



Assignment #1 Hadoop

Introduction to Big Data Analytics

TA: Dongmin Hyun and Junyoung Hwang

(dm.hyun@postech.ac.kr, jyhwang@postech.ac.kr)

Overview

Assignment name	File for submission	Due date	
Inverted index	Join.java		
Join	InvertedIndex.java	10/15	
Matrix multiplication 1	Multiplication1_1.java		
Matrix multiplication 2	Multiplication1_2.java		
Matrix multiplication 3	Multiplication2.java		

- You must...
 - Submit the assignments to LMS
 - Utilize the MapReduce Framework
 - Read the lecture note and comments in given template codes carefully
 - Make sure that the output of your code is the same as the output provided
 - Do it yourself
- If your code encounters errors (compile time or runtime), the score will be zero.

HW 1 - Inverted index

• Submission : LMS

• File for submission : InvertedIndex.java

There are data and outputs under HW1_templates/InvertedIndex/

ID	Text	Term	Document ids
	Baseball is played during summer months.	baseball	[1]
		during	[1]
2	Summer is the time for picnics here.	found	[3]
3	Months later we found out why.	here	[2], [4]
4	Why is summer so hot here	hot	[4]
1	Sample document data	is	[1], [2], [4]
		months	[1], [3]
	Dictionary and posting lists \rightarrow	summer	[1], [2], [4]
		the	[2]
		why	[3], [4]

HW 1 - Inverted index

- Building Inverted Index from multiple files
 - Input files should be located in /user/input/ in HDFS
 - Your program must read files in the given directory.
 - Your output directory is /user/output/ in HDFS

Tips

- Use setup() and context in the Mapper to get the file names.
 - setup() is called once for each Map task (for each chunk) while map() is called multiple times (for each line).
- Write a loop in the main() to add all input files in the directory to the job.

HW 2 - Join

• Submission : LMS

• File for submission : Join.java

There are data, outputs, and template codes under HW1_templates/Join/

- Goal
 - Implement below relational join as a MapReduce query

SELECT *
FROM order, line_item
WHERE order.order_id = line_item.order_id

HW 2 - Join

- Input data 'records' file
 - There are 'order' and 'line_item' table in the file.
 - 1st column table name
 2nd column order_id (join them based on this column)
- InputFormat
 - TextInputFormat
- Output data
 - part-r-00000
- Program parameter (5 parameters)
 - [Input file] [output path] [table names delimited by comma(,)] [an index of 1st table for join] [an index of 2nd table for join]

HW 2 - Join

Constraint

- Don't touch main function and input/output key/value type
- You can use some data structure for Caching in Reduce task
- Result records are concatenated with records from "Order" and records from "line_item"

HW 2 - Join - How to test?

```
$HADOOP_HOME/bin/hadoop com.sun.tools.javac.Main Join.java
jar cf join.jar Join*.class
hadoop jar join.jar Join /user/input/records output/join order,line_item 1 1
hdfs dfs -cat output/join/part-r-00000
```

→ We'll check and score this assignment in similar way!

- Submission : LMS
- File for submission: Multiplication1_1.java, Multiplication1_2.java, and Multiplication2.java
- There are data, outputs, and template codes under HW1_templates/MatrixMultiplication/
- Goal
 - Implement a MapReduce algorithm to compute three matrix multiplications
 - Multiplication1_1: a*b (Two matrices in single file)
 - Multiplication1_2: a*b*c (Three matrices in two files)
 - Multiplication2 : a*b*c (Three matrices in single file)

- Input data matrix1_1, matrix1_2, matrix2
 - a: 3 X 5, b: 5 x 2, c: 2 x 3
 - Matrix name and element information are <u>separated by a tap</u>
 e.g.) matrix_name i,j,value
 - <u>It's a sparse matrix format.</u>
 If the value is 0, then the related record is not explicitly presented.
- Output data
 - output/multiple1_1/part-r-00000
 - output/multiple1_2/part-r-00000
 - output/multiple1_2/final/part-r-00000
 - output/multiple2/part-r-00000

Constraint

- Don't touch <u>main function</u> and input/output <u>key/value type</u>
- You must implement Multiplication1_2.java after implementing Multiplication1_1.java.
 - In Multiplication1_2.java, job1's output must be input of Matrix1_2_1_Mapper. If you don't modify the main function, this procedure works properly.
- Result must be the same as the given solution
 - Final result of Multiplication1_2 should be located in 'final' directory of the directory of intermediate result.
 - Output formats of Multiplication1_1, Multiplication1_2 and Multiplication2 are [row column value] (separated by a tap).

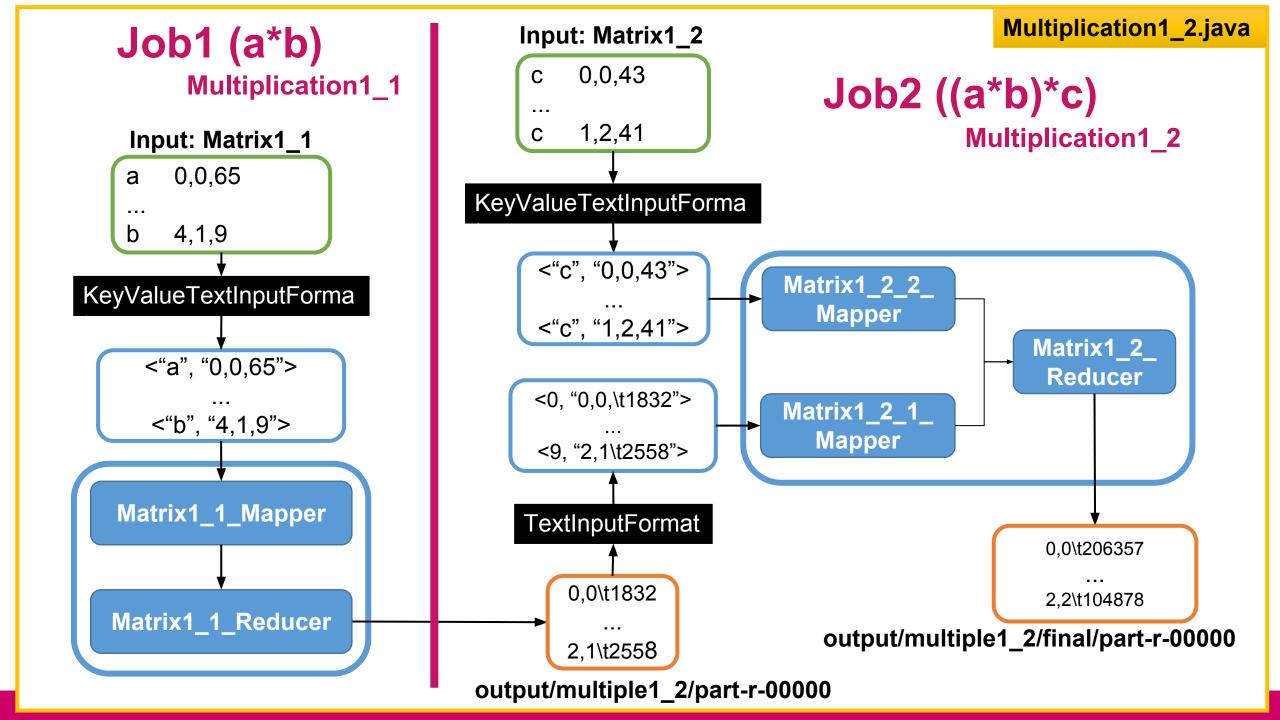
- Goal
 - Implement a MapReduce algorithm to compute the multiplication of "two" matrices which <u>comes from single file</u>.

- Source code Multiplication1_1.java
- Input Data matrix1_1
- InputFormat KeyValueTextInputFormat
- Output Data output/multiple1_1/part-r-00000
- Program parameter 5 parameters
 - [Input file] [output path] [# of first matrix's rows] [# of first matrix's columns] [# of second matrix's columns]

- Goal
 - Implement a MapReduce algorithm to compute the multiplication of "three" matrices which <u>comes from two different files</u> (a, b in first file, c in second file).
 - Program to compute a*b first using Multiplication1_1, and compute (a*b)*c using the algorithm in Multiplication1_2 <u>sequentially</u>.

- Source code Multiplication1_2.java
 - Multiplication1_1.java must be implemented beforehand.
- Input Data matrix1_1, matrix1_2
- Output Data output/multiple1_2/part-r-00000 (for a*b)
 - output/multiple1_2/final/part-r-00000 (for (a*b)*c)

- MultipleInputs class and Multiple mappers
 - Because the program will <u>use two input files</u> which are different in format, your program should use the MultipleInputs class.
 - Two different InputFormats are used.
 - Two different Mappers are used as well.
- Program parameter 7 parameters
 - [Input file1] [Input file2] [output path] [# of first matrix's rows] [# of first matrix's columns] [# of second matrix's columns] [# of third matrix's columns]
 - Intermediate result (a*b) will be written in [output path] and final result ((a*b)*c) will be written in [output path/final]



- Goal
 - Implement a MapReduce algorithm to compute the multiplication of "three" matrix which comes from single file.

- Source code Multiplication2.java
- Input Data matrix2
- Output Data output/multiple2
- Program parameter 6 parameters
 - [Input file] [output path] [# of first matrix's rows] [# of first matrix's columns]
 [# of second matrix's columns] [# of third matrix's columns]

HW 3 Matrix Multiplication - How to test? (1/2)

Compile the files, make jar file and run the 'jar' file like following commands

1. Multiplication1_1

```
$HADOOP_HOME/bin/hadoop com.sun.tools.javac.Main Multiplication1_1.java
jar cf multiple1_1.jar Multiplication1_1*.class
hadoop jar multiple1_1.jar Multiplication1_1 /user/input/matrix1_1 output/multiple1_1 3 5 2
hdfs dfs -cat output/multiple1_1/part-r-00000
```

Parameters

2. Multiplication1_2 (complies Multiplication1_1.java and Multiplication1_2.java simultaneously)

```
$HADOOP_HOME/bin/hadoop com.sun.tools.javac.Main Multiplication1*.java
jar cf multiple1_2.jar Multiplication1*.class
hadoop jar multiple1_2.jar Multiplication1_2 /user/input/matrix1_1 /user/input/matrix1_2 output/multiple1_2 3 5 2 3
hdfs dfs -cat output/multiple1_2/part-r-00000
```

hdfs dfs -cat output/multiple1_2/final/part-r-00000

HW 3 Matrix Multiplication - How to test? (2/2)

Compile the files, make jar file and run the 'jar' file like following commands

3. Multiplication2

\$HADOOP_HOME/bin/hadoop com.sun.tools.javac.Main Multiplication2.java

jar cf multiple2.jar Multiplication2*.class

hadoop jar multiple2.jar Multiplication2 /user/input/matrix2 output/multiple2 3 5 2 3

hdfs dfs -cat output/multiple2/part-r-00000

Parameters

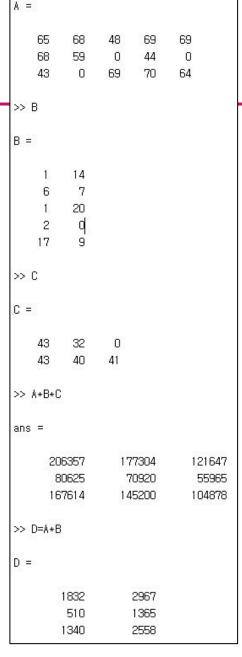


Figure. Matlab example