Lab 2: Matrix Multiply with Hadoop-Streaming

DS-GA 1004 (Big Data)

February 5, 2018

Download and Follow Lab2.pdf

1. Log in (via ssh) to Dumbo (see Lab 1 for instructions)
2. Copy the Lab 2 files from HDFS to your home directory on Dumbo. To do this, type

```
hfs -get /user/ecc290/lab2
```

3. Load python 3.4.4. Type:

```
module load python/gnu/3.4.4
```

4. Type `cd lab2` to move into the folder you just copied and type `ls` to see the files.

You should see:

- src: a directory with the following source files:
  - map.py: the Python code skeleton for the map function
  - map.sh: a Bash script to load the necessary libraries for the map.py code
  - reduce.py: the Python code skeleton for the reduce function
  - reduce.sh: a Bash script to load the necessary libraries for the map.py code
- matsmall.txt: an input file with small input matrices for testing your code
- example: a directory with example map and reduce Python codes for the wordcount example from Lab 1. These may be helpful for you to examine as you flesh out the map.py and reduce.py skeletons.

Together: Let's look at the files in the example folder and go over the structure of the map and reduce functions

```
#!/usr/bin/env python

# example using hadoop-streaming, from
# http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-pytho
n/

import sys
import string
# the system python does not have numpy, but the python/gnu/3.4.4 does
# (we don't actually need it, but attempting to import it will trigger
# an error if the mapper can't see the version of python we want to use)
import numpy

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        w=word.strip(string.punctuation)
        if w:
            print('{0:s}\t{1:d}'.format(w.lower(), 1))
```

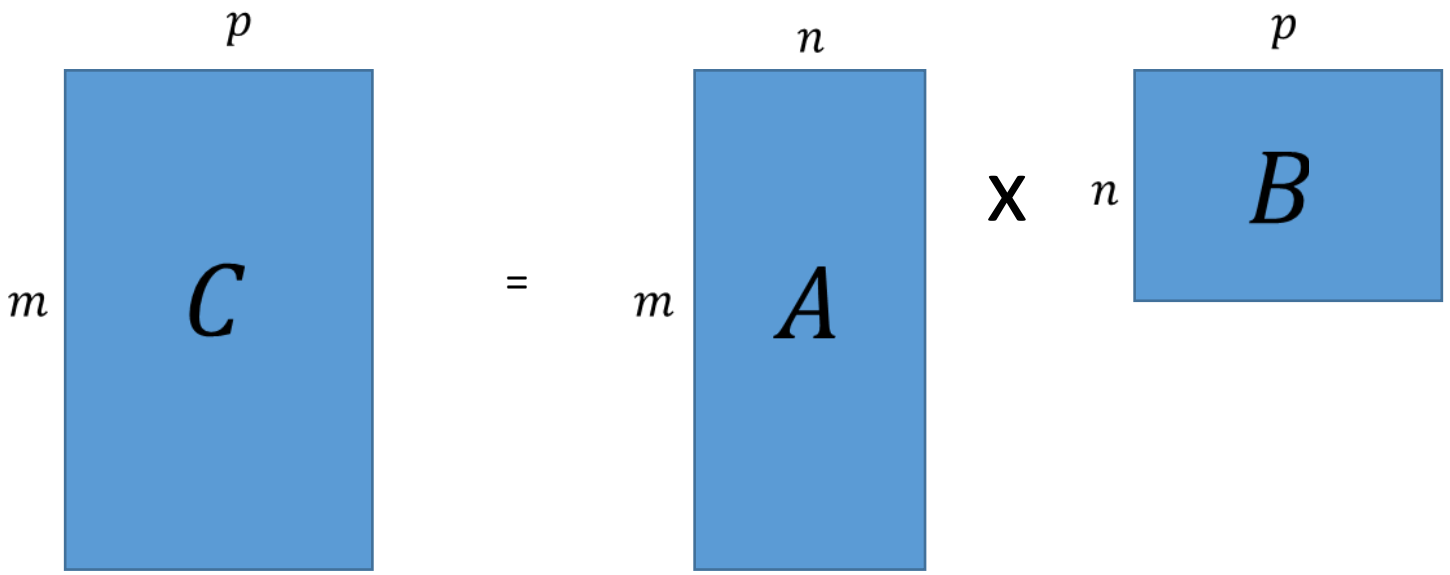
```
#!/usr/bin/env python

# example using hadoop-streaming, from
# http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

from operator import itemgetter
import sys
# the system python does not have numpy, but the python/gnu/2.7.10 does
# (we don't actually need it, but attempting to import it will trigger
# an error if the mapper can't see the version of python we want to use)
#import numpy

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print('{0:s}\t{1:d}'.format(current_word, current_count))
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
        print('{0:s}\t{1:d}'.format(current_word, current_count))
~
```

What are the .sh files?

```
#!/bin/bash

# the Hadoop-streaming enviroment does not read the standard bash startup files,
# so we must use a wrapper to explicitly set up the modules environment and load
# the relevant modules

. /etc/profile.d/modules.sh
module load python/gnu/3.4.4
wc_mapper.py
```

```
#!/bin/bash

# the Hadoop-streaming enviroment does not read the standard bash startup files,
# so we must use a wrapper to explicitly set up the modules environment and load
# the relevant modules

. /etc/profile.d/modules.sh
module load python/gnu/3.4.4
wc_reducer.py
```

Today's task: Matrix Multiplication



C = A x B

Today's task: Matrix Multiplication

$$\begin{bmatrix} 1 & 8 & 5 \\ 0 & 9 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

$$C(i,k) = \sum_{j=0}^{n-1} A(i,j) * B(j,k)$$

Today's task: Matrix Multiplication

$$\begin{bmatrix} 1 & 8 & 5 \\ 0 & 9 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

$$C(i,k) = \sum_{j=0}^{n-1} A(i,j) * B(j,k)$$

Today's task: Matrix Multiplication

$$\begin{bmatrix} 1 & 8 & 5 \\ 0 & 9 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

$$C(i,k) = \sum_{j=0}^{n-1} A(i,j) * B(j,k)$$

Today's task: Matrix Multiplication

$$\begin{bmatrix} 1 & 8 & 5 \\ 0 & 9 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

$$C(i,k) = \sum_{j=0}^{n-1} A(i,j) * B(j,k)$$

Input File Format

```
A,0,0,0.0
A,0,1,1.0
A,0,2,2.0
A,0,3,3.0
A,0,4,4.0
A,1,0,5.0
A,1,1,6.0
A,1,2,7.0
A,1,3,8.0
A,1,4,9.0
B,0,0,0.0
B,0,1,1.0
B,0,2,2.0
B,1,0,3.0
B,1,1,4.0
B,1,2,5.0
B,2,0,6.0
B,2,1,7.0
B,2,2,8.0
B,3,0,9.0
B,3,1,10.0
B,3,2,11.0
B,4,0,12.0
B,4,1,13.0
B,4,2,14.0
```

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix}$$

Naïve Matrix Multiplication w/MapReduce

Make each index in C a key (i.e., 0 0, 0 1, 0 2, 1 0, 1 1, 1 2)

For each entry in A, create p (key-value) pairs (since a given entry of A will be needed to compute p entries in C

For each entry in B, create m (key-value) pairs (since a given entry of A will be needed to compute m entries in C

$$C = \begin{bmatrix} \_ & \_ & \_ \\ \_ & \_ & \_ \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix}$$

Naïve Matrix Multiplication w/MapReduce

Make each index in C a key (i.e., 0 0, 0 1, 0 2, 1 0, 1 1, 1 2)

For each entry in A, create p (key-value) pairs (since a given entry of A will be needed to compute p entries in C

For each entry in B, create m (key-value) pairs (since a given entry of A will be needed to compute m entries in C

Example: For line
A, 1, 2, 7.0
Generate key-value pairs
<1 0, A 2 7.0>, <1 1, A 2 7.0>, <1 2, A 2 7.0>

$$C = \begin{bmatrix} \_ & \_ & \_ \\ \_ & \_ & \_ \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix}$$

Naïve Matrix Multiplication w/MapReduce

Make each index in C a key (i.e., 0 0, 0 1, 0 2, 1 0, 1 1, 1 2)

For each entry in A, create p (key-value) pairs (since a given entry of A will be needed to compute p entries in C

For each entry in B, create m (key-value) pairs (since a given entry of A will be needed to compute m entries in C

Example: For line
A, 1, 2, 7.0
Generate key-value pairs
<1 0, A 2 7.0>, <1 1, A 2 7.0>, <1 2, A 2 7.0>

$$C = \begin{bmatrix} - & - & - \\ - & - & - \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix}$$

For line
B, 1, 0, 3.0
Generate key-value pairs
<0 0, B 1 3.0>, <1 0, B 1 3.0>

Reduce Function

A reducer that receives key (i k) will then have all the data it needs to compute entry $C(i,k)$.

Example: For key (0 1), the reducer will get

<0 1, A 0 0.0>, <0 1, A 1 1.0>, <0 1, A 2 2.0>, <0 1, A 3 3.0>, <0 1, A 4 4.0>,
<0 1, B 0 1.0>, <0 1, B 1 4.0>, <0 1, B 2 7.0>, <0 1, B 3 10.0>, <0 1, B 4 13.0>

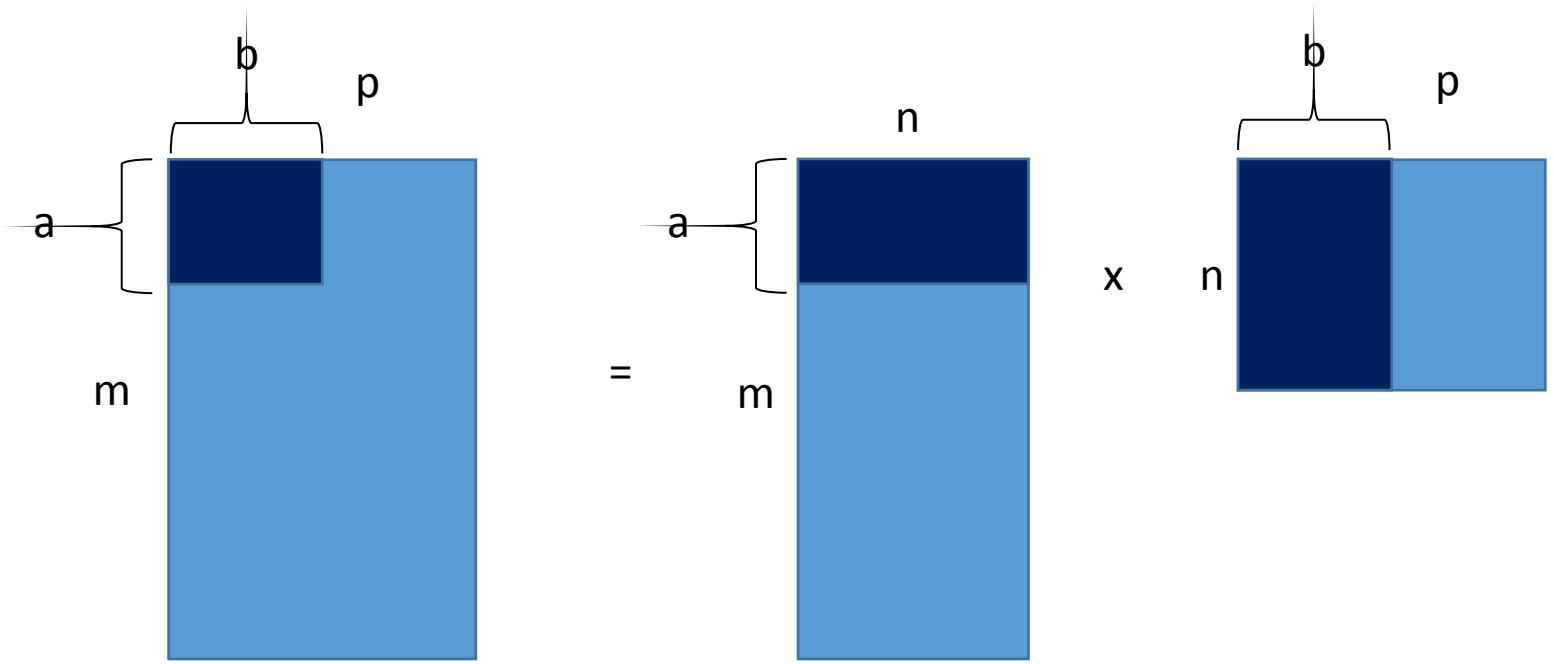
$$C = \begin{bmatrix} - & - & - \\ - & - & - \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix}$$

The reducer can store these entries in local arrays, and compute their result once all entries for this key have been received.
The reduce then outputs a line to STDOUT of the form, e.g.,

```
(0, 1), 100.0
```

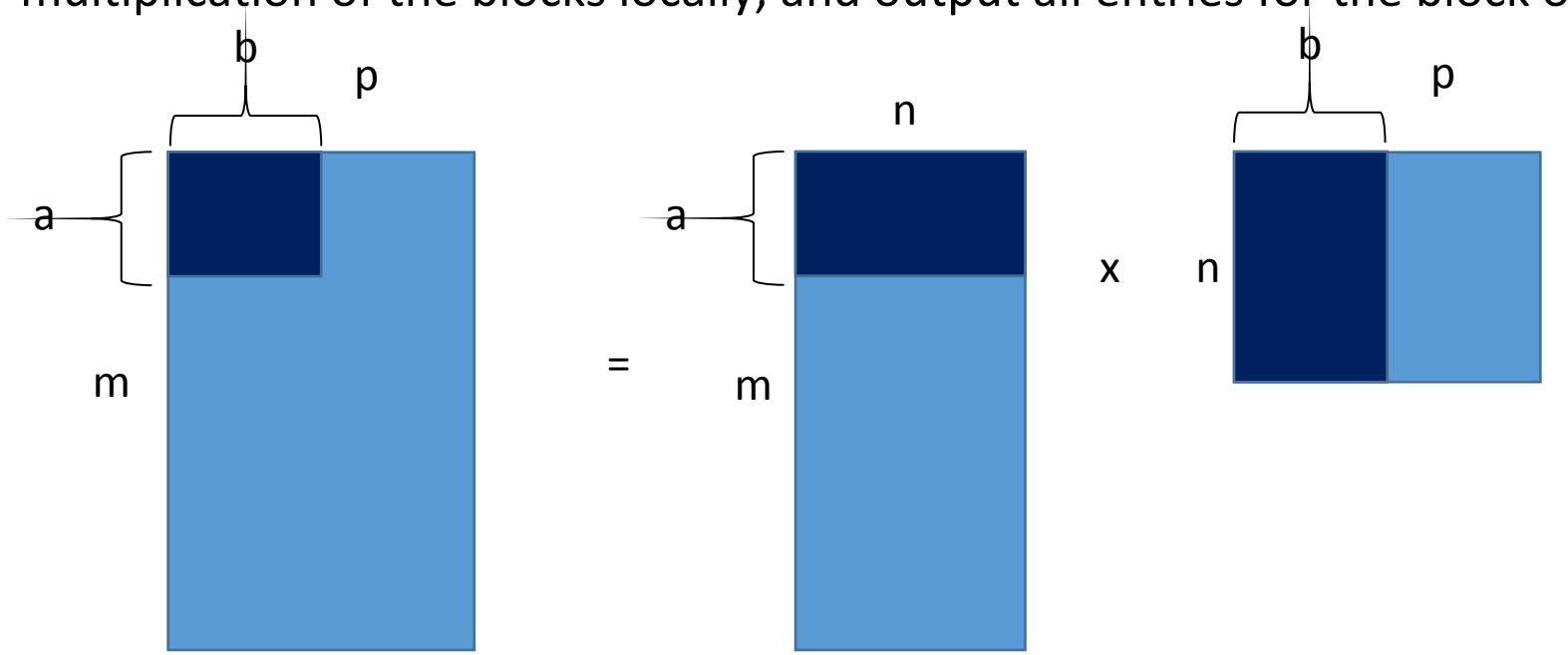
A Smarter Matrix Multiply

- This naïve version generates a lot of intermediate data (becomes very large for large matrices)
- The performance can be improved by blocking
- Instead of making a key for each entry in C, group the entries of C into larger blocks and *use a key for each block*



A Smarter Matrix Multiply

- This naïve version generates a lot of intermediate data (becomes very large for large matrices)
- The performance can be improved by blocking
- Instead of making a key for each entry in C, group the entries of C into larger blocks and *use a key for each block*
- Then each entry of A creates p/b key-value pairs (versus p), and each entry of B creates m/a key-value pairs (versus m)
- Reducer can reconstruct local submatrices of A and B, perform the multiplication of the blocks locally, and output all entries for the block of C.



Now it's your turn

- Flesh out the skeleton map.py and reduce.py files to implement matrix multiplication
  - Look at the wordcount example files (Lab 1) if you need guidance on Python syntax, structure, etc.

- Test locally with matsmall.txt

- Once your code is working, use Hadoop-Streaming with larger input matrix (matbig.txt) to study the scalability of your map and reduce functions

Gathering timing data

map took 19 seconds to run

reduce took 13 seconds to run

```
bash-4.1$ hjs -files /home/ecc290/BigData2017/lab2/src  -mapper src/map.sh -reducer src/red
packageJobJar: [] [/opt/cloudera/parcels/CDH-5.9.0-1.cdh5.9.0.p0.23/jars/hadoop-streaming-2
17/01/30 13:25:43 INFO client.RMProxy: Connecting to ResourceManager at babar.es.its.nyu.ed
17/01/30 13:25:43 INFO client.RMProxy: Connecting to ResourceManager at babar.es.its.nyu.ed
17/01/30 13:25:44 INFO mapred.FileInputFormat: Total input paths to process : 1
17/01/30 13:25:45 INFO mapreduce.JobSubmitter: number of splits:2
17/01/30 13:25:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1484865967044
17/01/30 13:25:45 INFO impl.YarnClientImpl: Submitted application application_1484865967044
17/01/30 13:25:45 INFO mapreduce.Job: The url to track the job: http://babar.es.its.nyu.edu
17/01/30 13:25:45 INFO mapreduce.Job: Running job: job_1484865967044_0662
17/01/30 13:25:49 INFO mapreduce.Job: Job job_1484865967044_0662 running in uber mode : fal
17/01/30 13:25:49 INFO mapreduce.Job:  map 0% reduce 0%
17/01/30 13:25:59 INFO mapreduce.Job:  map 41% reduce 0%
17/01/30 13:26:02 INFO mapreduce.Job:  map 62% reduce 0%
17/01/30 13:26:05 INFO mapreduce.Job:  map 67% reduce 0%
17/01/30 13:26:08 INFO mapreduce.Job:  map 100% reduce 0%
17/01/30 13:26:15 INFO mapreduce.Job:  map 100% reduce 63%
17/01/30 13:26:16 INFO mapreduce.Job:  map 100% reduce 88%
17/01/30 13:26:19 INFO mapreduce.Job:  map 100% reduce 96%
17/01/30 13:26:21 INFO mapreduce.Job:  map 100% reduce 100%
17/01/30 13:26:22 INFO mapreduce.Job: Job job_1484865967044_0662 completed successfully
```