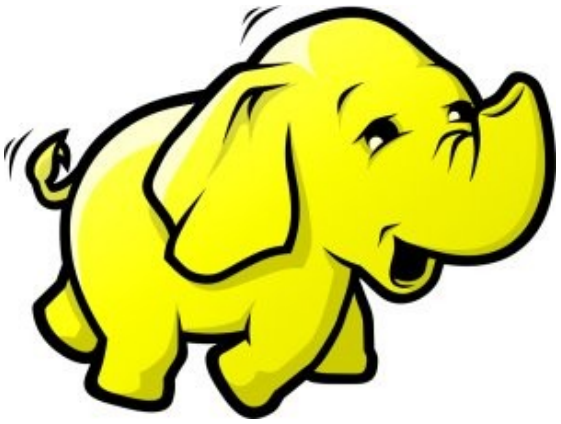
MapReduce 簡介

Introduction to MapReduce

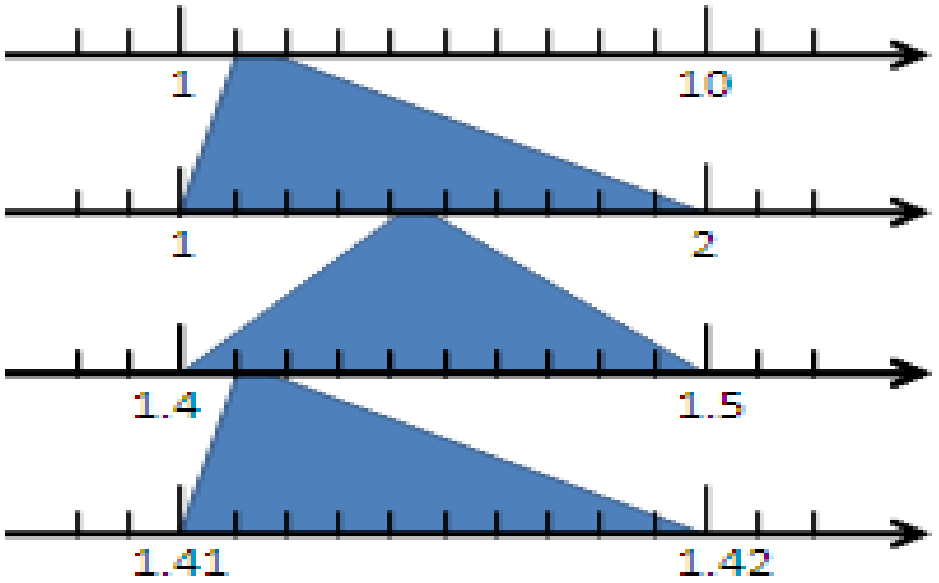
Jazz Wang
Yao-Tsung Wang
jazz@nchc.org.tw



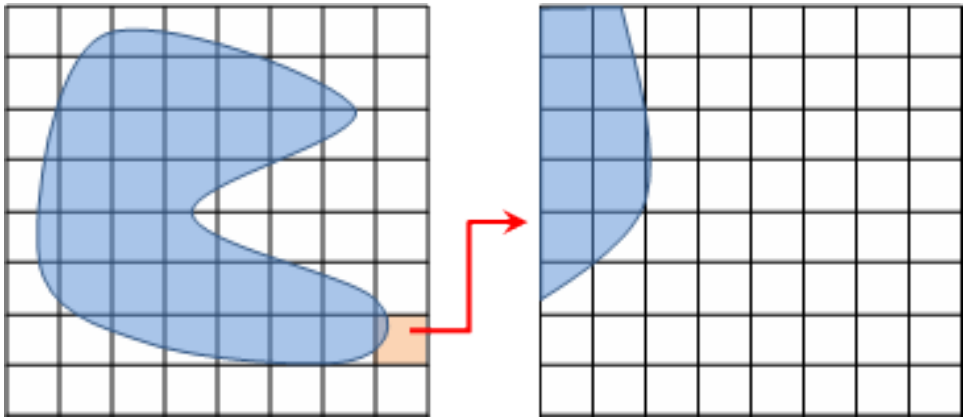
Divide and Conquer Algorithms

分而治之演算法

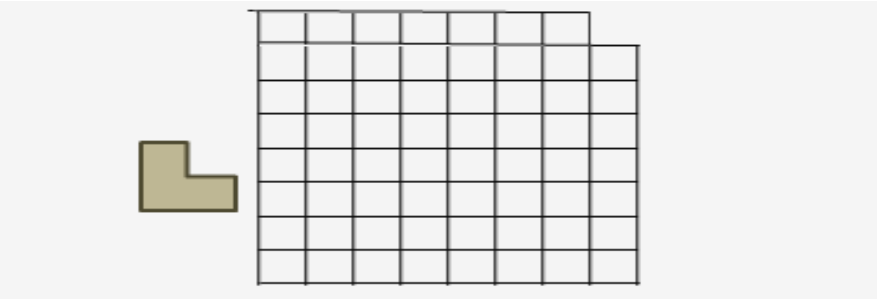
Example 1:



Example 2:



Example 3:



Example 4: The way to climb 5 steps stair within 2 steps each time. 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)

What is MapReduce ??

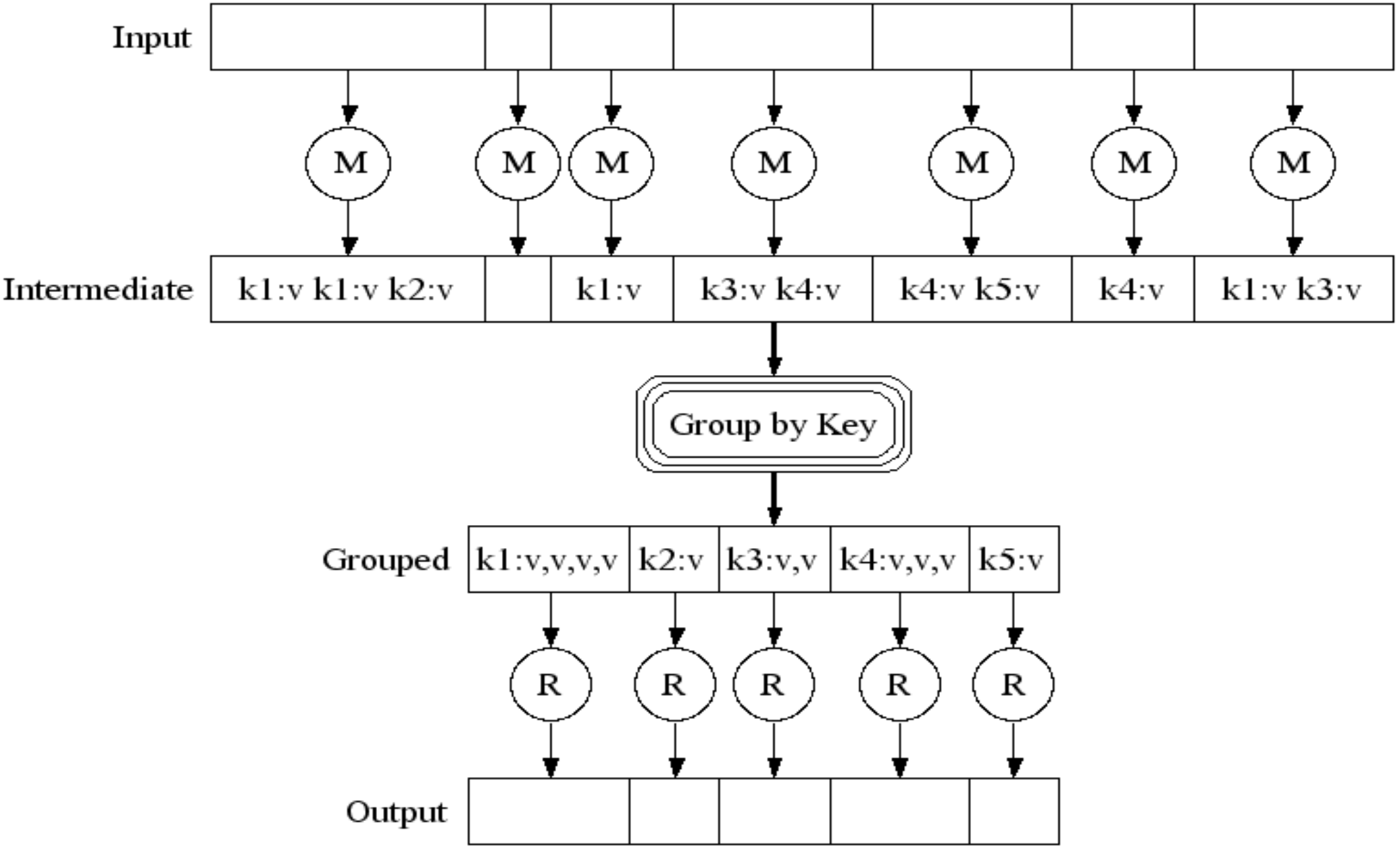
什麼是 MapReduce ??

- MapReduce 是 Google 申請的軟體專利，主要用來處理大量資料
- MapReduce is a patented software framework introduced by Google to support distributed computing on large data sets on clusters of computers.
- 啟發自函數編程中常用的 map 與 reduce 函數。
- The framework is inspired by map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as their original forms
  - Map(...) : N → N
    - Ex. [1,2,3,4] – (*2) -> [2,4,6,8]
  - Reduce(...): N → 1
    - [1,2,3,4] - (sum) -> 10
- Logical view of MapReduce
    - Map(k1,v1) -> list(k2,v2)
    - Reduce(k2, list (v2)) -> list(v3)

Source: http://en.wikipedia.org/wiki/MapReduce

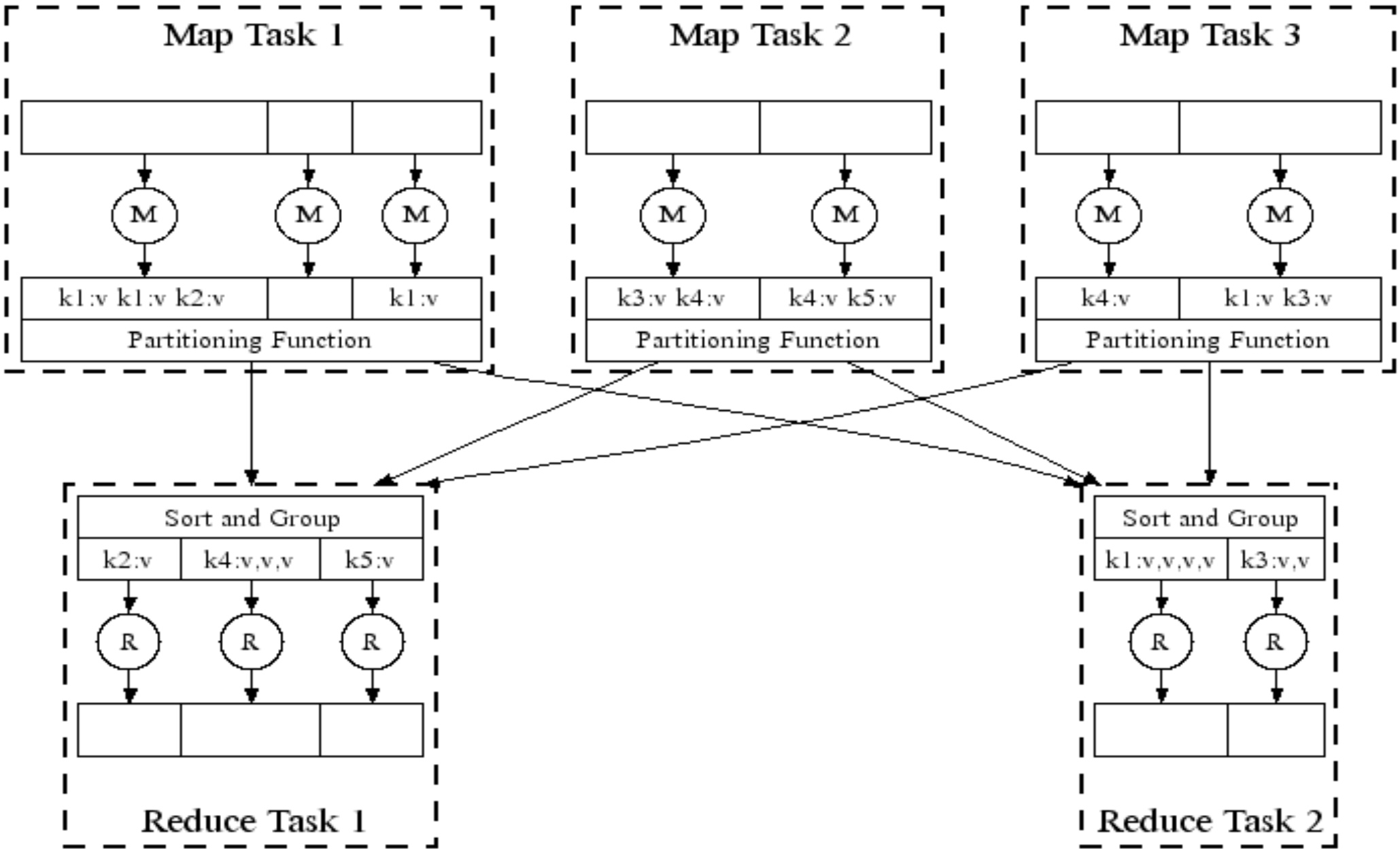
Google's MapReduce Diagram

Google 的 MapReduce 圖解



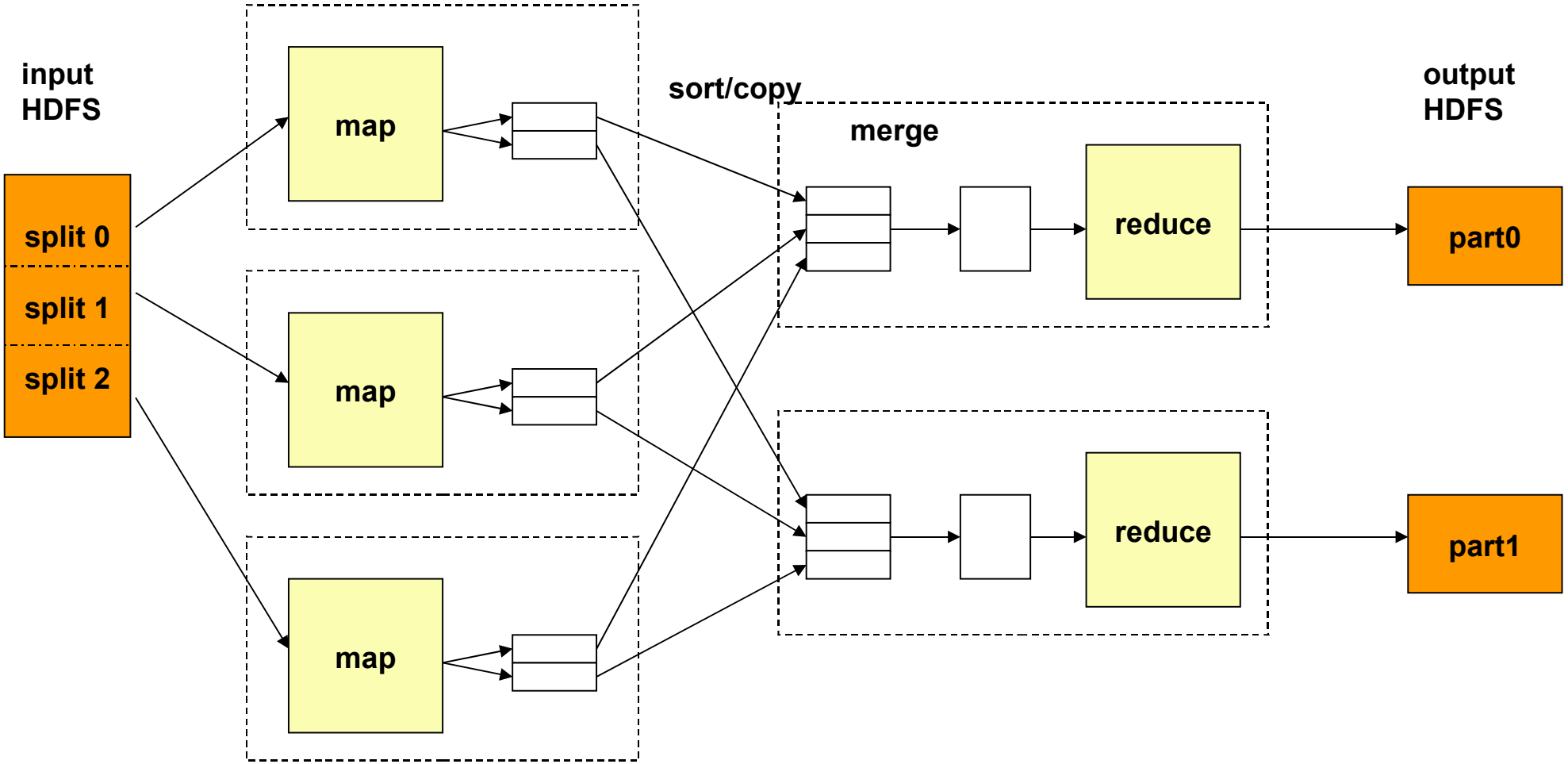
Google's MapReduce in Parallel

Google 的 MapReduce 平行版圖解



How does MapReduce work in Hadoop

Hadoop 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

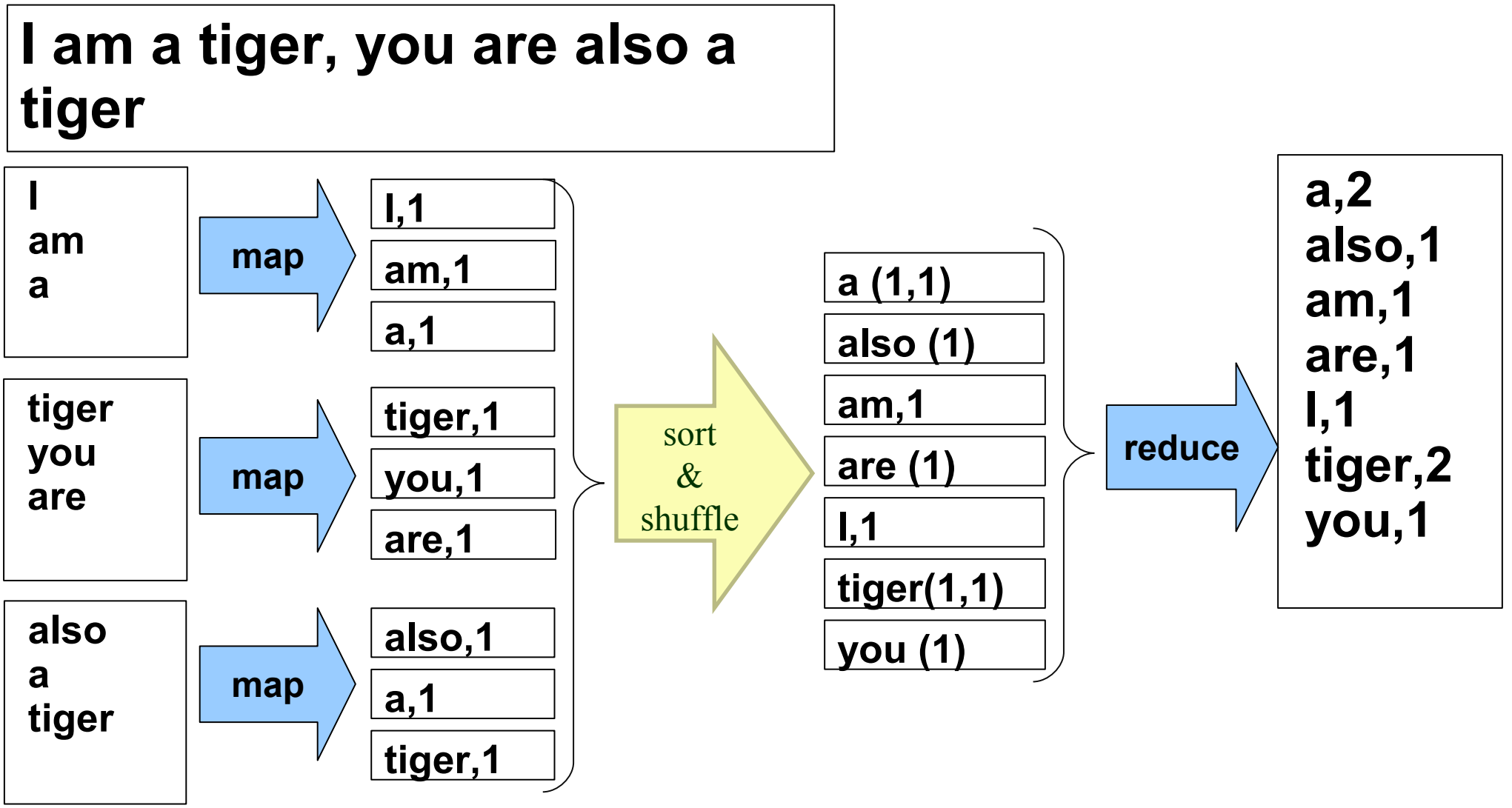
JobTracker 將中間檔案整合排序後，複製到需要的 TaskTracker 去

JobTracker 派遣 TaskTracker 作 reduce

reduce 完後通知 JobTracker 與 Namenode 以產生 output

MapReduce by Example (1)

MapReduce 運作實例 (1)



JobTracker 先選了三個 Tracker 做 map

Map 結束後，hadoop 進行中間資料的重組與排序

JobTracker 再選一個 TaskTracker 作 reduce

MapReduce by Example (2)

MapReduce 運作實例 (2)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow \begin{bmatrix} \text{sqrt}(a+b) \\ \text{sqrt}(c+d) \end{bmatrix}$$

$$\begin{bmatrix} 1.0 & 0.0 & 3.0 \\ 3.2 & 0.8 & 32.0 \\ 1.0 & 14.0 & 1.0 \end{bmatrix} \Rightarrow ?$$

```
(0,sqrt(1.0 +  0.0 +  3.0))
(1,sqrt(3.2 +  0.8 + 32.0))
(2,sqrt(1.0 + 14.0 +  1.0))
```

Input File

```
0 0 1.0   // A[0][1] = 1.0
0 1 0.0   // A[0][1] = 0.0
0 2 3.0   // A[0][2] = 3.0
1 0 3.2   // A[1][0] = 3.2
1 1 0.8   // A[1][1] = 0.8
```

```
1 2 32.0 // A[1][2] = 32.0
2 0 1.0   // A[2][0] = 1.0
2 1 14.0 // A[2][1] = 14.0
2 2 1.0   // A[2][2] = 1.0
```

map

```
(0,1.0)
(0,0.0)
(0,3.0)
(1,3.2)
(1,0.8)
```

map

```
(1,32.0)
(2,1.0)
(2,14.0)
(2,1.0)
```

sort / merge

```
(0,{1.0,0.0,3.0})
(1,{3.2,0.8,32.0})
(2,{1.0,14.0,1.0})
```

reduce

MapReduce is suitable to ....
MapReduce 合適用於 ....

- 大規模資料集
- **Large Data Set**
- 可拆解
- **Parallelization**

- 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

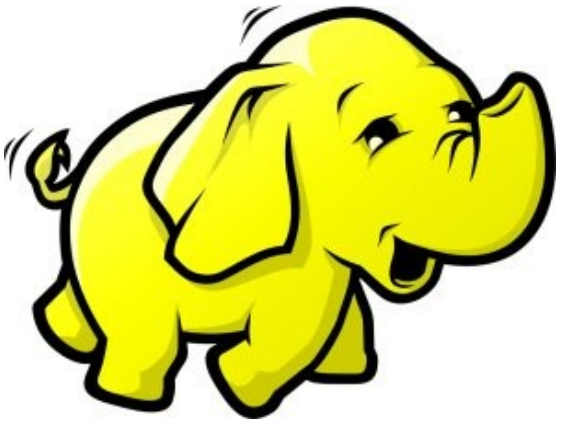
MapReduce 程式設計入門

MapReduce Programing 101

Jazz Wang

Yao-Tsung Wang

jazz@nchc.org.tw



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]));
        Other configure 其他的設定參數程式碼
        job.waitForCompletion(true);
}}
```

Mapper 程式碼

Reducer 程式碼

Program Prototype (v 0.18)

Map 區

Reduce 區

設定區

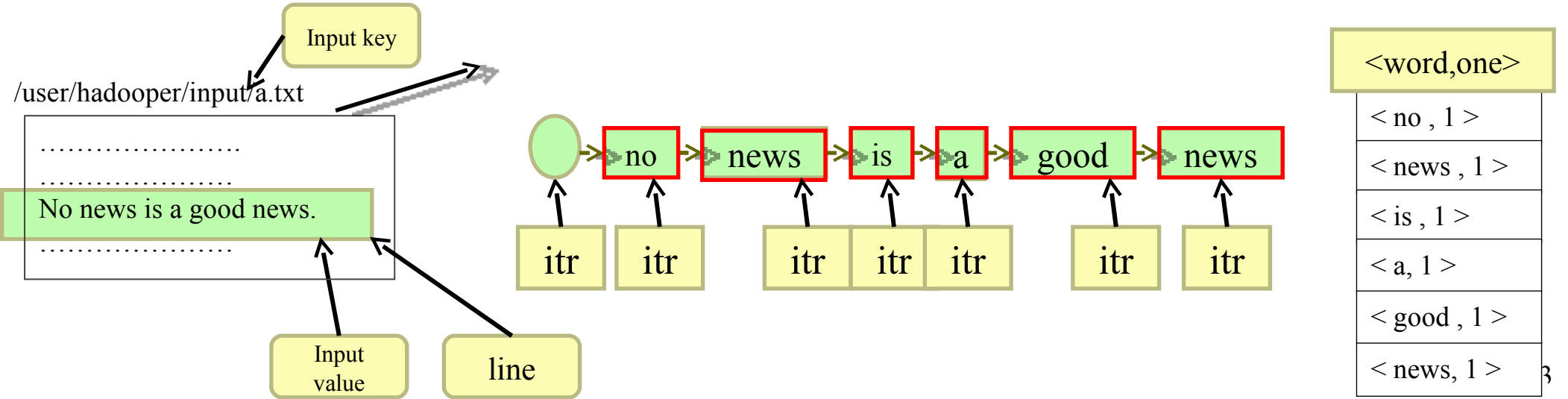
```
Class MR{
    static public Class Mapper …{
    }
    static public Class Reducer …{
    }
    main(){
        JobConf conf = new JobConf( MR.class );
        conf.setMapperClass(Mapper.class);
        conf.setReduceClass(Reducer.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        Other configure 其他的設定參數程式碼
        JobClient.runJob(conf);
}}
```

Map 程式碼

Reduce 程式碼

Word Count - mapper

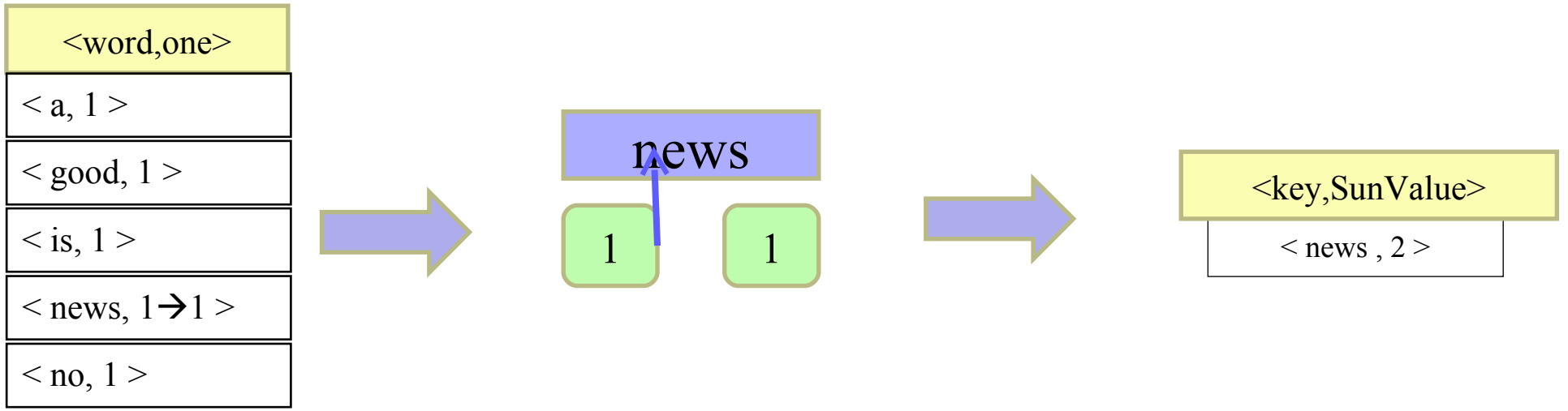
```
1  class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
       private final static IntWritable one = new IntWritable(1);
2      private Text word = new Text();
3      public void map( LongWritable key, Text value, Context context)
4      throws IOException , InterruptedException {
           String line = ((Text) value).toString();
5          StringTokenizer itr = new StringTokenizer(line);
6          while (itr.hasMoreTokens()) {
7              word.set(itr.nextToken());
8              context.write(word, one);
9  }}}
```



Word Count - reducer

```
1  class MyReducer extends Reducer< Text, IntWritable, Text, IntWritable> {
2          IntWritable result = new IntWritable();
3          public void reduce( Text key, Iterable <IntWritable> values, Context context)
           throws IOException, InterruptedException {
4                  int sum = 0;
5                  for ( IntWritable val : values )
6                  sum += val.get();
7                  result.set(sum);
8                  context.write ( key, result);
   }}
```

```
for ( int i ; i < values.length ; i ++ ){
  sum += values[i].get()
}
```



Word Count – main program

```
Class WordCount{
    main()
        Configuration conf = new Configuration();
        Job job = new Job(conf, “job name” );
        job.setJarByClass(thisMainClass.class);
        job.setMapperClass(MyMapper.class);
        job.setReduceClass(MyReducer.class);
        FileInputFormat.addInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);
}}
```

Questions?

Slides - http://trac.nchc.org.tw/cloud

Jazz Wang
Yao-Tsung Wang
jazz@nchc.org.tw