Introduction to MapReduce

Louis Jachiet

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A $\Rightarrow$

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A $\Rightarrow$

A	3
B	1
C	2
D	2

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A $\Rightarrow$

A	3
B	1
C	2
D	2

A very basic and classical problem in data mining!

e.g. anomaly / spam / bots detection

Example: Counting the occurences of each item in a list

Easily solved in Python:

Example: Counting the occurences of each item in a list

Easily solved in Python:

```
nb = dict()
for item in inputList:
  if not item in nb:
    nb[item] = 0
  nb[item] += 1
```

Example: Counting the occurences of each item in a list

Easily solved in Python:

```
nb = dict()
for item in inputList:
  if not item in nb:
    nb[item] = 0
  nb[item] += 1
```

How to make this program run on many machines?

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input
- Each machine will be responsible for some of the items

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input
- Each machine will be responsible for some of the items
- Items will be sent to the machine responsible for them

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input
- Each machine will be responsible for some of the items
- Items will be sent to the machine responsible for them

Machine A-B
A, B, D, A, C, C, D, A

Machine C-D
A, C, C, B, A, A, D, C

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input
- Each machine will be responsible for some of the items
- Items will be sent to the machine responsible for them

Machine A-B		Machine C-D
A, B, A, A	$\leftarrow$ A, B, A, A	C, C, D, C
	D, C, C, D $\rightarrow$	

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input
- Each machine will be responsible for some of the items
- Items will be sent to the machine responsible for them

Machine A-B
A, B, A, A, A, B, A, A

Machine C-D
C, C, D, C, D, C, C, D

Example: Counting the occurences of each item in a list

- Each machine will host a part of the input
- Each machine will be responsible for some of the items
- Items will be sent to the machine responsible for them

Machine A-B	
A: 6	B:2

Machine C-D	
C: 5	D:3

Example: Counting the occurences of each item in a list

What needs to be done for very large data?

- Partition data

Example: Counting the occurences of each item in a list

What needs to be done for very large data?

- Partition data
- Start computation

Example: Counting the occurences of each item in a list

What needs to be done for very large data?

- Partition data
- Start computation
- Shuffle data

Example: Counting the occurences of each item in a list

What needs to be done for very large data?

- Partition data
- Start computation
- Shuffle data
- Handle failure (data + computation)

Very hard to get right!

The MapReduce Model

Typical Big Data Problem

A typical Big Data problem can be divided into 5 phases

1. Iterate over a large number of records
2. Extract something of interest from each
3. Shuffle and sort intermediate results
4. Aggregate intermediate results
5. Generate final output

Typical Big Data Problem

A typical Big Data problem can be divided into 5 phases

1. Iterate over a large number of records
2. Extract something of interest from each –MAP–
3. Shuffle and sort intermediate results
4. Aggregate intermediate results –REDUCE–
5. Generate final output

MapReduce Model

For a MapReduce job, the programmer needs to provide:

- A MAP function: `value` $\rightarrow$ `(key,value)`

  Transforms each record into a (possibly empty) list of key-value pairs

MapReduce Model

For a MapReduce job, the programmer needs to provide:

- A MAP function: `value` $\rightarrow$ `(key,value)`

  Transforms each record into a (possibly empty) list of key-value pairs

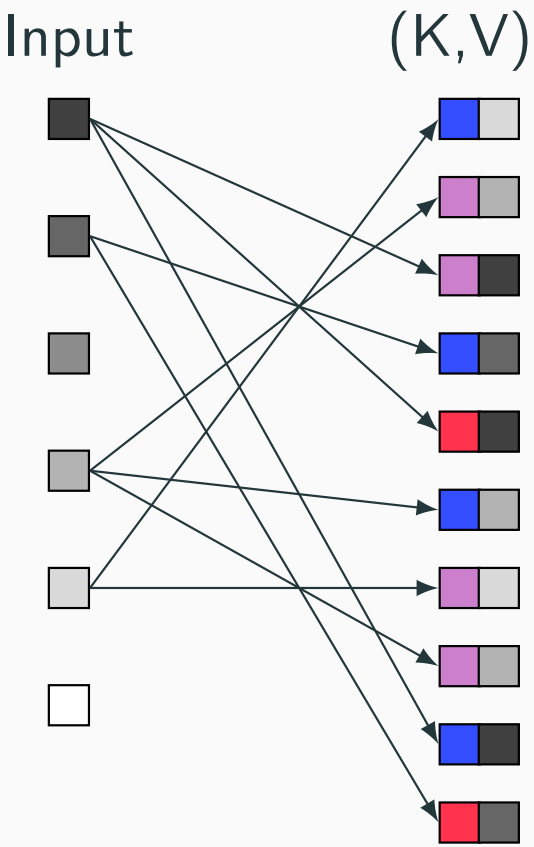
- A REDUCE function: `(key, list of values)` $\rightarrow$ `value`

  Take a key and the list of values with this key

MapReduce Model

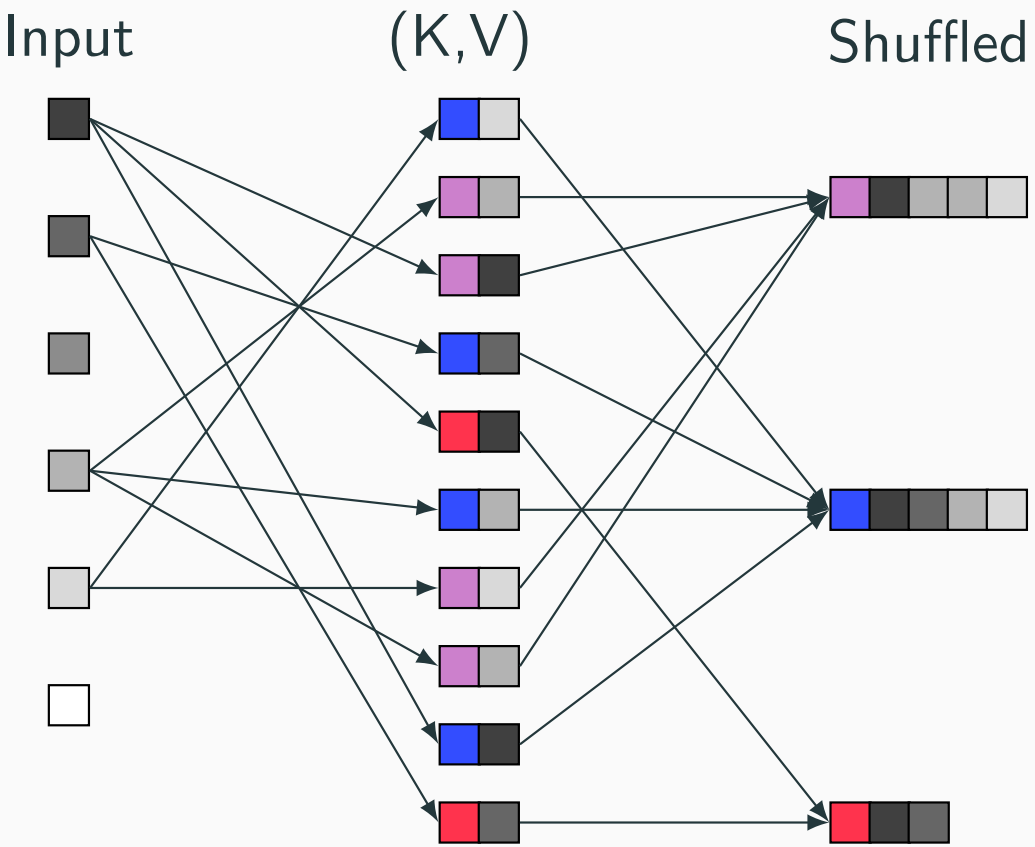
Input

MapReduce Model



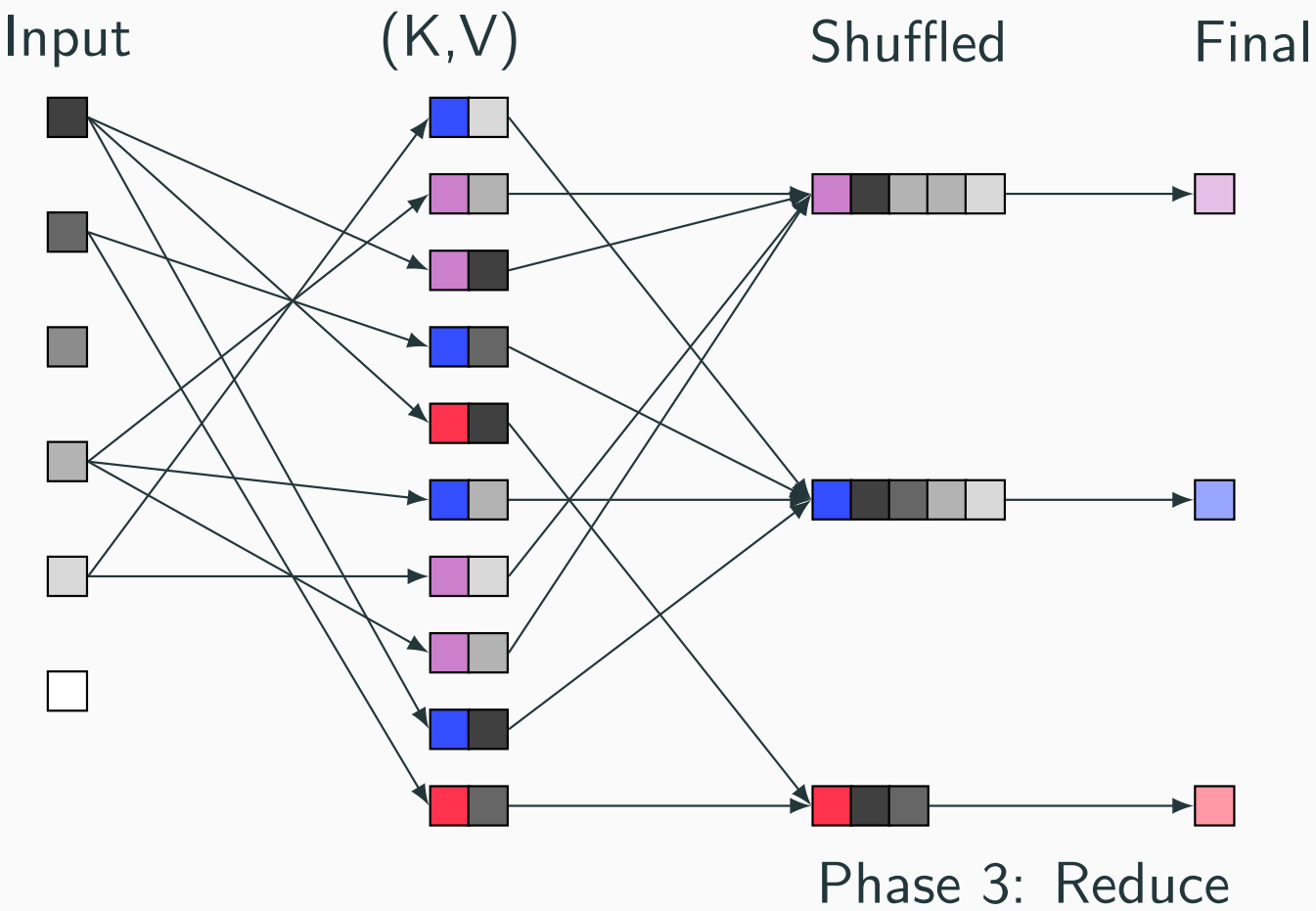
Phase 1: Map

MapReduce Model



Phase 2: Shuffle

MapReduce Model



Example: Counting the occurences of each item in a list

MAP

Each item i is transformed into the singleton list key-value pair $[(i, 1)]$

REDUCE

Given a pair (i, l) where i is an item and l a list, the reducer returns $(i, \texttt{length}(l))$

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$\downarrow$ SHUFFLE $\downarrow$

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$\downarrow$ SHUFFLE $\downarrow$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$\downarrow$ SHUFFLE $\downarrow$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

Example: Counting the occurences of each item in a list

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$\downarrow$ SHUFFLE $\downarrow$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,3), (B,1), (D,2), (C,2)

MapReduce

It is possible to chain several MapReduce jobs!

MapReduce

It is possible to chain several MapReduce jobs!

It is possible to use several inputs for a MapReduce job!

Example: Keep distinct items appearing more than twice

A, B, D, A, C, C, D, A

$\downarrow$ MAP: $i \rightarrow [(i, 1)]$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$\downarrow$ SHUFFLE $\downarrow$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,3), (B,1), (D,2), (C,2)

Example: Keep distinct items appearing more than twice

$$\downarrow \text{MAP: } i \rightarrow [(i, 1)] \downarrow$$

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$$\downarrow \text{SHUFFLE} \downarrow$$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$$\downarrow \text{REDUCE: } (i, l) \rightarrow (i, \mathit{length}(l)) \downarrow$$

(A,3), (B,1), (D,2), (C,2)

$$\downarrow \text{MAP: } (i, n) \rightarrow \begin{cases} [] & \text{when } n < 2 \\ [(i, n)] & \text{otherwise} \end{cases} \downarrow$$

Example: Keep distinct items appearing more than twice

(A,1), (B,1), (D,1), (A,1), (C,1), (C,1), (D,1), (A,1)

$\downarrow$ SHUFFLE $\downarrow$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,3), (B,1), (D,2), (C,2)

$$\downarrow \text{ MAP: } (i, n) \rightarrow \begin{cases} [] & \text{when } n < 2 \\ [(i, n)] & \text{otherwise} \end{cases} \downarrow$$

(A,3), (D,2), (C,2)

Example: Keep distinct items appearing more than twice

$\downarrow$ SHUFFLE $\downarrow$

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,3), (B,1), (D,2), (C,2)

$$\downarrow \text{ MAP: } (i, n) \rightarrow \begin{cases} [] & \text{when } n < 2 \\ [(i, n)] & \text{otherwise} \end{cases} \downarrow$$

(A,3), (D,2), (C,2)

$\downarrow$ SHUFFLE $\downarrow$

Example: Keep distinct items appearing more than twice

(A,[1,1,1]), (B,[1]), (D,[1,1]), (C,[1,1])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,3), (B,1), (D,2), (C,2)

$$\downarrow \text{ MAP: } (i, n) \rightarrow \begin{cases} [] & \text{when } n < 2 \\ [(i, n)] & \text{otherwise} \end{cases} \downarrow$$

(A,3), (D,2), (C,2)

$\downarrow$ SHUFFLE $\downarrow$

(A,3), (D,2), (C,2)

Example: Keep distinct items appearing more than twice

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,3), (B,1), (D,2), (C,2)

$$\downarrow \text{MAP: } (i, n) \rightarrow \begin{cases} [] & \text{when } n < 2 \\ [(i, n)] & \text{otherwise} \end{cases} \downarrow$$

(A,3), (D,2), (C,2)

$\downarrow$ SHUFFLE $\downarrow$

(A,3), (D,2), (C,2)

$\downarrow$ REDUCE: $(i, l) \rightarrow i$ $\downarrow$

Example: Keep distinct items appearing more than twice

(A,3), (B,1), (D,2), (C,2)

$$\downarrow \text{MAP: } (i, n) \rightarrow \begin{cases} [] & \text{when } n < 2 \\ [(i, n)] & \text{otherwise} \end{cases} \downarrow$$

(A,3), (D,2), (C,2)

$\downarrow$ SHUFFLE $\downarrow$

(A,3), (D,2), (C,2)

$\downarrow$ REDUCE: $(i, l) \rightarrow i$ $\downarrow$

A, D, C

Exercise (easy)

Input

You are given a list of pairs (k_i, v_i) where k_i is a string and v_i an integer.

Problem

Compute the average value for each key.

Example

INPUT	
A	42
B	17
A	12
B	99

OUTPUT	
A	$\frac{42+12}{2} = 27$
B	$\frac{17+99}{2} = 58$

Exercise (medium)

Input

You are given two lists of items.

Problem

Compute the list of item appearing in the first one but not in the second.

Example

INPUT 1
A
B
C

INPUT2
A
C
E

OUTPUT
B

Exercise (hard)

Input

You are given the Twitter following list: each record is a pair (A_i, B_i) indicating that account A_i follows B_i.

Problem

Compute the accounts that have more followers than all the accounts that they follow.

Example

INPUT	
A	B
A	D
B	C
B	D
C	E

OUTPUT
E
D
C

Exercise (hardest)

Input

You are given the Twitter following list: each record is a pair (A_i, L_i) indicating that account A_i follows the accounts in the list L_i .

Problem

Compute for each account A the list of accounts that are followed by an account followed by A.

Example

INPUT	
A	B,D
B	C,D
C	E

OUTPUT	
A	C,D
B	E

MapReduce Beyond Map and Reduce

MapReduce extensions: Combiner

Context

Counting the number of times each item appears.

Problem

A few set of words appear very often.

MapReduce extensions: Combiner

Context

Counting the number of times each item appears.

Problem

A few set of words appear very often.

Solution: Combiner

A Combiner is similar to a reduce phase but applied before the shuffle on each local output of mappers.

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

$\downarrow$ COMBINE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

$\downarrow$ COMBINE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,2), (B,1), (D,1), **(C,2), (D,1), (A,1)**

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

↓ MAP: $i \rightarrow (i, 1)$ ↓

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

↓ COMBINE: $(i, l) \rightarrow (i, length(l))$ ↓

(A,2), (B,1), (D,1), **(C,2), (D,1), (A,1)**

↓ SHUFFLE ↓

Example: Counting the occurences of each item in a list

A, B, D, A, **C, C, D, A**

↓ MAP: $i \rightarrow (i, 1)$ ↓

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

↓ COMBINE: $(i, l) \rightarrow (i, length(l))$ ↓

(A,2), (B,1), (D,1), **(C,2), (D,1), (A,1)**

↓ SHUFFLE ↓

(A,[2,1]), (B,[1]), (D,[1,1]), (C,[2])

Example: Counting the occurences of each item in a list

$\downarrow$ MAP: $i \rightarrow (i, 1)$ $\downarrow$

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

$\downarrow$ COMBINE: $(i, l) \rightarrow (i, length(l))$ $\downarrow$

(A,2), (B,1), (D,1), **(C,2), (D,1), (A,1)**

$\downarrow$ SHUFFLE $\downarrow$

(A,[2,1]), (B,[1]), (D,[1,1]), (C,[2])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, \Sigma l)$ $\downarrow$

Example: Counting the occurences of each item in a list

(A,1), (B,1), (D,1), (A,1), **(C,1), (C,1), (D,1), (A,1)**

$\downarrow$ COMBINE: $(i, l) \rightarrow (i, length(l)) \downarrow$

(A,2), (B,1), (D,1), **(C,2), (D,1), (A,1)**

$\downarrow$ SHUFFLE $\downarrow$

(A,[2,1]), (B,[1]), (D,[1,1]), (C,[2])

$\downarrow$ REDUCE: $(i, l) \rightarrow (i, \Sigma l) \downarrow$

(A,3), (B,1), (D,2), (C,2)

Exercise on combiner

Input

You are given a list of pairs (k_i, v_i) where k_i is a string and v_i an integer.

Problem

Compute the average value for each key.

Example

INPUT	
A	42
B	17
A	12
B	99

OUTPUT	
A	$\frac{42+12}{2} = 27$
B	$\frac{17+99}{2} = 58$

MapReduce extensions: Partitioner

In MapReduce, each reducer task is responsible for a subset of the keys. Deciding which reducer is in charge of what key is the job of the Partitioner.

MapReduce extensions: Partitioner

In MapReduce, each reducer task is responsible for a subset of the keys. Deciding which reducer is in charge of what key is the job of the Partitioner.

By default, keys are assigned to reducers pseudo-randomly using a hash function.

$$K \rightarrow hash(K)\%nbReducers$$

MapReduce extensions: Partitioner

In MapReduce, each reducer task is responsible for a subset of the keys. Deciding which reducer is in charge of what key is the job of the Partitioner.

By default, keys are assigned to reducers pseudo-randomly using a hash function.

$$K \rightarrow hash(K)\%nbReducers$$

You can manually control the partitioning.

MapReduce extensions: Partitioner

```
public class HashPartitioner<K2, V2>
             implements Partitioner<K2, V2> {

    public void configure(JobConf job) {}
    /** Use {@link Object#hashCode()} to partition. */

    public int getPartition(K2 key, V2 value,
                            int numReduceTasks) {
      return (key.hashCode() & Integer.MAX_VALUE)
                               % numReduceTasks;
    }
}
```

Figure 1: Default Partitioner in Hadoop

HADOOP Implementation of MapReduce

Storage

- Data is stored in plain text files that are split into chunks

Chunks are typically 64 or 128 MB

Storage

- Data is stored in plain text files that are split into chunks

  Chunks are typically 64 or 128 MB

- Each chunk is replicated

  For resiliency, the replication factor is 3

Storage

- Data is stored in plain text files that are split into chunks

  Chunks are typically 64 or 128 MB

- Each chunk is replicated

  For resiliency, the replication factor is 3

- The Namenode handles where the chunks of files are stored

Storage

- Data is stored in plain text files that are split into chunks

  Chunks are typically 64 or 128 MB

- Each chunk is replicated

  For resiliency, the replication factor is 3

- The Namenode handles where the chunks of files are stored

- A Secondary Namenode handles resiliency

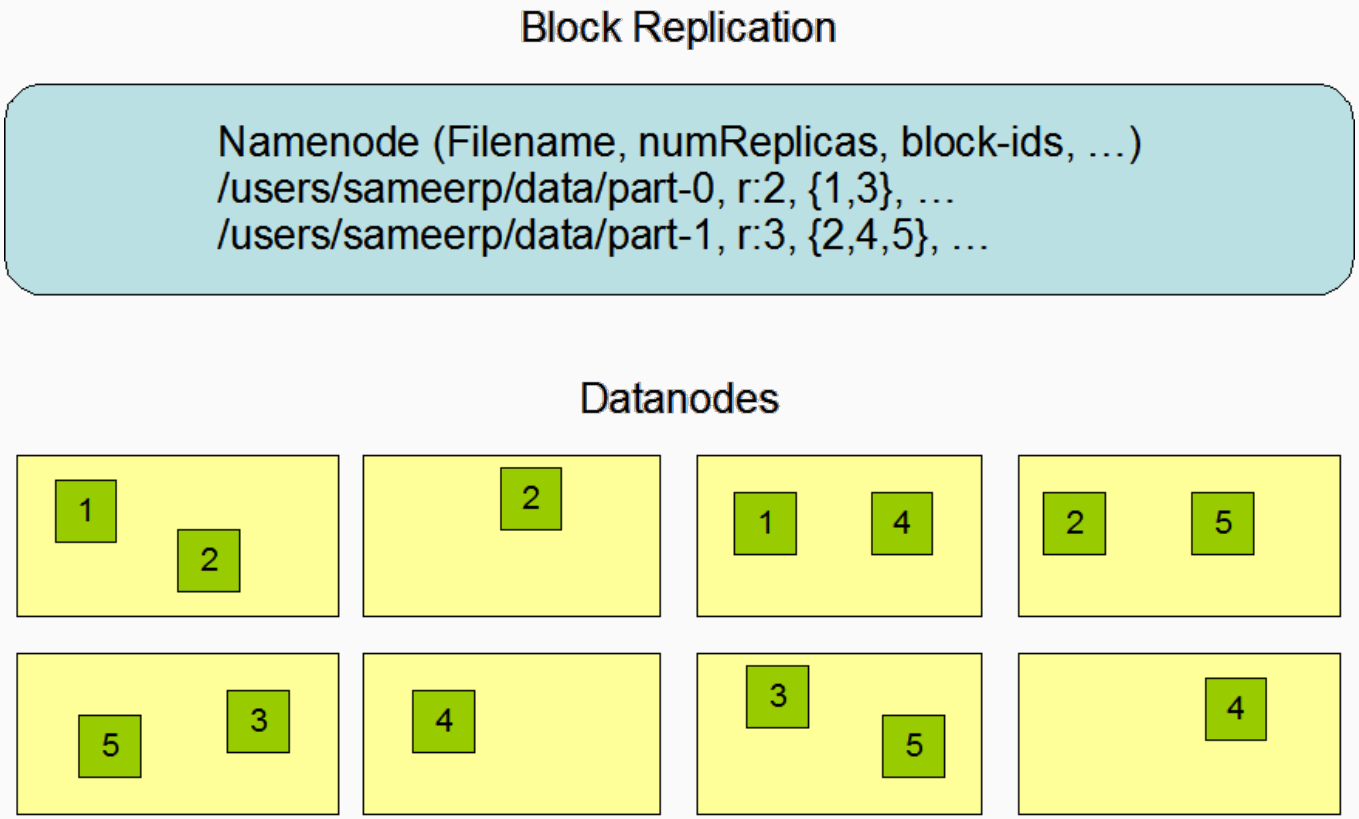


Figure 2: source

`https://hadoop.apache.org/docs/r1.2.1/hdfs_design`

Storage

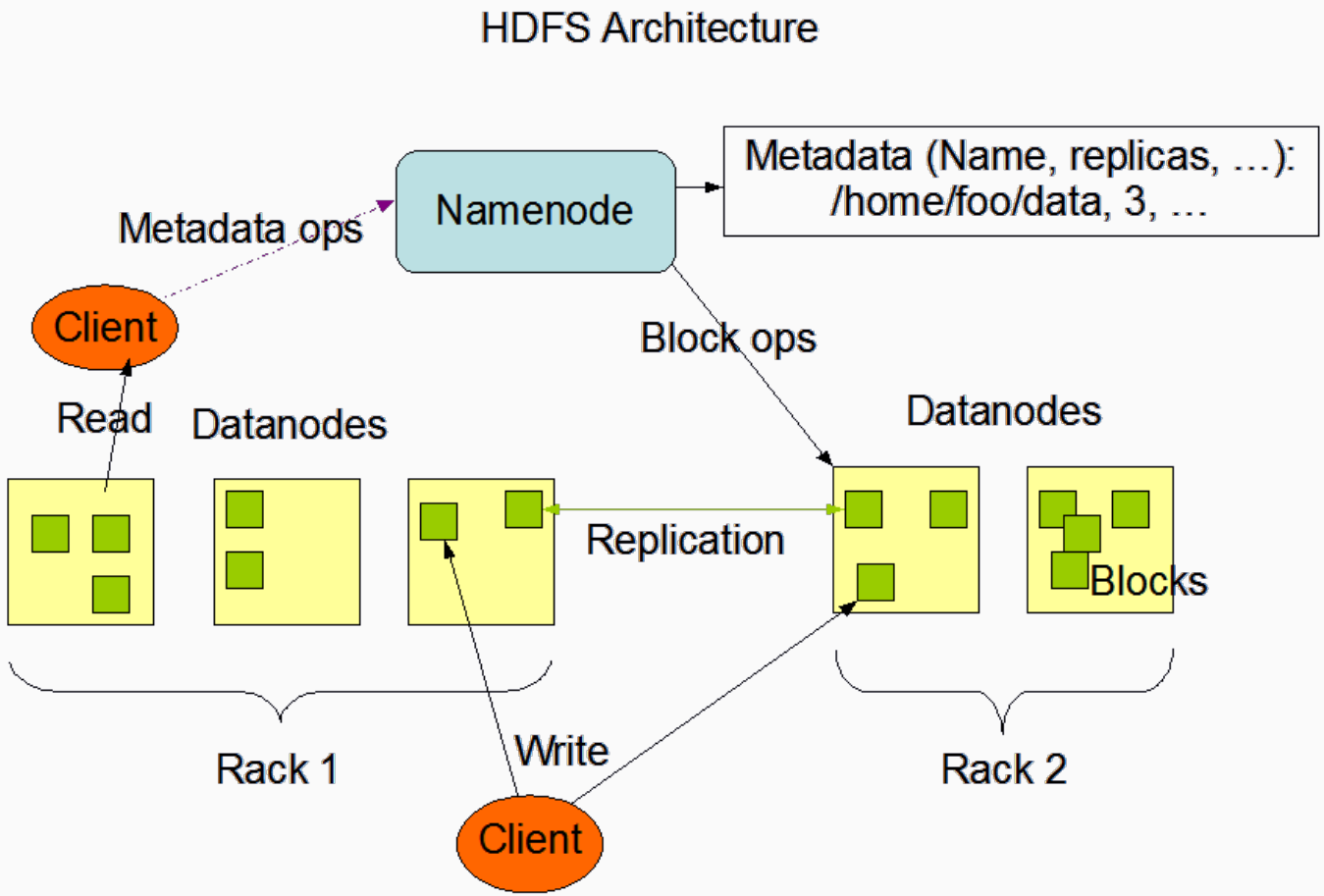


Figure 3: source
`https://hadoop.apache.org/docs/r1.2.1/hdfs_design`

Computation

- The Ressource Manager will start a number M of mapper tasks to process chunks

Mapper tasks are allocated where the data is!

Computation

- The Ressource Manager will start a number M of mapper tasks to process chunks

  Mapper tasks are allocated where the data is!

- The Ressource Manager will also start a number R of reducers tasks

Computation

- The Ressource Manager will start a number M of mapper tasks to process chunks

  Mapper tasks are allocated where the data is!

- The Ressource Manager will also start a number R of reducers tasks

- The outputs of mappers are sent to the reducers tasks according to a Partitioner

Computation

- The Ressource Manager will start a number M of mapper tasks to process chunks

  Mapper tasks are allocated where the data is!

- The Ressource Manager will also start a number R of reducers tasks

- The outputs of mappers are sent to the reducers tasks according to a Partitioner

Practical Hadoop

Hadoop

Hadoop is OPEN SOURCE and developped in JAVA

Hadoop

Hadoop is OPEN SOURCE and developped in JAVA

Submitting a MapReduce jobs is usually done by submitting a JAR file. The code will control and launch one or several MapReduce jobs.

Hadoop

Hadoop is OPEN SOURCE and developped in JAVA

Submitting a MapReduce jobs is usually done by submitting a JAR file. The code will control and launch one or several MapReduce jobs.

DEMO

Hadoop Streaming

Despite the name, Hadoop Streaming is for use of arbitrary binary. In particular it allows to use MapReduce with any programming language.

Hadoop Streaming

Despite the name, Hadoop Streaming is for use of arbitrary binary. In particular it allows to use MapReduce with any programming language.

The input is read on the standard input, with one record per line, the separator between key and values is a tabulation.

Hadoop Streaming

Despite the name, Hadoop Streaming is for use of arbitrary binary. In particular it allows to use MapReduce with any programming language.

The input is read on the standard input, with one record per line, the separator between key and values is a tabulation.

DEMO

Installing Hadoop

Installing Hadoop 1/3

Requirement

- Optional: a VM for hosting hadoop
- Java
- Create an account that can `ssh localhost` with ssh keys

Get Hadoop

- Download Hadoop 3.2
  `https://www.apache.org/dyn/closer.cgi/hadoop/core`
- Unzip it (e.g. in a Hadoop folder)

Installing Hadoop 2/3

Modify `.bashrc` or `.profile` of the account:

```
export HADOOP_HOME=/path/to/hadoop/folder
export JAVA_HOME=/usr/lib/jvm/default-runtime
alias hls=''fs -ls''
export PATH=$PATH:$HADOOP_HOME/bin
```

Adapt values to your configuration!

Launch Hadoop single node

```
hdfs namenode -format #initialize namenode
cd $HADOOP_HOME/sbin
bash start_all.sh
#don't forget to bash stop_all.sh at the end!
hadoop fs -mkdir /data /out
```

Installing Hadoop 3/3

Putting data on Hadoop:

```
hadoop fs -copyFromLocal /path/to/local /data/filename
```

Getting data from Hadoop:

```
hadoop fs -copyToLocal /data/filename  /path/to/local
```

Explore data on Hadoop:

```
hadoop fs -cat /data/filename
hadoop fs -head /data/filename
hadoop fs -tail /data/filename
```

Hands On: exploring BAN

Problem

The BAN contains addresses and we want to find out the most popular street names.

BAN csv files

What the file look like

```
hadoop fs -head /datasets/ban/ban-01.csv
```

```
id_ban_position;id_ban_adresse;cle_interop;id_ban_group;id_fantoi
```

BAN csv files

```
hadoop fs -head /datasets/ban/ban-01.csv | tr ';' '\n' | nl
     1              id_ban_position
     2        id_ban_adresse
     3        cle_interop
     4        id_ban_group
     5        id_fantoir
     6        numero
     7        suffixe
     8        nom_voie
     9        code_postal
    10        nom_commune
    11        code_insee
    12        nom_complementaire
    13        x
    14        y
    15        lon
    16        lat
    17        typ_loc
```

Counting street name

What is a unique street?

Each street appears multiple times

```
hadoop fs -cat /datasets/ban/ban-75.csv | grep -i barrault
    ban-position-f7b26917d0e7483b81140c9c3abea54e;
    ban-housenumber-927804e4ca734591a257d1fc91771c02;75113_0679_93;
    ban-group-d94295ec4c1a409a893c3aaa21ebcca6;751130679;93;;
    Rue Barrault;75013;Paris 13e Arrondissement;75113;;
    652047.483621478;6858292.61265954;2.346877;48.822911;
    entrance;ign;2018-10-21

    ban-position-cfe9a6c883184b448930656d2a021862;
    ban-housenumber-927804e4ca734591a257d1fc91771c02;75113_0679_93;
    ban-group-d94295ec4c1a409a893c3aaa21ebcca6;751130679;93;;
    Rue Barrault;75013;Paris 13e Arrondissement;75113;;
    652054.959452335;6858290.99409157;2.346979;48.822897;
    parcel;dgfip;2018-10-21

    ban-position-411cce6058474e53bb95ee3aea3466fd;
    ban-housenumber-92d8c6bc6d5c433eb1ed5c310177db3c;75113_0679_94;
```

Counting street name

What is a unique street?

A street is a name and a zip code.

BAN csv files

Columns 8 and 9!

```
 1            id_ban_position
 2        id_ban_adresse
 3        cle_interop
 4        id_ban_group
 5        id_fantoir
 6        numero
 7        suffixe
 8        nom_voie
 9        code_postal
10        nom_commune
11        code_insee
12        nom_complementaire
13        x
14        y
15        lon
16        lat
17        typ_loc
```

Counting occurrences of pairs of (zip code,street name)

Map

record $\rightarrow$ ((street name,zip code),1)

Reduce

Classical word count ($l \rightarrow \mathit{length}(l)$)

Counting occurrences of pairs of (zip code,street name) MAP

```python
#/usr/bin/python

import sys

for myline in sys.stdin:
    myline = myline.strip()
    subs = myline.split(';')
    if len(subs)>8:
        voie = subs[7]
        print('%s %s\t%s' % (subs[8],voie, 1))
```

Counting occurrences of pairs of (zip code,street name) RED

```
from operator import itemgetter
import sys
current_word = ""
current_count = 0
word = ""
for myline in sys.stdin:
   myline = myline.strip()
   subs = myline.split('\t',1)
   if len(subs)>1:
      count = int(str(subs[1]))
      word=subs[0]
      if current_word == word:
         current_count += count
      else:
         if current_word:
            print('%s\t%s' % (current_word, current_count) )
         current_count = count
         current_word = word
if current_word == word:
   print('%s\t%s' % (current_word, current_count))
```

Counting number of zip code for each street name

Map

((street name,zip code),n) $\rightarrow$ (street name,1)

Reduce

Classical word count reducer

Counting occurrences of pairs of (zip code,street name) MAP

```
#/usr/bin/python

import sys

for myline in sys.stdin:
    myline = myline.strip()
    subs = myline.split('\t',1)
    if len(subs)>1:
        subs2 = subs[0].split(' ',1)
        if len(subs2)>1:
            print('%s\t1' % (subs2[1]))
```

Extract the popular street names

Map

$(\text{street name}, n) \rightarrow (\text{street name}, n)$ when $n > 1000$

Reduce

Identity

Counting occurrences of pairs of (zip code,street name) MAP

```python
#/usr/bin/python

import sys

for myline in sys.stdin:
    myline = myline.strip()
    subs = myline.split('\t')
    if len(subs) > 1:
        print('%.9d\t%s' % (int(subs[1]), subs[0]))
```