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

Step 2: Shuffle

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

Step 3: Reducer

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

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
```

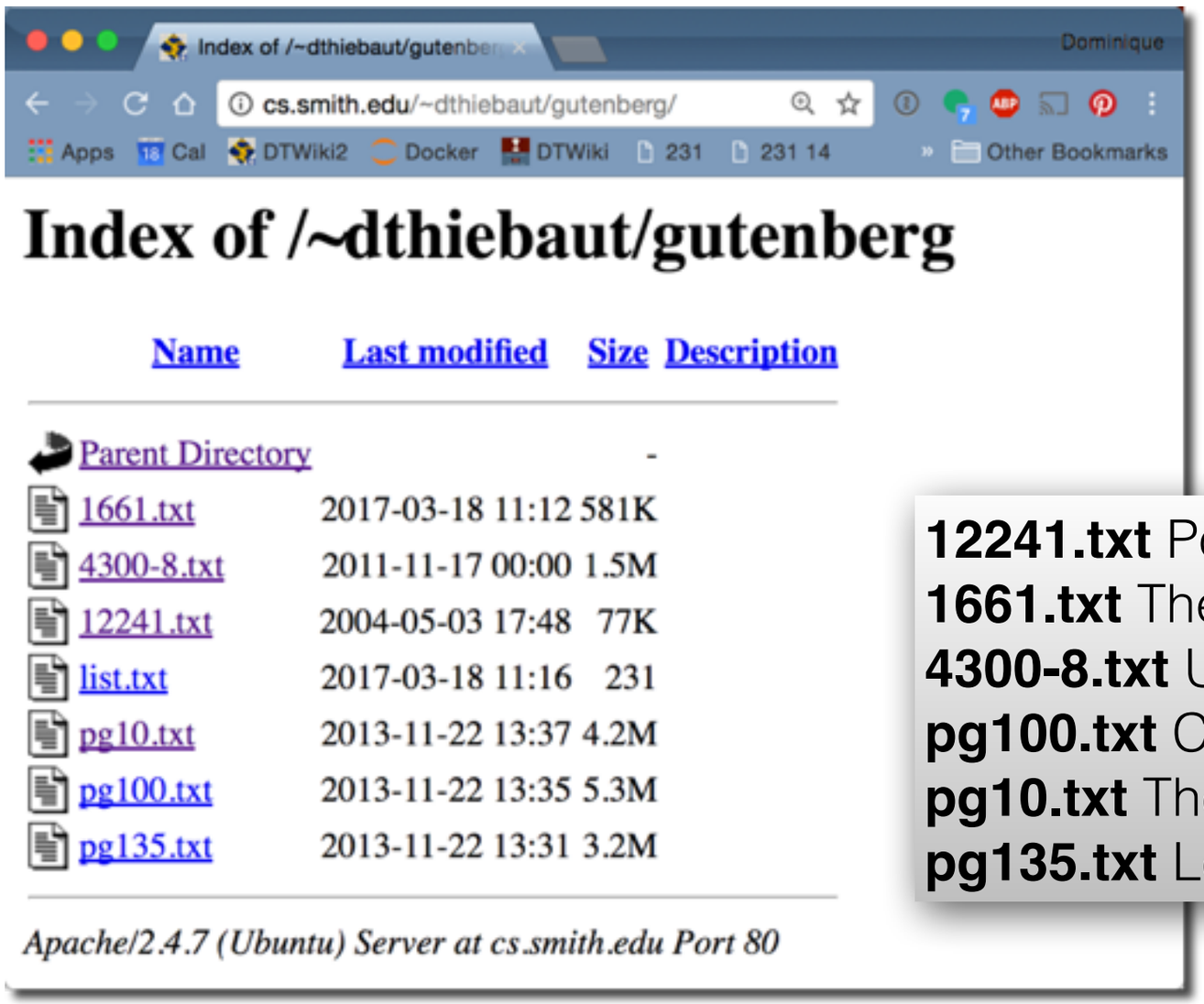
We can use Linux's sort command!

Exercise 1

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

Exercise 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
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 5

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

Expected Output

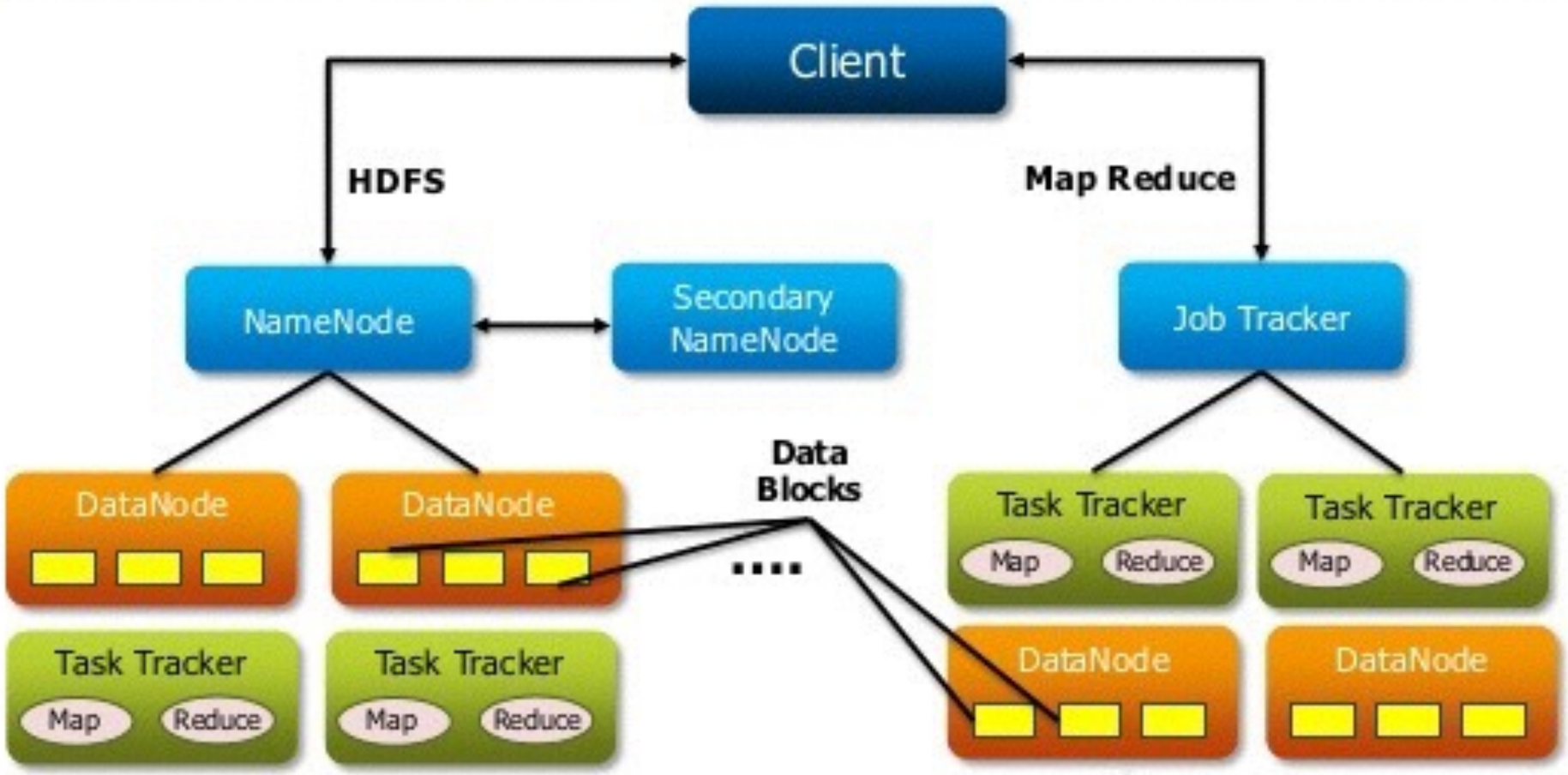
```
accept   voltaire3.txt
alone    voltaire3.txt
also     voltaire2.txt
are      voltaire3.txt
…
to       voltaire1.txt,voltaire2.txt,voltaire3.txt
…
```

Exercise 6

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

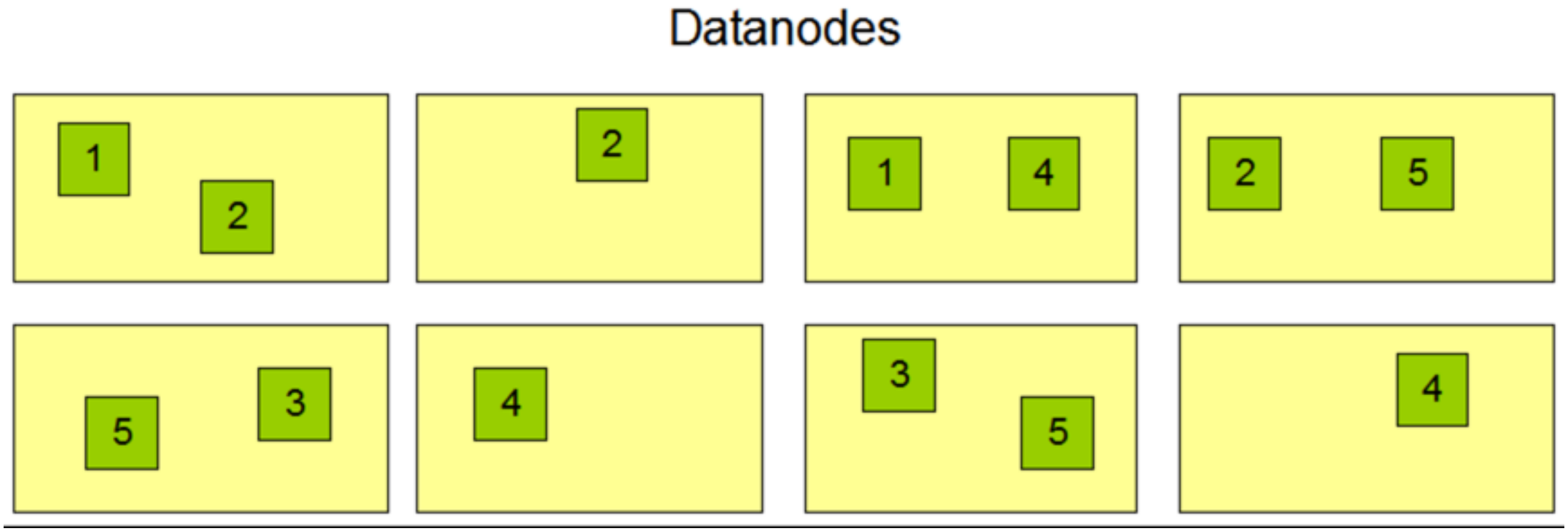
AWS Hadoop

Hadoop Infrastructure



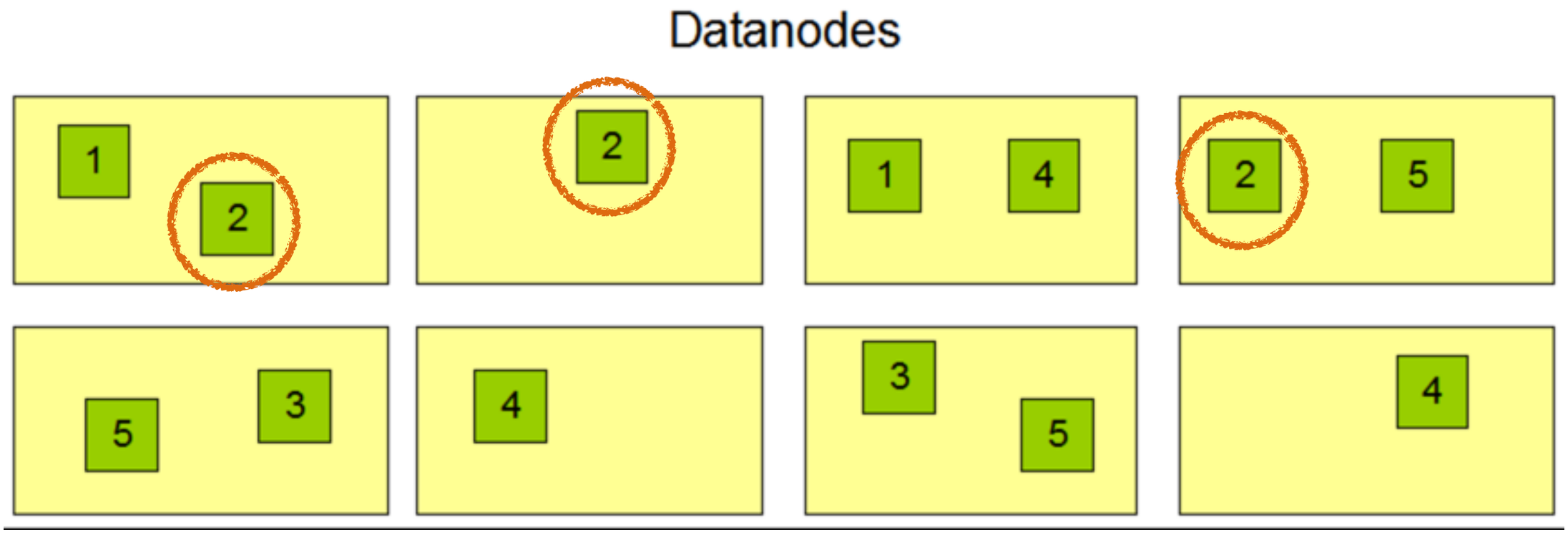
https://www.slideshare.net/EdurekaIN/hadoop-20-architecture-hdfs-federation-namenode-high-availability

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)