

Homework #1: Block-Based Matrix Multiplication using Hadoop MapReduce & Apache Spark

Due: September 23, Sunday
100 points

In this homework, we consider using Hadoop MapReduce and Spark to implement the block-based matrix multiplication. You can assume that each matrix is stored as a text file. Each line of the file corresponds to a block of the matrix in the format of index-content. You may assume all blocks are 2-by-2 and all matrices are 6-by-6. That is each matrix is divided into 9 blocks of equal size (2-by-2 or 4 elements). The content of block is stored in a sparse format: (row index, column index, value) where row and column indexes are relative to the block. All indexes start from 1.

$$\begin{array}{cc}
 A^{11} & A^{12} \\
 \uparrow & \uparrow \\
 \left[\begin{array}{cc} [2 & 1] \\ [0 & 3] \\ [4 & 2] \\ [1 & 2] \\ [3 & 0] \\ [1 & 1] \end{array} \right] & \left[\begin{array}{cc} [0 & 3] \\ [2 & 0] \\ [1 & 2] \\ [2 & 1] \\ [0 & 1] \\ [1 & 0] \end{array} \right] \quad \left[\begin{array}{cc} [1 & 3] \\ [2 & 0] \\ [3 & 0] \\ [2 & 2] \\ [2 & 0] \\ [0 & 0] \end{array} \right]
 \end{array}$$

For example, the following is a fragment of a file for the matrix A above.

```
(1,1),[(1,1,2),(1,2,1),(2,2,3)]
(1,2),[(1,2,3),(2,1,2)]
...
```

Note that the first line describes the block A^{11} , 2nd line describes A^{12} . Note also the format as illustrated (e.g., [] encloses the content of a block; no white spaces used; comma separates the numbers, etc.).

- [70 points] Write a Hadoop MapReduce program, BlockMult.java, to multiply matrix A and B. Use one-phase approach and we use only one reducer here as described in class.

Sample invocation:

```
hadoop jar bm.jar BlockMult file-A file-B output_path
```

Where dir-A stores the content of matrix A in the format described above.

- [30 points] Implement the same function as question 1 but use Spark in Python instead. Name your program: BlockMult.py. (You do not need output path as the specific directory here, you can just create a output.txt in your code).

Sample invocation:

```
spark-submit BlockMult.py file-A file-B (output_path)
```

Submission notes:

- Submit a zip file that contain your solution, input and output files as shown below. Name the zip file as Lastname_Firstname_hw1.zip. For example, Smith_John_hw1.zip.

Name	Date Modified
▼ output	Today at 11:48 PM
output.txt	Today at 1:52 PM
▼ solutions	Today at 1:54 PM
BlockMult.py	Today at 1:52 PM
BM.jar	Yesterday at 8:26 P
BlockMult.java	Yesterday at 8:25 P
▼ input	Yesterday at 9:46 P
file-A.txt	Sep 4, 2018 at 2:01
file-B.txt	Sep 4, 2018 at 2:01

- Your input should be formatted according to the sample output below and we do not specify the size of matrix B here, but we can assume matrix A and B have the same size:

```

(1,1),[(1,1,2),(1,2,1),(2,2,3)]
(1,2),[(1,2,3),(2,1,2)]
(1,3),[(1,1,1),(1,2,3),(2,1,2)]
(2,1),[(1,1,4),(1,2,2),(2,1,1),(2,2,2)]
(2,2),[(1,1,1),(1,2,2),(2,1,2),(2,2,1)]
(2,3),[(1,1,3),(2,1,2),(2,2,2)]
(3,1),[(1,1,3),(2,1,1),(2,2,1)]
(3,2),[(1,2,1),(2,1,1)]
(3,3),[(1,1,2)]

```

- Your output should be formatted according to the sample output below (these values is gotten from $A \cdot A$), and if a block contains all zeros, then the whole row will not show up in input or output:

```

(1, 1) ,[(1, 1, 13),(1, 2, 14),(2, 1, 14),(2, 2, 13)]
(1, 2) ,[(1, 1, 11),(1, 2, 10),(2, 1, 8),(2, 2, 6)]
(1, 3) ,[(1, 1, 12),(1, 2, 12),(2, 1, 16)]
(2, 1) ,[(1, 1, 23),(1, 2, 16),(2, 1, 19),(2, 2, 15)]
(2, 2) ,[(1, 1, 9),(1, 2, 19),(2, 1, 10),(2, 2, 10)]
(2, 3) ,[(1, 1, 21),(1, 2, 16),(2, 1, 17),(2, 2, 5)]
(3, 1) ,[(1, 1, 13),(1, 2, 5),(2, 1, 6),(2, 2, 6)]
(3, 2) ,[(1, 1, 2),(1, 2, 12),(2, 1, 3),(2, 2, 5)]
(3, 3) ,[(1, 1, 9),(1, 2, 11),(2, 1, 6),(2, 2, 3)]

```