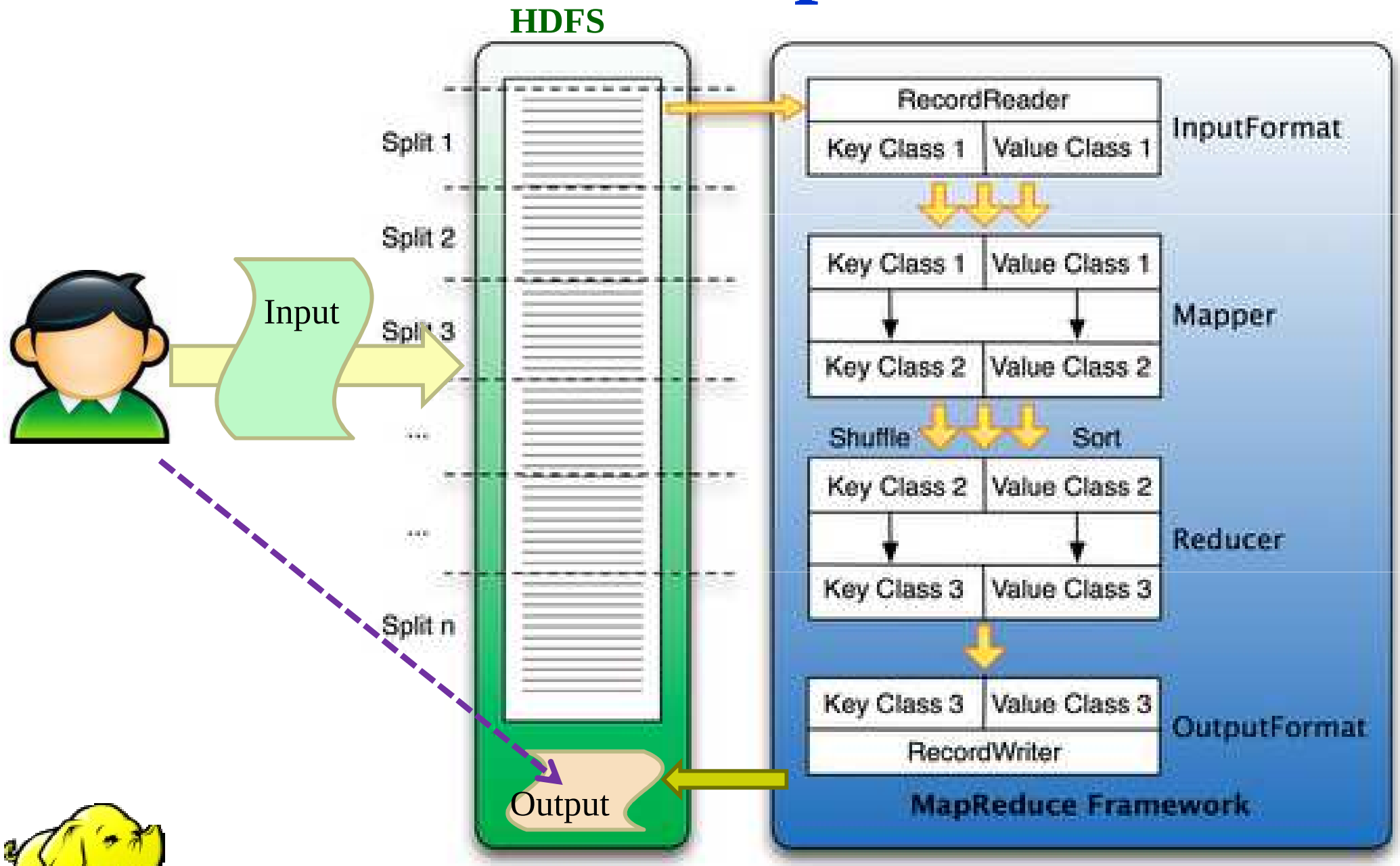
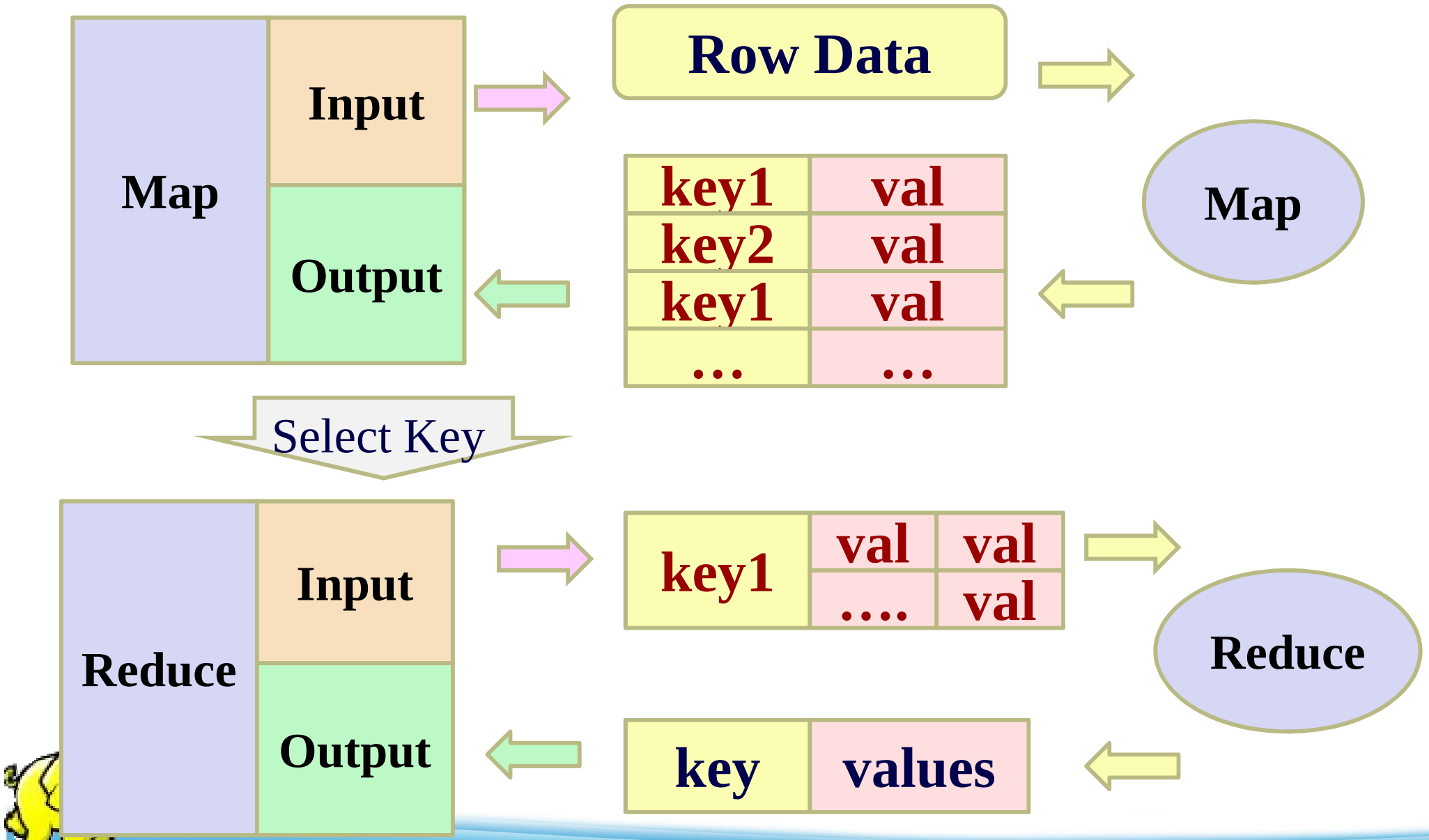
HDFS & MapReduce



部份圖片來源：http://www.larsgeorge.com/2009/05/hbase-mapreduce-101-part-i.html

<Key, Value> Pair



Program Prototype (v 0.20)

Map 區

Reduce 區

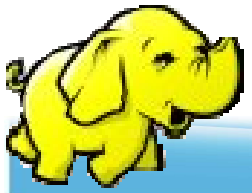
設定區

```
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]));
        job.waitForCompletion(true);
}}
```

Map 程式碼

Reduce 程式碼

其他的設定參數程式碼



Class Mapper (v 0.20)

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

Class Reducer (v 0.20)

```
import org.apache.hadoop.mapreduce.Reducer;
class MyRed extends
        Reducer < INPUT KEY Class , INPUT VALUE Class , OUTPUT KEY Class , OUTPUT VALUE Class >
{
// 全域變數區
public void reduce ( INPUT KEY Class key, Iterable< INPUT VALUE Class > values,
        Context context) throws IOException,InterruptedException
        {
        // 區域變數與程式邏輯區
        context.write( NewKey, NewValue);
        }
}
```

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)

Java 之編譯與執行

1. 編譯
   - javac △ -classpath △ hadoop-*-core.jar △ -d △ MyJava △ MyCode.java
2. 封裝
   - jar △ -cvf △ MyJar.jar △ -C △ MyJava △ .
3. 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
- My jar. 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; mkdir input_local
2. echo "I like NCHC Cloud Course." > input_local/input1
3. echo "I like nchc Cloud Course, and we enjoy this crouse." > input_local/input2
4. bin/hadoop dfs -put input_local input
5. 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

Hadoop 的MapReduce API提供

- 自動的平行化與工作分配
- 容錯特性
- 狀態監控工具
- 一個乾淨的抽象化(abstration)供程式設計師使用