

# Assignment #1

# Hadoop

Introduction to Big Data Analytics

TA: Dongmin Hyun and Junyoung Hwang

([dm.hyun@postech.ac.kr](mailto:dm.hyun@postech.ac.kr), [jyhwang@postech.ac.kr](mailto:jyhwang@postech.ac.kr))

# Overview

---

Assignment name	File for submission	Due date
Inverted index	Join.java	10/15
Join	InvertedIndex.java	
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
1	Baseball is played during summer months.	baseball	[1]
2	Summer is the time for picnics here.	during	[1]
3	Months later we found out why.	found	[3]
4	Why is summer so hot here	here	[2], [4]
↑	Sample document data	hot	[4]
		is	[1], [2], [4]
		months	[1], [3]
		summer	[1], [2], [4]
		the	[2]
		why	[3], [4]

Dictionary and posting lists →

# 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
```

Parameters

```
hdfs dfs -cat output/join/part-r-00000
```

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



# HW 3 Matrix Multiplication

---

- 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)

# HW 3 Matrix Multiplication

---

- 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 separated by a tap  
e.g.) matrix\_name    i,j,value
  - It's a sparse matrix format.  
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

# HW 3 Matrix Multiplication

---

- Constraint

- Don't touch main function and input/output key/value type
- 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).

# HW 3 Matrix Multiplication1\_1

---

- Goal
  - Implement a MapReduce algorithm to compute the multiplication of “two” matrices which comes from single file.
- 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]

# HW 3 Matrix Multiplication1\_2

---

- Goal
  - Implement a MapReduce algorithm to compute the multiplication of “three” matrices which comes from two different files (*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 sequentially.
- 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  $(a*b)*c$ )

# HW 3 Matrix Multiplication1\_2

---

- MultipleInputs class and Multiple mappers
  - Because the program will use two input files 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]

# Job1 (a\*b)

Multiplication1\_1

Input: Matrix1\_1

a 0,0,65  
...  
b 4,1,9

KeyValueTextInputForma

<"a", "0,0,65">  
...  
<"b", "4,1,9">

Matrix1\_1\_Mapper

Matrix1\_1\_Reducer

Input: Matrix1\_2

c 0,0,43  
...  
c 1,2,41

KeyValueTextInputForma

<"c", "0,0,43">  
...  
<"c", "1,2,41">

<0, "0,0,\t1832">  
...  
<9, "2,1\t2558">

TextInputFormat

0,0\t1832  
...  
2,1\t2558

output/multiple1\_2/part-r-00000

Multiplication1\_2.java

# Job2 ((a\*b)\*c)

Multiplication1\_2

Matrix1\_2\_2\_Mapper

Matrix1\_2\_Reducer

Matrix1\_2\_1\_Mapper

0,0\t206357  
...  
2,2\t104878

output/multiple1\_2/final/part-r-00000

# HW 3 Matrix Multiplication2

---

- 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 (compiles 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

```
A =  
  
    65    68    48    69    69  
    68    59     0    44     0  
    43     0    69    70    64  
  
>> B  
  
B =  
  
     1    14  
     6     7  
     1    20  
     2     4  
    17     9  
  
>> C  
  
C =  
  
    43    32     0  
    43    40    41  
  
>> A+B+C  
  
ans =  
  
    206357    177304    121647  
     80625     70920     55965  
    167614    145200    104878  
  
>> D=A*B  
  
D =  
  
     1832     2967  
      510     1365  
     1340     2558
```

Figure. Matlab example