Map Reduce Programming

王耀聰 陳威宇
Jazz@nchc.org.tw
waue@nchc.org.tw
國家高速網路與計算中心(NCHC)

Outline

- 概念
- 程式基本框架及執行步驟方法
- 範例一:
  - Hadoop 的 Hello World => Word Count
  - 說明
  - 動手做
- 範例二:
  - 進階版=> Word Count 2
  - 說明
  - 動手做

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

程式基本框架

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

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);
        }
}
```

程式基本框架

Class Mapper (v0.18)

```
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);
        }
}
```

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);
        }
}
```

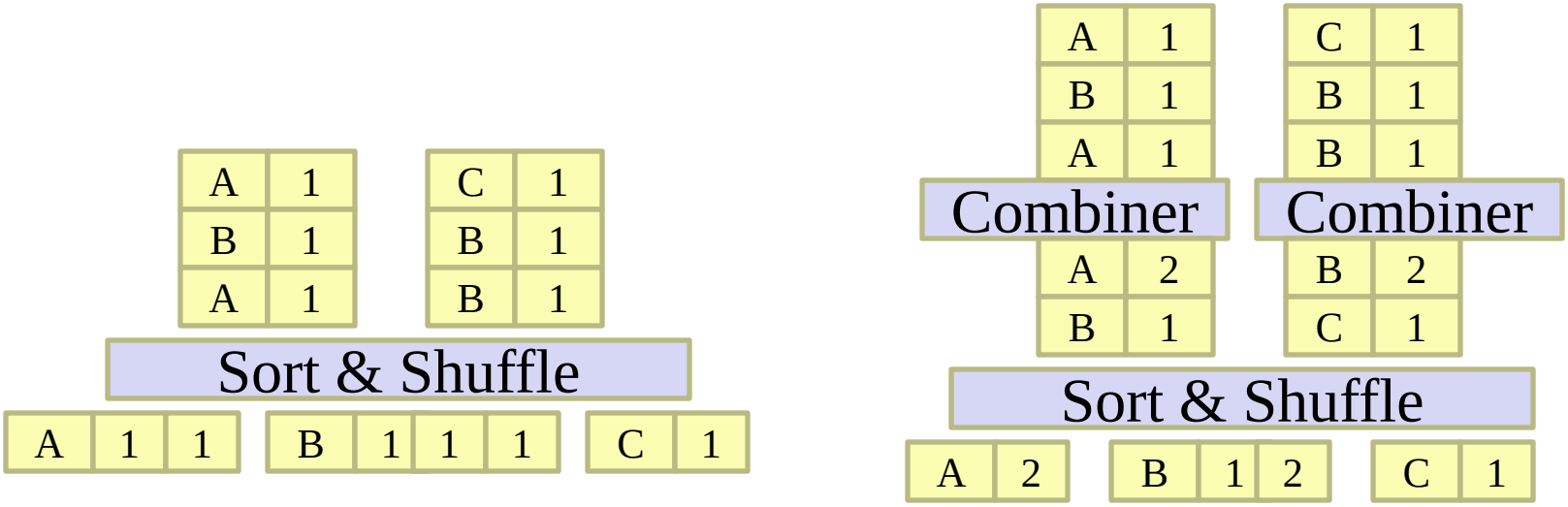
程式基本框架

Class Reducer (v0.18)

```
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);
        }
}
```

Class Combiner

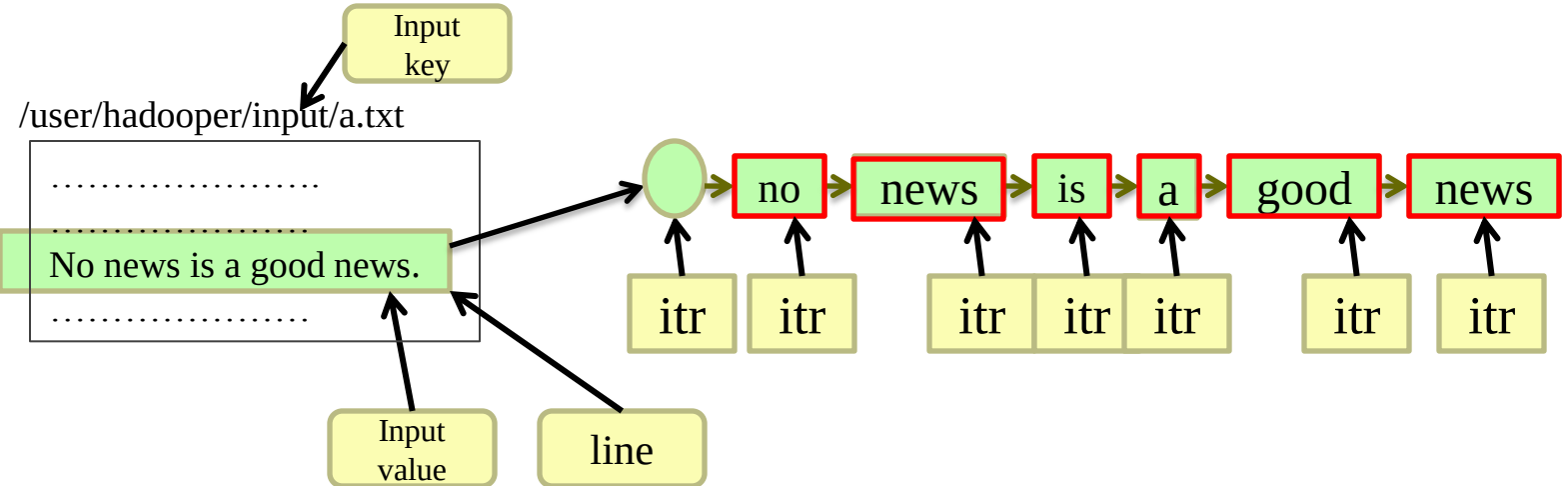
- 指定一個combiner，它負責對中間過程的輸出進行聚集，這會有助於降低從Mapper 到 Reducer數據傳輸量。
  - 引用Reducer
  - Job.setCombinerClass(Class);



範例程式一

Word Count - map

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



<word,one>
< no , 1 >
< news , 1 >
< is , 1 >
< a, 1 >
< good , 1 >
< news, 1 >

範例程式一

Word Count - reduce

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

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

<word,one>
< a, 1 >
< good, 1 >
< is, 1 >
< news, 1→1 >
< no, 1 >

news

1 → 1

<key,SunValue>
< news , 2 >

範例
程式一

Word Count Sample (3)

```
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);
}}
```

執行流程
基本步驟

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
- 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

範例一
動手做

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$
```

範例一
動手做

WordCount1 練習 (II)

1. 編輯WordCount.java
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

範例一
動手做

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
```

範例一
動手做

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
```

範例二
動手做

WordCount 進階版

- WordCount2
- 功能
  - 不計標點符號
  - 不管大小寫
- 步驟 (接續 WordCount 的環境)
  1. echo "\." >pattern.txt && echo "\," >>pattern.txt
  2. bin/hadoop dfs -put pattern.txt ./
  3. mkdir MyJava2
  4. javac -classpath hadoop-*-core.jar -d MyJava2 WordCount2.java
  5. jar -cvf wordcount2.jar -C MyJava2 .

範例二
動手做

不計標點符號

- 執行
  - bin/hadoop jar wordcount2.jar WordCount2 input output2 -skip pattern.txt dfs -cat output2/part-00000

```
waue@vPro:/opt/hadoop$ bin/hadoop dfs -cat output2/part-00000
Cloud   2
Course  2
I       2
NCHC    1
and     1
course  1
enjoy   1
like    2
nchc    1
this    1
we      1
```

範例二
動手做

不管大小寫

- 執行
  - bin/hadoop jar wordcount2.jar WordCount2 -Dwordcount.case.sensitive=false input output3 -skip pattern.txt

```
waue@vPro:/opt/hadoop$ bin/hadoop dfs -cat output3/part-00000
and     1
cloud   2
course  3
enjoy   1
i       2
like    2
nchc    2
this    1
we      1
```

範例二
補充說明

Tool

- 處理Hadoop命令執行的選項

  -conf <configuration file>

  -D <property=value>

  -fs <local|namenode:port>

  -jt <local|jobtracker:port>

- 透過介面交由程式處理
  - ToolRunner.run(Tool, String[])

範例二
補充說明

DistributedCache

- 設定特定有應用到相關的、超大檔案、或只用來參考卻不加入到分析目錄的檔案
  - 如pattern.txt檔
- DistributedCache.addCacheFile(URI,conf)
  - URI＝hdfs://host:port/FilePath

非JAVA
不可？

Options without Java

- 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

結論

- Hadoop 提供乾淨簡潔的API，程式開發者只需要專注於Map Reduce的演算法，其他細節交給函式庫幫忙
- 透過Hadoop的API，寫平行分散運算程式，演算法需符合Map-Reduce 以及 Key-Value
- 問題
  - 透過Streaming或Pipes的技術，是否就可以直接把以前的程式碼放到 Hadoop上跑？
  - Map的輸入為文件檔的某一行，所以是否Hadoop就只能處理類似字數統計的應用？