Smith College

Computer Science

Introduction to Map-Reduce

CSC352—Week #11

Dominique Thiébaut
dthiebaut@smith.edu

The Reference

- **MapReduce: Simplified Data Processing on Large Clusters**, by Dean and Ghemawat, First published in OSDI 2004, also in *Comm. ACM* 51, 1 (January 2008), 107-113.

Inspiration

- CSC490, U. Washington.

Map-Reduce

- Based on **Functional Programming**
- 3 Major Functions in Functional Programming:
  - *Map*
  - *Reduce*
  - ~~Filter~~

Properties of Functional Programming

- Functional operations **do not modify data**. New data is created (immutable data)

- Original data **always available** (can rollback easily)

- Data flow is **implicit** (no need to set communication pattern)

- **Order of operations** does not influence result (free to restart slow operations)

Mapping Example in Python

```
>>> L = [ "hello\n  ", "   there   \n", "  Smithies!  " ]
>>> L2 = [ line.strip() for line in L ]
>>> print( L2)
['hello', 'there', 'Smithies!']
>>>
```

Mapping Example in Python

List Comprehension

```
>>> L = [ "hello\n  ", "  there   \n", "  Smithies!  " ]
>>> L2 = [ line.strip() for line in L ]
>>> print( L2)
['hello', 'there', 'Smithies!']
>>>
```

Mapping Example in Python

```
>>> L = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> L3 = [ k+1 for k in L if k%3==0 ]
>>> print( L3 )
[4, 7, 10]
```

The Python Map Function

```
>>> def firstLast( s ):
        return s[0]+s[-1]

>>> L = [ 'Hello', 'There', 'Smith' ]
>>> L2 = map( firstLast, L )
>>> list( L2 )
['Ho', 'Te', 'Sh']
```

The Python Filter Function

```
>>> def isMultOf3( x ):
        return x % 3 == 0

>>> L = [ 1, 2, 3, 5, 8, 9, 21, 22, 23 ]
>>> L2 = filter( isMultOf3, L )
>>> list( L2 )
[3, 9, 21]
>>>
```

The Python Reduce Function

```
>>> import functools, operator
>>> L = [ 1, 2, 3, 4 ]
>>> L2 = functools.reduce( operator.add, L, 0 )
>>> L2
10
```

The Python Reduce Function

Reducing a list to the sum of its values so common, that sum() shortcut created.

```
>>>
>>> L = [ 1, 2, 3, 4 ]
>>> L2 = sum( L )
>>> L2
10
```

The Python Reduce Function

```
>>> import functools, operator

>>> L = [ "ACGG", "TTA", "GAT" ]
>>> L2 = functools.reduce( operator.concat, L, "" )
>>> L2
'ACGGTTAGAT'
```

We should be able to Create a Map-Reduce Platform in Python!

Step 1: Mapper

Given a list of words, use Python to **map** each word to the number 1, representing its "level of occurrence".

```
>>> text = """Perfection is achieved, not when there is nothing more to add,
but when there is nothing left to take away. —Saint Exupéry"""
>>> L = text.split()
```

Step 1: Mapper

```
>>> text = """Perfection is achieved, not when there is nothing more to add,
but when there is nothing left to take away. —Saint Exupéry"""
```

```
>>> L = [ (word.strip().lower(), 1 ) for word in text.split() ]
>>> L
[('perfection', 1), ('is', 1), ('achieved', 1), ('not', 1), ('when', 1),
('there', 1), ('is', 1), ('nothing', 1), ('more', 1), ('to', 1), ('add', 1),
('but', 1), ('when', 1), ('there', 1), ('is', 1), ('nothing', 1), ('left', 1),
('to', 1), ('take', 1), ('away', 1), ('—saint', 1), ('exupéry', 1)]
>>>
```

Step 1: Mapper

```
>>> text = """Perfection is achieved, not when there is nothing more to add,
but when there is nothing left to take away. —Saint Exupéry"""
```

```
>>> L = [ (word.strip().lower(), 1 ) for word in text.split() ]
>>> L
[('perfection', 1), ('is', 1), ('achieved', 1), ('not', 1), ('when', 1),
('there', 1), ('is', 1), ('nothing', 1), ('more', 1), ('to', 1), ('add', 1),
('but', 1), ('when', 1), ('there', 1), ('is', 1), ('nothing', 1), ('left', 1),
('to', 1), ('take', 1), ('away', 1), ('—saint', 1), ('exupéry', 1)]
>>>
```

key

value

```
#!/usr/bin/env python
# A basic mapper function/program that
# takes whatever is passed on the input and
# outputs tuples of all the words formatted
# as (word, 1)
from __future__ import print_function
import sys

# input comes from STDIN (standard input)
for line in sys.stdin:

    # create tuples of all words in line
    L = [ (word.strip().lower(), 1 ) for word in line.strip().split() ]

    # output tuples of the form (word, 1)
    for (word, n) in L:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reducer.  Tuple is tab-delimited;
        print( '%s\t%d' % (word, n) )
```

```
getcopy MapReduce/BuildingMapReduce/mapper.py
```

Step 2: Shuffle

Write a Python Program that
takes the tuples output by **mapper.py**
and **sorts** them out alphabetically

Step 2: Shuffle

```
#!/usr/bin/env python
# shuffleSort.py
from __future__ import print_function
import sys

L = []

# input comes from STDIN
for line in sys.stdin:
      # remove leading and trailing whitespace
      word, count = line.strip().split('\t', 1)
      L.append( (word, int(count) ) )

# sort the tuples
L.sort( )

# output the sorted tuples
for (word, count) in L:
    print( '%s\t%d' % (word, count) )
```

```
getcopy MapReduce/BuildingMapReduce/shuffleSort.py
```

Step 3: Reducer

Write a Python Program that
takes the tuples output by **shuffleSort.py**
and reduces them while counting the
ones with similar

Step 3: Reducer

```
#!/usr/bin/env python
# reducer.py
from __future__ import print_function
import sys

lastWord = None
sum = 0

for line in sys.stdin:
      word, count = line.strip().split('\t', 1)
      count = int(count)
      if lastWord==None:
          lastWord = word
          sum = count
          continue

      if word==lastWord:
          sum += count
      else:
          print( "%s\t%d" % ( lastWord, sum ) )
          sum = count
          lastWord = word

# output last word
if lastWord == word:
    print( '%s\t%s' % (lastWord, sum ) )
```

```
getcopy MapReduce/BuildingMapReduce/reducer.py
```

Computing Word Frequencies

```
352b@aurora ~ $ cat > dummy.txt
this is
a text with
several lines
everything is lowercase.
^D
352b@aurora ~ $ cat dummy.txt | ./mapper.py | ./shuffleSort.py | ./reducer.py
class.  1
csc352  1
everything 1
is 3
lines 1
lowercase. 1
several 1
text  1
the   1
this  1
with  1
```

Computing Word Frequencies

James Joyce's *Ulysses*

```
352b@aurora ~ $ wget http://cs.smith.edu/~dthiebaut/gutenberg/4300-8.txt
--2017-04-11 05:47:00--  http://cs.smith.edu/~dthiebaut/gutenberg/4300-8.txt
Resolving cs.smith.edu (cs.smith.edu)... 131.229.72.74
Connecting to cs.smith.edu (cs.smith.edu)|131.229.72.74|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1573082 (1.5M) [text/plain]
Saving to: ‘4300-8.txt’

100%[=============================================================>] 1,573,082   --.-K/s   in 0.008s

2017-04-11 05:47:00 (183 MB/s) - ‘4300-8.txt’ saved [1573082/1573082]

352b@aurora ~ $ cat 4300-8.txt | ./mapper.py | ./shuffleSort.py | ./reducer.py | less
"defects,"      1
"i      1
"information    1
"j"     1
"plain  1
"project        2
```

Computing Word Frequencies

```
352b@aurora ~ $ wget http://cs.smith.edu/~dthiebaut/gutenberg/4300-8.txt
--2017-04-11 05:47:00--  http://cs.smith.edu/~dthiebaut/gutenberg/4300-8.txt
Resolving cs.smith.edu (cs.smith.edu)... 131.229.72.74
Connecting to cs.smith.edu (cs.smith.edu)|131.229.72.74|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1573082 (1.5M) [text/plain]
Saving to: ‘4300-8.txt’

100%[=============================================================>] 1,573,082   --.-K/s   in 0.008s

2017-04-11 05:47:00 (183 MB/s) - ‘4300-8.txt’ saved [1573082/1573082]

352b@aurora ~ $ cp 4300-8.txt Ulysses.txt
352b@aurora ~ $ cat 4300-8.txt | ./mapper.py | sort | ./reducer.py | less
"defects,"      1
"i      1
"information    1
"j"     1
"plain  1
"project        2
```

We can use Linux's sort command!

Exercise 1

- Use the Map-Reduce framework we just created to find the **most frequent word**(s) in a text file.

Exercise 1

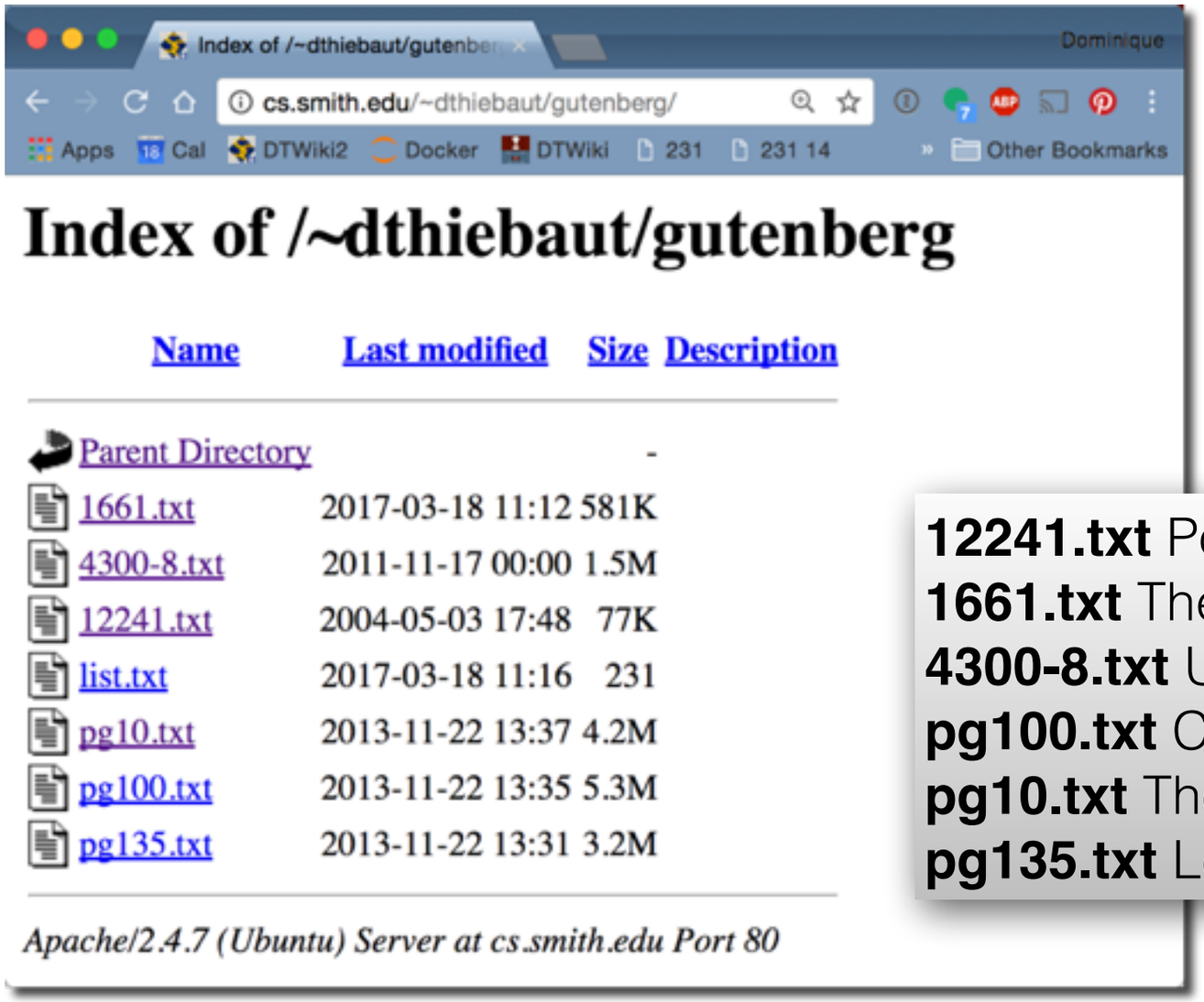
```
352b@aurora ~ $ time cat ulysses.txt | ./mapper1.py | ./shuffleSort.py | ./reducer1.py
| ./mapper2.py | sort -n | ./reducer2.py
the,  14854

real  0m2.513s
user  0m2.417s
sys   0m0.077s


352b@aurora ~ $ time cat ulysses.txt | tr -c '[:alnum:]' '[\n*]' | sort | uniq -c | sort
-nr | head  -2
 100963
  13683 the

real  0m0.958s
user  0m0.931s
sys   0m0.019s
```

Same work, using Linux commands

http://unix.stackexchange.com/questions/41479/find-n-most-frequent-words-in-a-file

Exercise 2
Multiple Input Files

- Using the same "command-line-Python" approach, find the word-frequencies of several books at once.



12241.txt Poems: Third Series, by Emily Dickinson
1661.txt The Adventures of Sherlock Holmes
4300-8.txt Ulysses
pg100.txt Complete Works of William Shakespeare
pg10.txt The King James Bible
pg135.txt Les Miserables, by Victor Hugo

Exercise 2
Multiple Input Files

```
352b@aurora ~ $ cat wordCount1.sh
#! /bin/bash

files=$@
{ for file in $files; do cat $file; done; }  | ./mapper.py | sort | ./reducer.py


352b@aurora ~ $ ./wordCount1.sh 4300-8.txt pg10.txt | wc
74863  149722  787514
```

http://unix.stackexchange.com/questions/41479/find-n-most-frequent-words-in-a-file

Exercise 2
Timing

```
352b@aurora ~ $ time cat ulysses.txt | ./mapper1.py | ./shuffleSort.py | ./reducer1.py
| ./mapper2.py | sort -n | ./reducer2.py
the,  14854

real  0m2.513s
user  0m2.417s
sys   0m0.077s


352b@aurora ~ $ time cat ulysses.txt | tr -c '[:alnum:]' '[\n*]' | sort | uniq -c | sort
-nr | head  -2
 100963
  13683 the

real  0m0.958s
user  0m0.931s
sys   0m0.019s
```

Sorting with Python

Sorting with Linux commands

http://unix.stackexchange.com/questions/41479/find-n-most-frequent-words-in-a-file

Exercise 3

- Compute **Pi** using Map-Reduce

Serial version

```
public class PiSerial {

        private static double f( double x ) {
                return 4.0 / ( 1 + x*x );
        }

        public static void main( String[] args ) {

                //--- syntax: java -jar PiSerial.jar N ---
                if ( args.length == 0 ) {
                        System.out.println( "Syntax: PiSerial N\nwhere N is the number of iterations\n\n" );
                        return;
                }
                int N = Integer.parseInt( args[0] );

                double sum = 0;
                double deltaX = 1.0/N;

                //--- iterate ---
                for ( int i=0; i<N; i++ )
                        sum += f( i * deltaX );

                System.out.println( N + " iterations.  Result = " + sum*deltaX + "\n\n" );
        }
}
```

```
352b@aurora ~/handout/MapReduce/Pi $ echo "1000" | ./mapper.py | sort | ./reducer.py
3.1425924850  0
```

Exercise 3

- Timing **Pi** using Map-Reduce vs Serial Python

```
352b@aurora ~ $ for i in 100000 do
    echo "N=$i"
    echo "----- Map-Reduce ------"
    time runIt.sh $i
    echo "----- Serial Python -----"
    time ./pi.py $i
done

N=100000

---- MapReduce -----
3.1416026569  0

real  0m1.099s
user  0m0.581s
sys   0m0.023s

----- Serial Python -----
3.1416026536

real  0m0.089s
user  0m0.037s
sys   0m0.006s
```

Exercise 4

- Implement **grep** using Map-Reduce

- *How should you feed **both** the expression to search for **and** the text to the mapper?*
- *What should the output format for mapper be?*
- *What should the output format for reducer be?*

Exercise 4

- Implement **grep** using Map-Reduce

```
352b@aurora ~ $ { echo Mulligan ; cat 4300-8.txt; } | ./mapper.py | sort | ./
reducer.py

     outputs all the lines from input file(s) that contain "Mulligan"
```

Exercise 4

- Implement **grep** using Map-Reduce

```
352b@aurora ~ $ { echo Mulligan ; cat ulysses.txt; } | ./mapper.py | sort | ./
reducer.py
mulligan  --Ah, go to God! Buck Mulligan said.
mulligan  Amused Buck Mulligan mused in pleasant murmur with himself,
selfnodding:
mulligan  --Are you not coming in? Buck Mulligan asked.
mulligan  --Are you up there, Mulligan?
…
mulligan  --Yes? Buck Mulligan said. What did I say? I forget.
mulligan  --Yes, Mulligan said. That's John Howard, his brother, our city
marshal.
mulligan  --Yes, what is it? Buck Mulligan answered. I don't remember anything.
```

last time…

Exercise 5

- Compute an **inverted index** of the words contained in several files.

Expected Output

```
accept   voltaire3.txt
alone    voltaire3.txt
also     voltaire2.txt
are      voltaire3.txt
…
to       voltaire1.txt,voltaire2.txt,voltaire3.txt
…
```

Exercise 5

- Compute an **inverted index** of the words contained in several files.

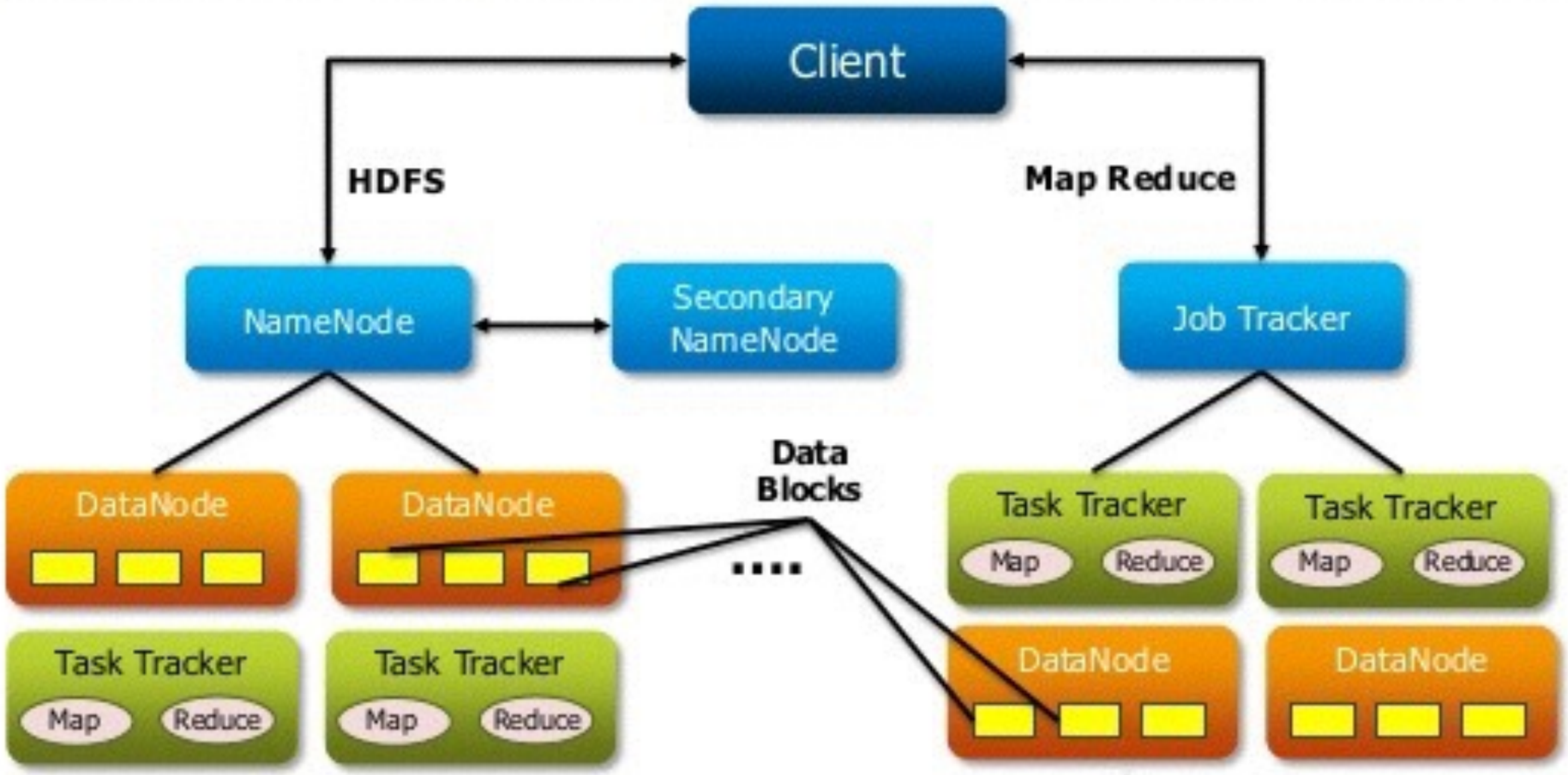
```
352b@aurora ~ $ for i in *.txt ; do
> { echo $i; cat $i; } | ./mapper.py
> done | sort | ./reducer.py

accept  voltaire3.txt
alone voltaire3.txt
also  voltaire2.txt
are   voltaire3.txt
…
to voltaire1.txt,voltaire2.txt,voltaire3.txt
…
```

Exercise 6

- Solve the 2D **Game of Life** problem using Map-Reduce.

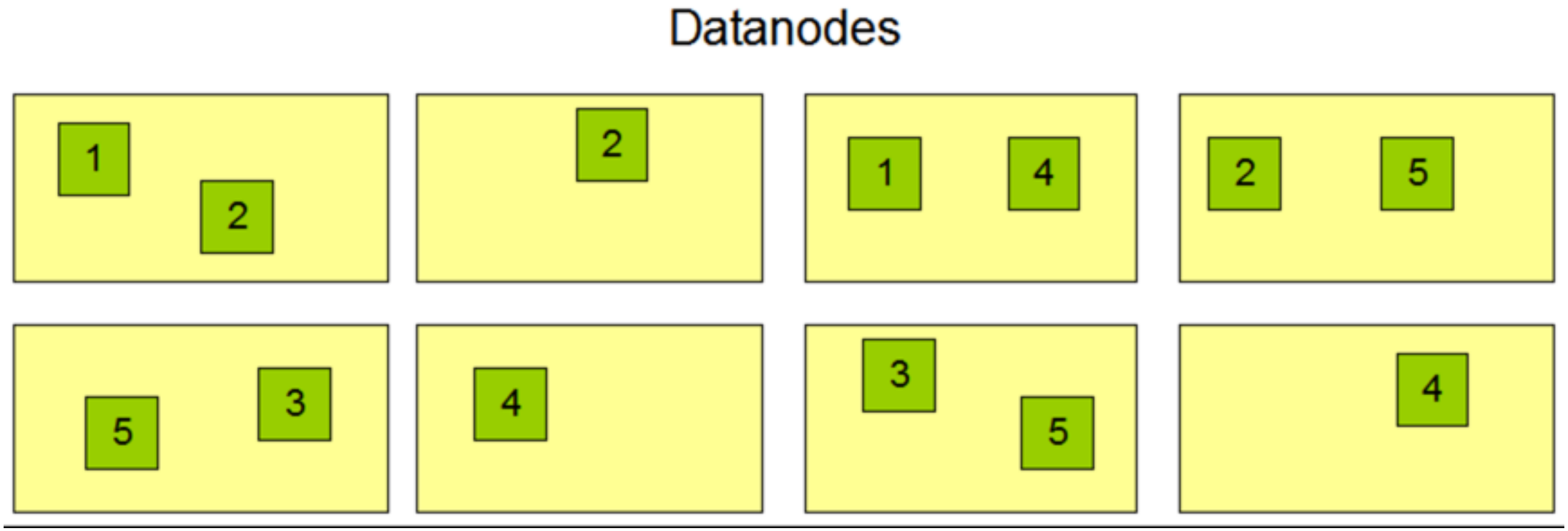
AWS Hadoop

Hadoop Infrastructure



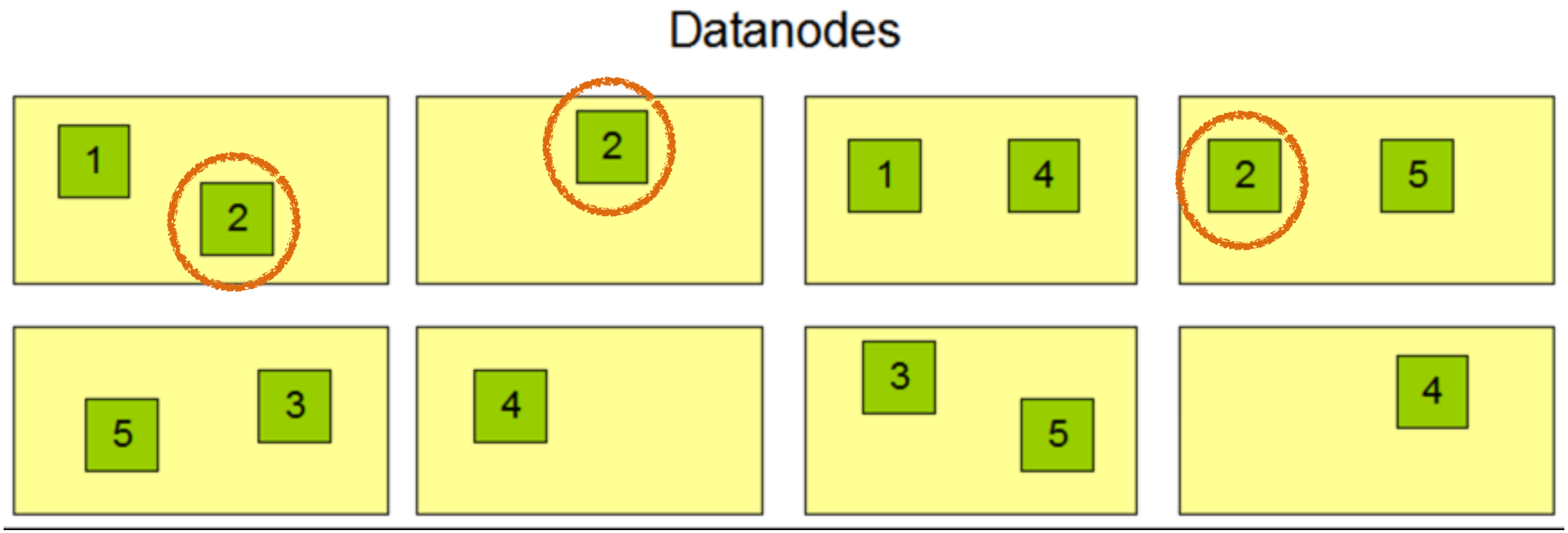
https://www.slideshare.net/EdurekaIN/hadoop-20-architecture-hdfs-federation-namenode-high-availability

DataNodes



https://hadoop.apache.org/docs/r1.2.1/images/hdfsdatanodes.gif

DataNodes



degree of replication: 3

https://hadoop.apache.org/docs/r1.2.1/images/hdfsdatanodes.gif

Self-Paced Labs

- Tutorial: Creating a Hadoop Cluster on AWS
  (http://www.science.smith.edu/dftwiki/index.php/Tutorial:_Creating_a_Hadoop_Cluster_on_Amazon_AWS)

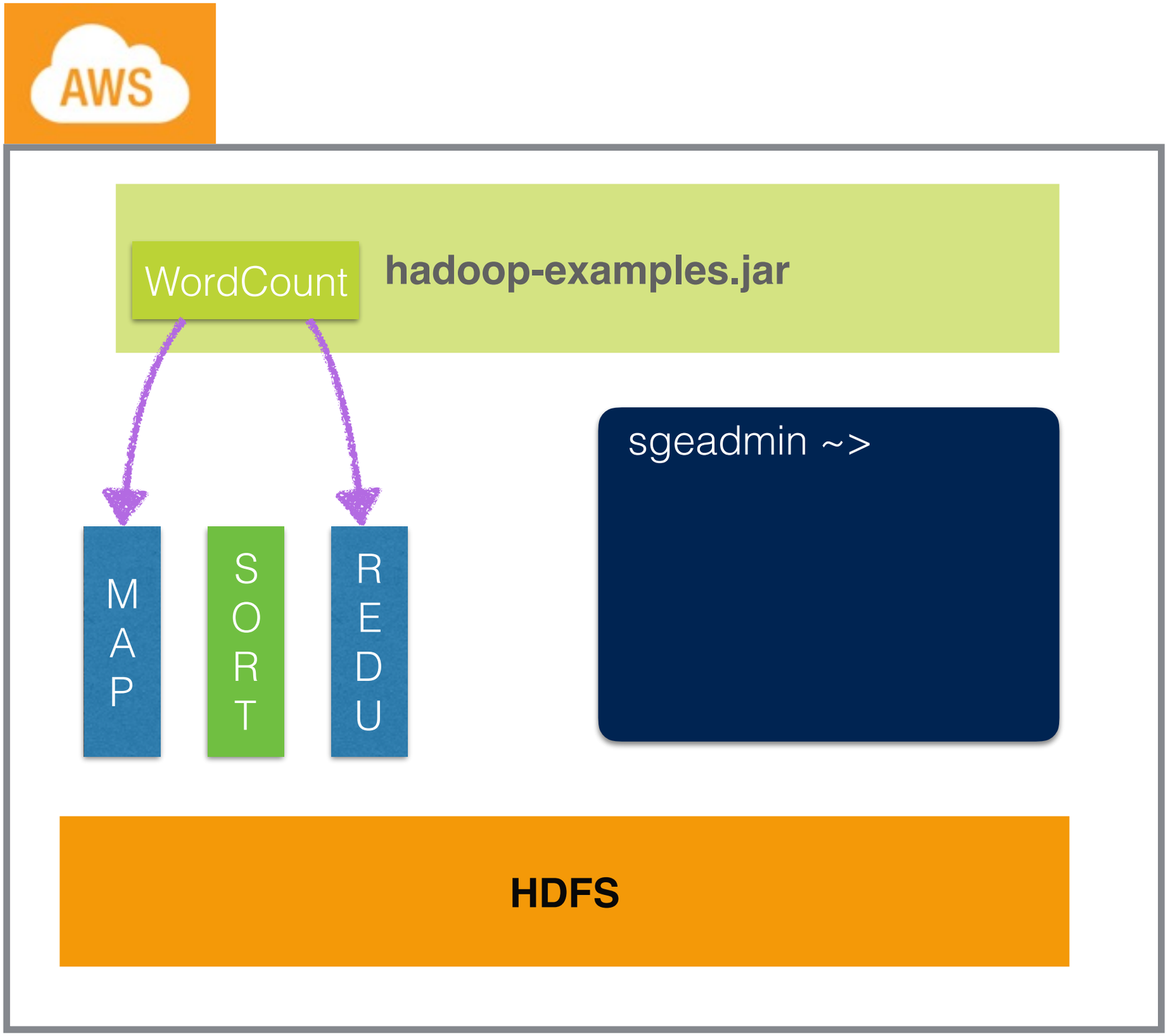
- Tutorial: Running WordCount in Python on AWS
  (http://www.science.smith.edu/dftwiki/index.php/Hadoop_Tutorial_2.3_--_Running_WordCount_in_Python_on_AWS)

- Tutorial: Creating A Task Graph
  (http://www.science.smith.edu/dftwiki/index.php/Hadoop_Tutorial_1.1_--_Generating_Task_Timelines)

Tutorial 1

STEP 1

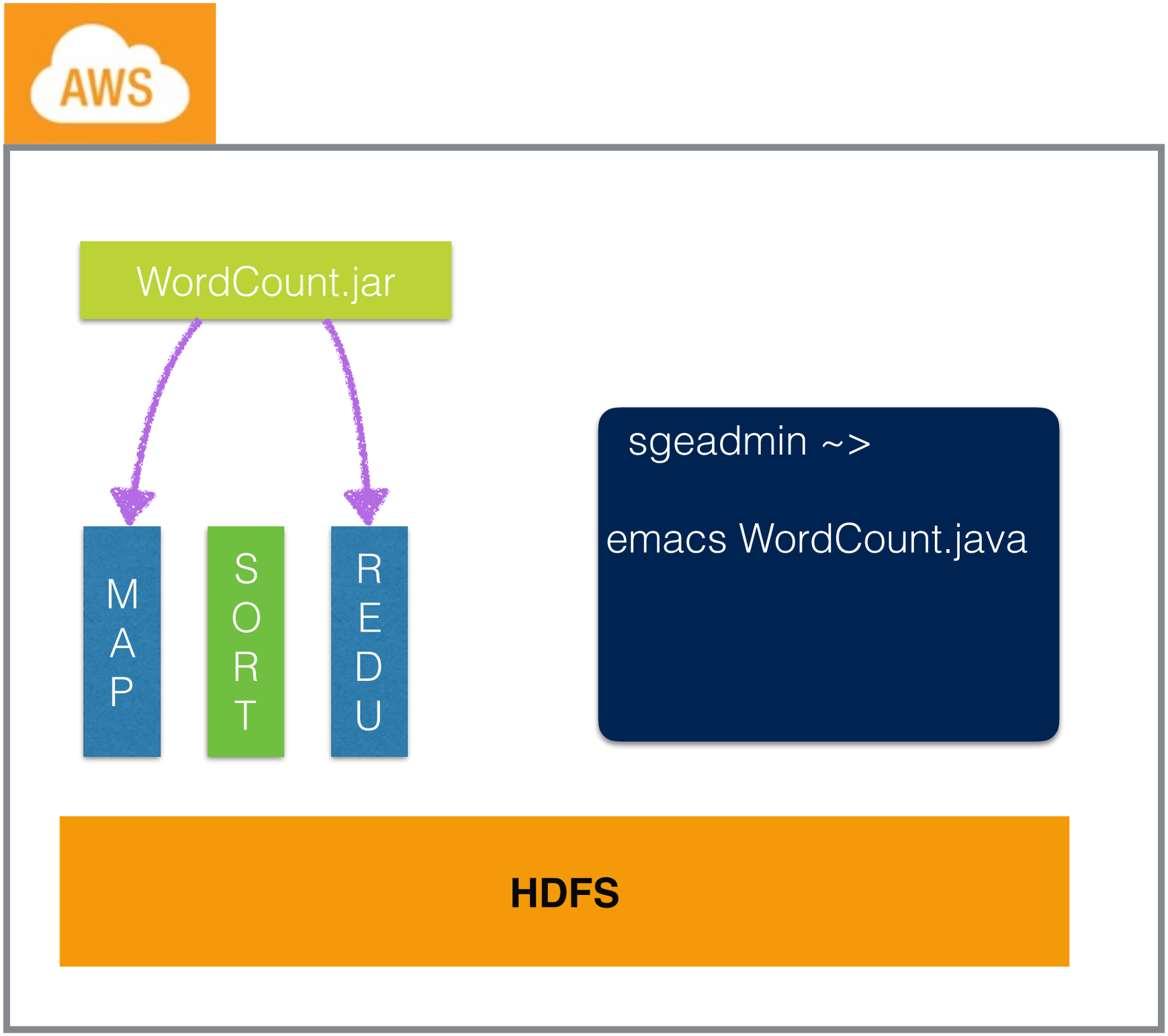
cs.smith.edu/~dthiebaut



Tutorial 1

STEP 2

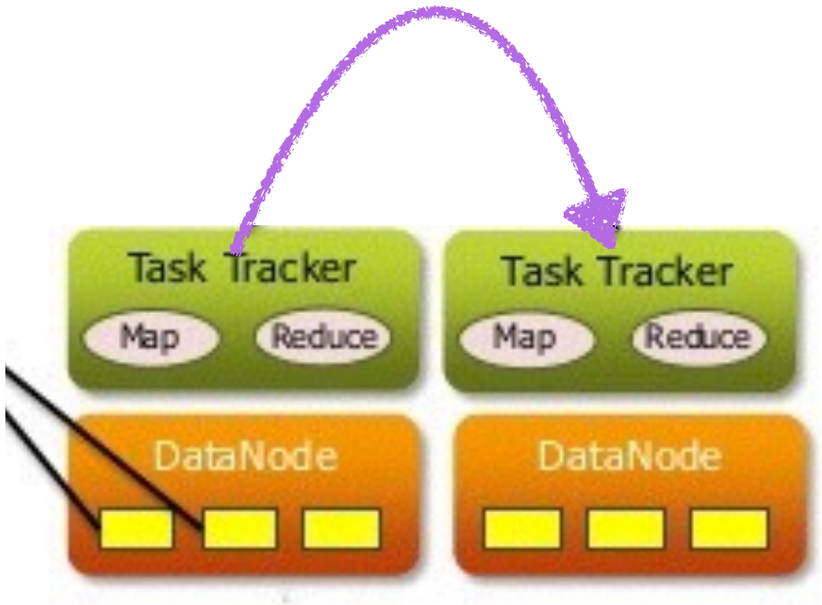
cs.smith.edu/~dthiebaut



Mapper

```
public class MyWordCount extends Configured implements Tool {
  /**
   * Counts the words in each line.
   * For each line of input, break the line into words and emit them as
   * (<b>word</b>, <b>1</b>).
   */
  public static class MapClass extends MapReduceBase
    implements 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,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }
}
```

Writables & WritableComparables



- Types that allow **serialization** and **deserialization** for **exchange over the network**: BooleanWritable, ByteWritable, IntWritable, VIntWritable, FloatWritable, LongWritable, VLongWritable, DoubleWritable, **Text**, ArrayWritable, TwoDArrayWritable

- Serialized data use **less storage**

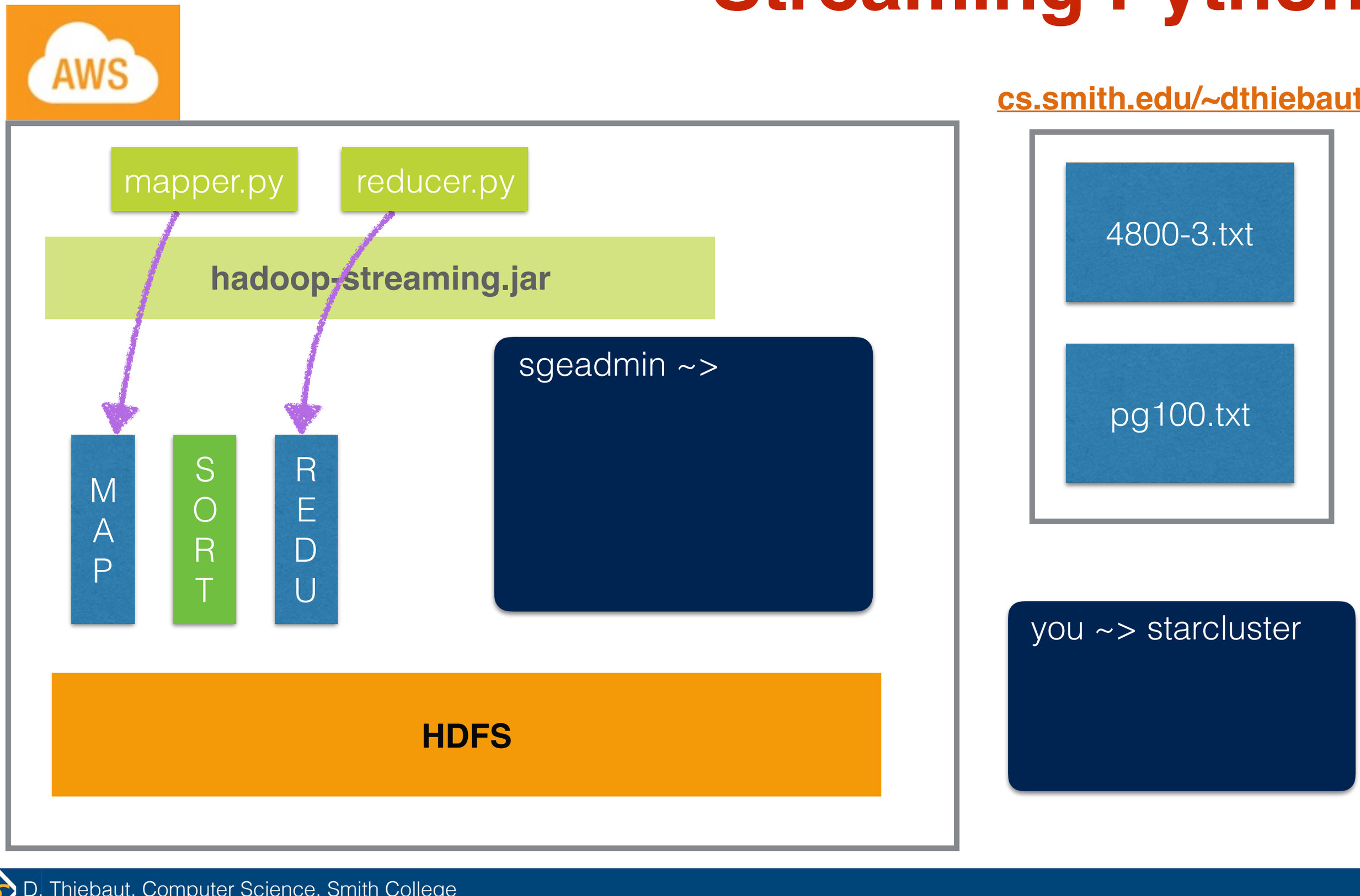
- **WritableComparables**: for non-standard types, so that they be compared to each other (sorted)

Reducer

```
/**
 * A reducer class that just emits the sum of the input values.
 */
public static class Reduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

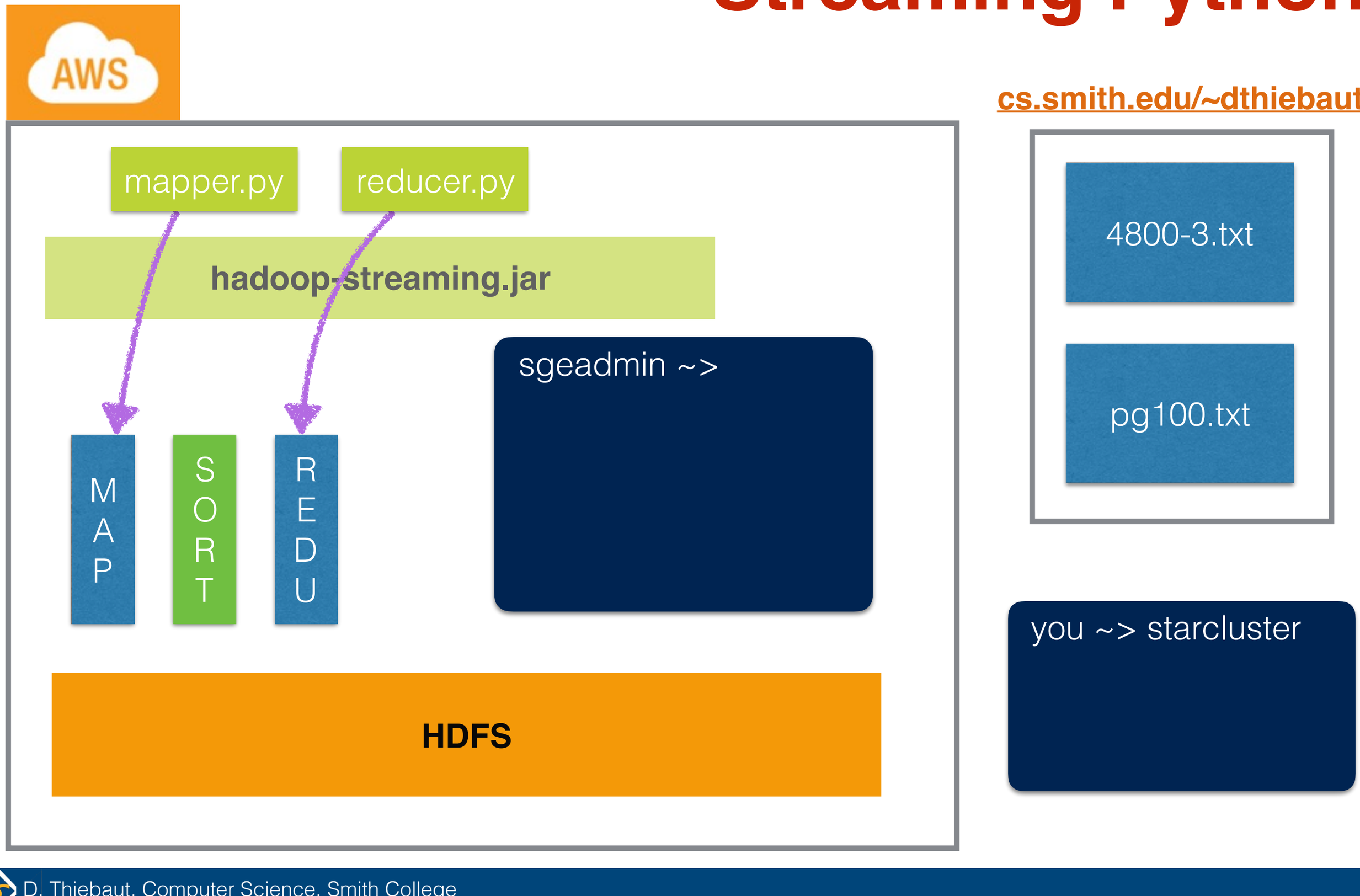
Tutorial 2: Streaming Python

cs.smith.edu/~dthiebaut



Tutorial 2: Streaming Python

cs.smith.edu/~dthiebaut



Mapper

```
#!/usr/bin/env python
# mapper.py
import sys

#--- get all lines from stdin ---
for line in sys.stdin:
    #--- remove leading and trailing whitespace---
    line = line.strip()

    #--- split the line into words ---
    words = line.split()

    #--- output tuples [word, 1] in tab-delimited format---
    for word in words:
        print '%s\t%s' % (word, "1")
```

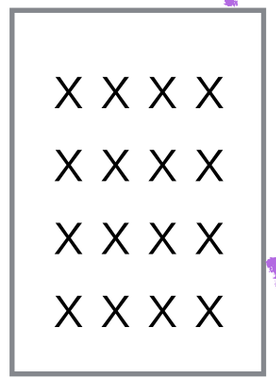
Reducer

```
#!/usr/bin/env python
# reducer.py
import sys

# maps words to their counts
word2count = {}

# 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:
        continue

    try:
        word2count[word] = word2count[word]+count
    except:
        word2count[word] = count

# write the tuples to stdout
# Note: they are unsorted
for word in word2count.keys():
    print '%s\t%s'% ( word, word2count[word] )
```

Tutorial 3


Number of Tasks as a Function of Time
Number of Tasks
Time (sec)
waste
reduce
merge
shuffle
maps


AWS


X X X X
X X X X
X X X X
X X X X

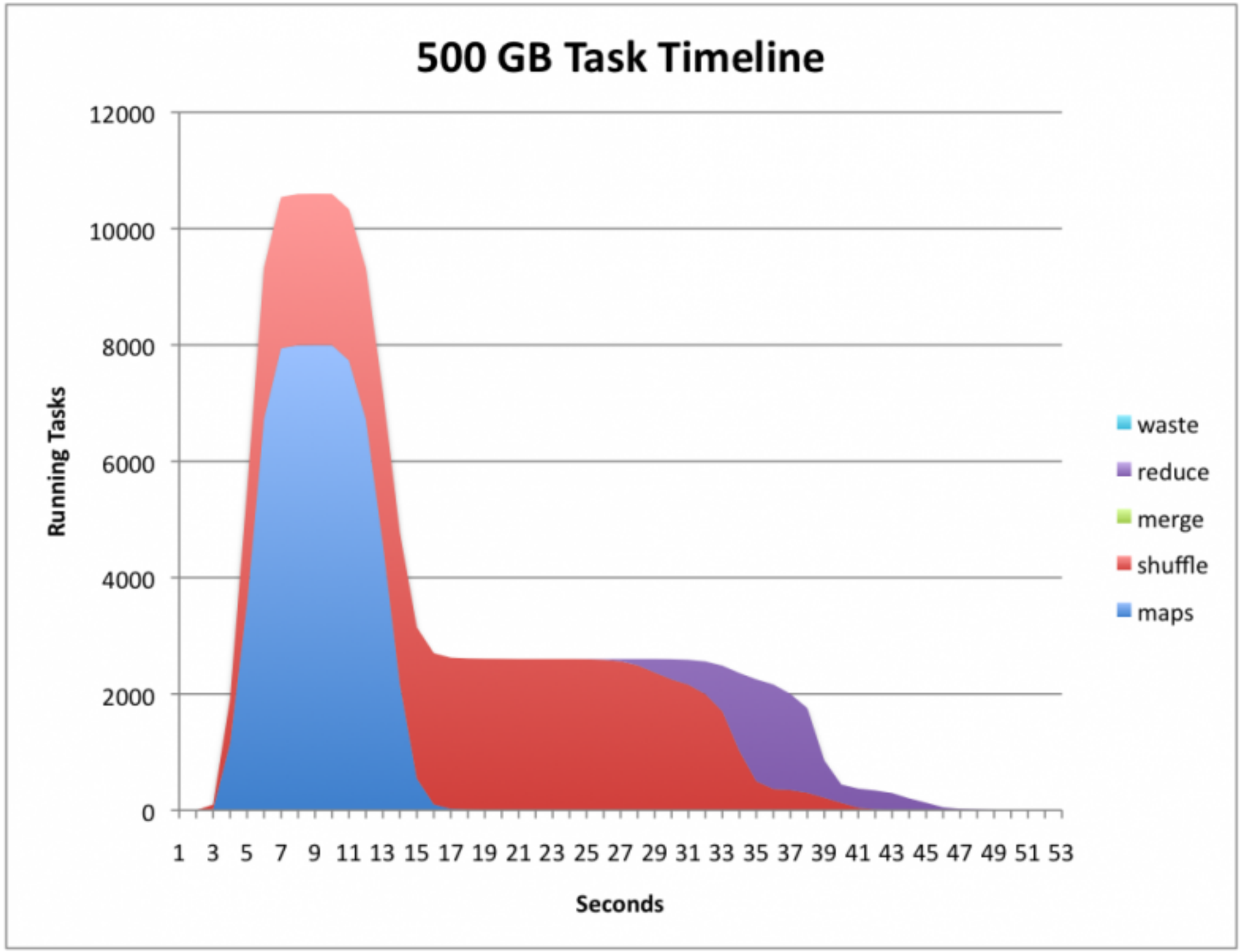

sgeadmin ~>


generateTimeLine.py


MAP
SORT
REDU


Logs


HDFS


500 GB Task Timeline
Running Tasks
12000
10000
8000
6000
4000
2000
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53
Seconds
waste
reduce
merge
shuffle
maps