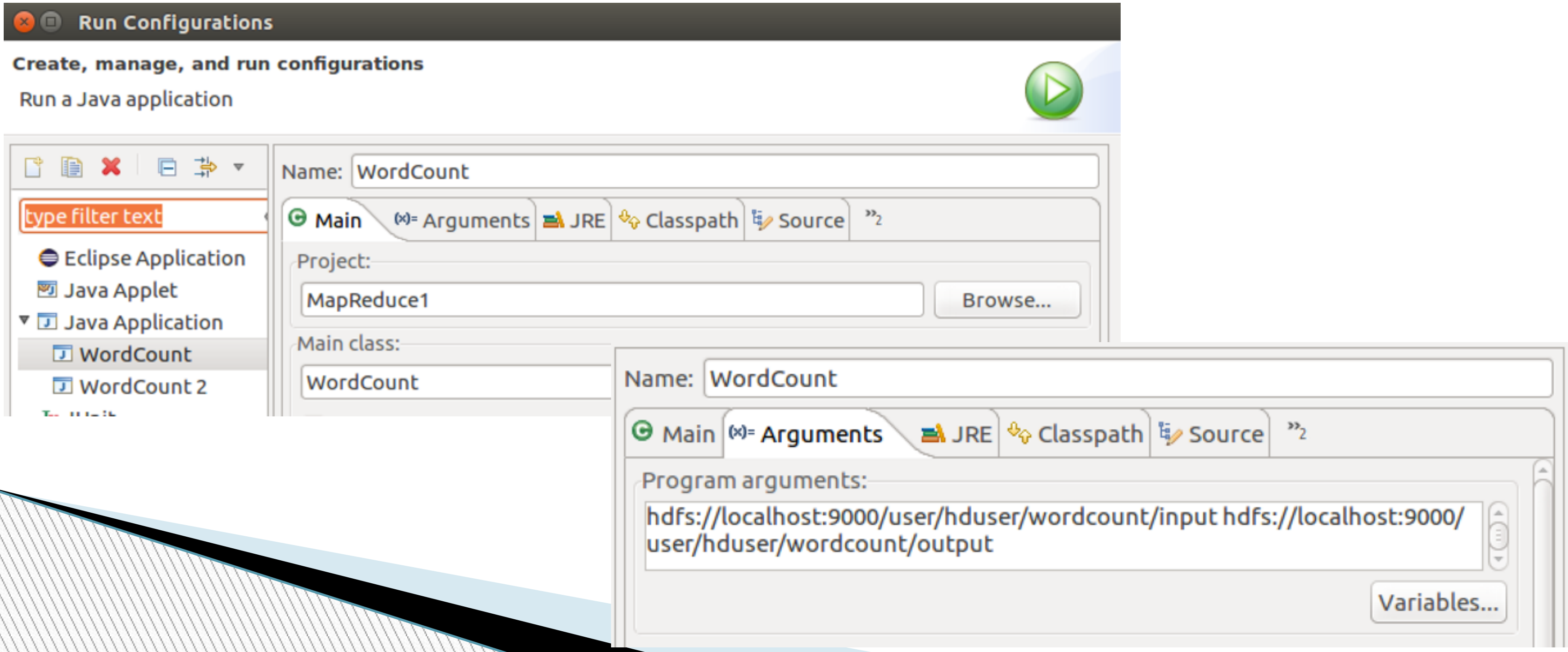
- 下載蓋茲堡宣言(https://github.com/yclee0418/hadoopTeach/blob/master/mapReduce/gettysburg.txt)
- 在HDFS建立input資料夾

  -
    ```
    hadoop fs -mkdir -p /user/hduser/wordcount/input
    ```
- 上傳檔案至HDSF

  -
    ```
    hadoop fs -copyFromLocal gettysburg.txt /user/hduser/wordcount/input
    ```

WordCount執行－IDE Debug / Run

- 設定debug / run configuration
- 設定Arguments
  - args[0] - hfs://localhost:9000/user/hduser/wordcount/input
  - args[1] - hfs://localhost:9000/user/hduser/wordcount/output



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: WordCount

Project: MapReduce1

Main class: WordCount

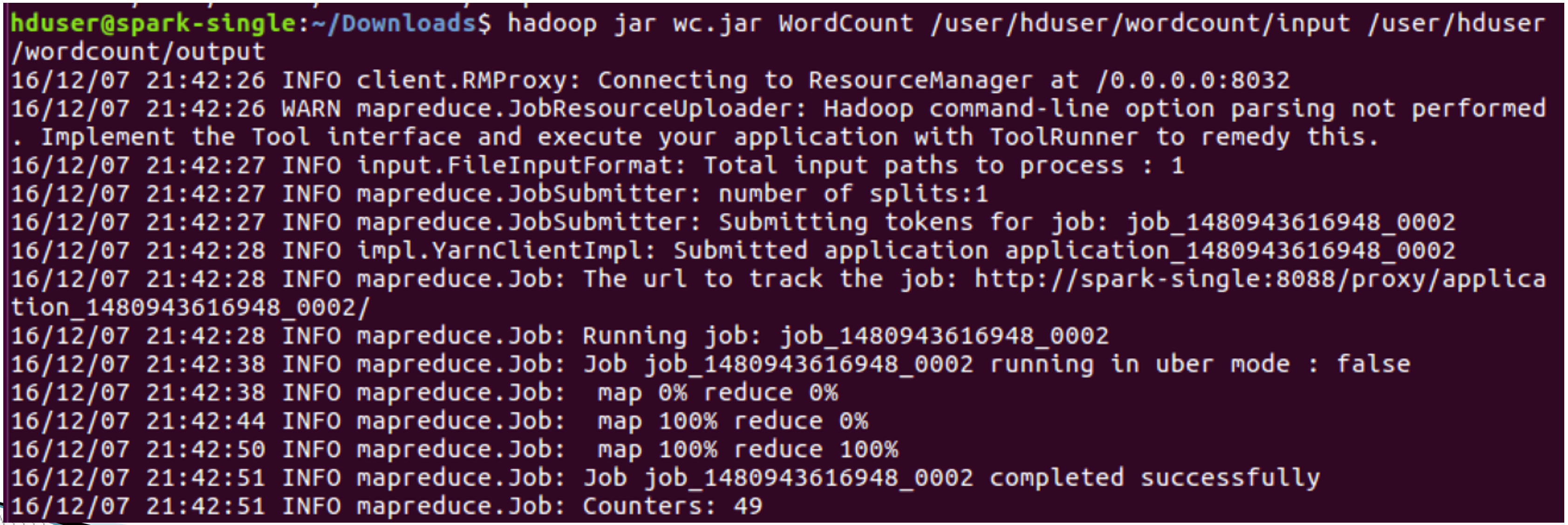
Name: WordCount

Program arguments:

```
hdfs://localhost:9000/user/hduser/wordcount/input hdfs://localhost:9000/
user/hduser/wordcount/output
```

WordCount執行－Hadoop指令

- 在IDE中將Java Project匯出成Jar檔(EX：wc.jar)
- Hadoop指令
  - hadoop jar wc.jar WordCount /user/hduser/wordcount/input /user/hduser/wordcount/output



```
hduser@spark-single:~/Downloads$ hadoop jar wc.jar WordCount /user/hduser/wordcount/input /user/hduser
/wordcount/output
16/12/07 21:42:26 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/12/07 21:42:26 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed
. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/12/07 21:42:27 INFO input.FileInputFormat: Total input paths to process : 1
16/12/07 21:42:27 INFO mapreduce.JobSubmitter: number of splits:1
16/12/07 21:42:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1480943616948_0002
16/12/07 21:42:28 INFO impl.YarnClientImpl: Submitted application application_1480943616948_0002
16/12/07 21:42:28 INFO mapreduce.Job: The url to track the job: http://spark-single:8088/proxy/applica
tion_1480943616948_0002/
16/12/07 21:42:28 INFO mapreduce.Job: Running job: job_1480943616948_0002
16/12/07 21:42:38 INFO mapreduce.Job: Job job_1480943616948_0002 running in uber mode : false
16/12/07 21:42:38 INFO mapreduce.Job:  map 0% reduce 0%
16/12/07 21:42:44 INFO mapreduce.Job:  map 100% reduce 0%
16/12/07 21:42:50 INFO mapreduce.Job:  map 100% reduce 100%
16/12/07 21:42:51 INFO mapreduce.Job: Job job_1480943616948_0002 completed successfully
16/12/07 21:42:51 INFO mapreduce.Job: Counters: 49
```