BIGDATA TEAM

MegaFon Course: Big Data

MapReduce Optimization

Dral Alexey, aadral@bigdatateam.org
CEO at BigData Team, http://bigdatateam.org/
https://www.facebook.com/bigdatateam/

17.06.2019, Moscow, Russia

BIGDATA TEAM

Прошлое занятие

- Вы можете объяснить, что происходит когда “умирает” Mapper или Reducer
- Вы знаете, за что отвечают ResourceManager и NodeManager в YARN
- Вы знаете 3 фазы MapReduce (Map, Shuffle & Sort, Reduce)
- Вы знаете, что такое MapReduce Streaming и как он работает (примеры: distributed grep, wc, LineCount, WordCount)

BIGDATA TEAM

На этом занятии

- [18:30 - 19:30] MapReduce WordCount WarmUp
- -- перерыв, Q&A
- [19:40 - 20:30] Distributed Cache
- -- перерыв, Q&A
- [20:40 - 21:30] Combiner, Partitioner, Comparator

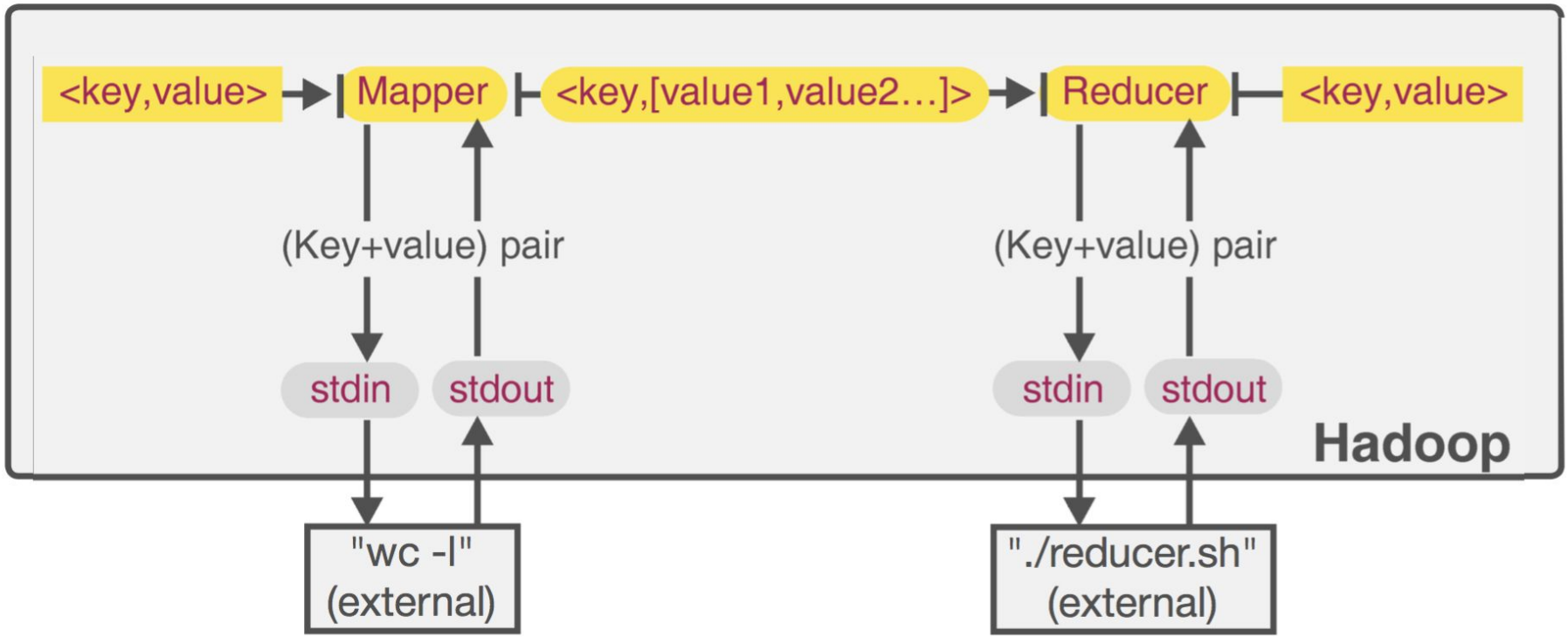
MapReduce Examples

<article id> <tab> <article content>

Line Count?

BIGDATA TEAM

MapReduce Examples



virtual-client.bigdatateam.ru: /home/aadral/public_examples/map_reduce

MapReduce Examples

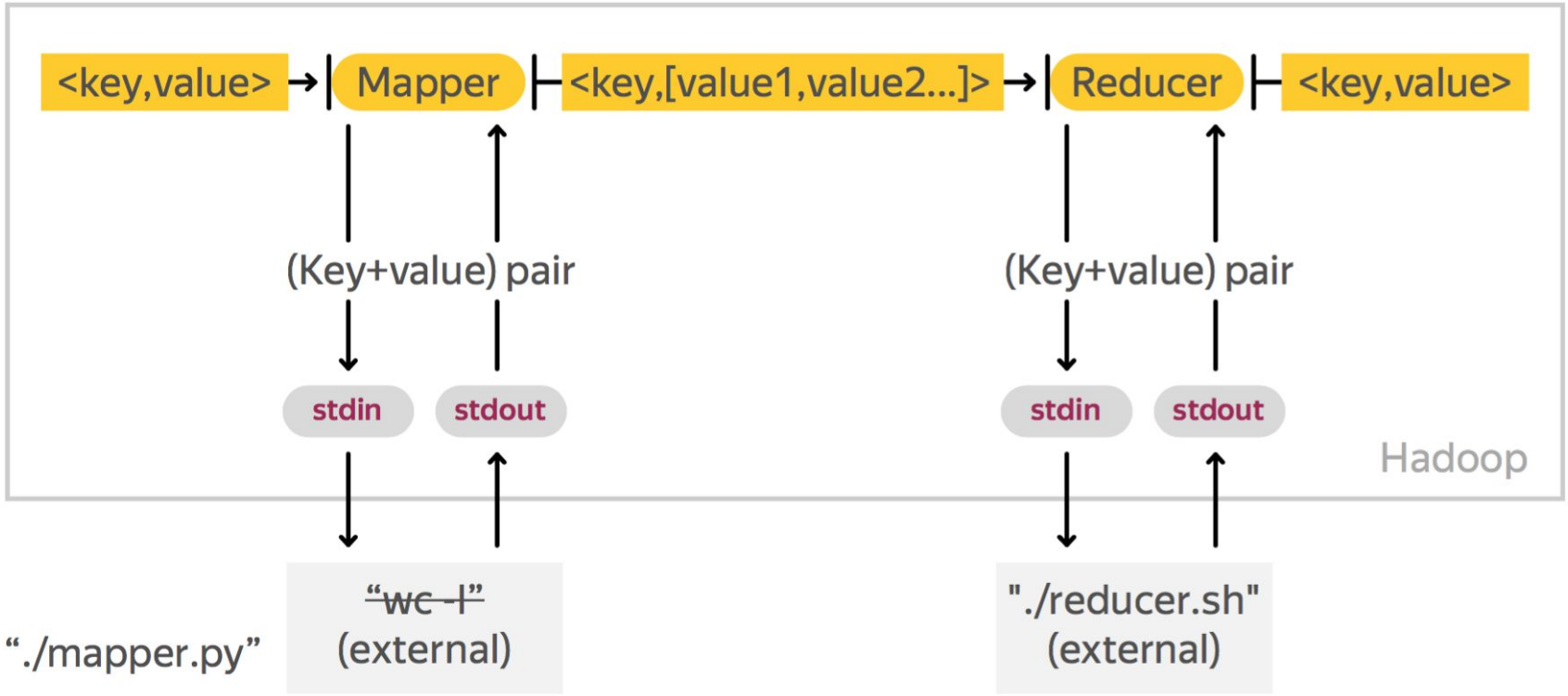
reducer.sh

```
#!/usr/bin/env bash
awk '{line_count += $1} END { print line_count }'
```

```
HADOOP_STREAMING_JAR="/path/to/hadoop-streaming.jar"
yarn jar $HADOOP_STREAMING_JAR \
        -mapper 'wc -l' \
        -reducer './reducer.sh' \
        -file reducer.sh \
        -numReduceTasks 1 \
        -input /data/wiki/en_articles \
        -output wc_mr_with_reducer
```

virtual-client.bigdatateam.ru: /home/aadral/public_examples/map_reduce

MapReduce Examples



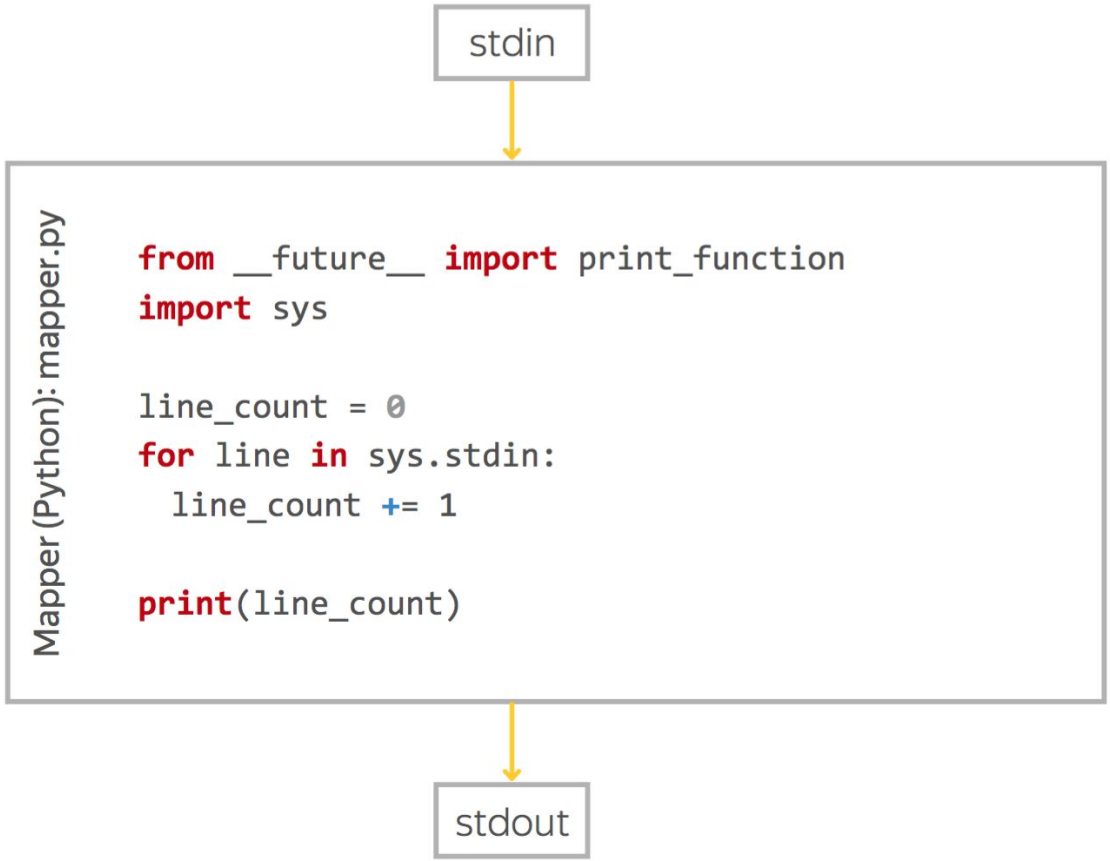
BIGDATA TEAM

MapReduce Examples

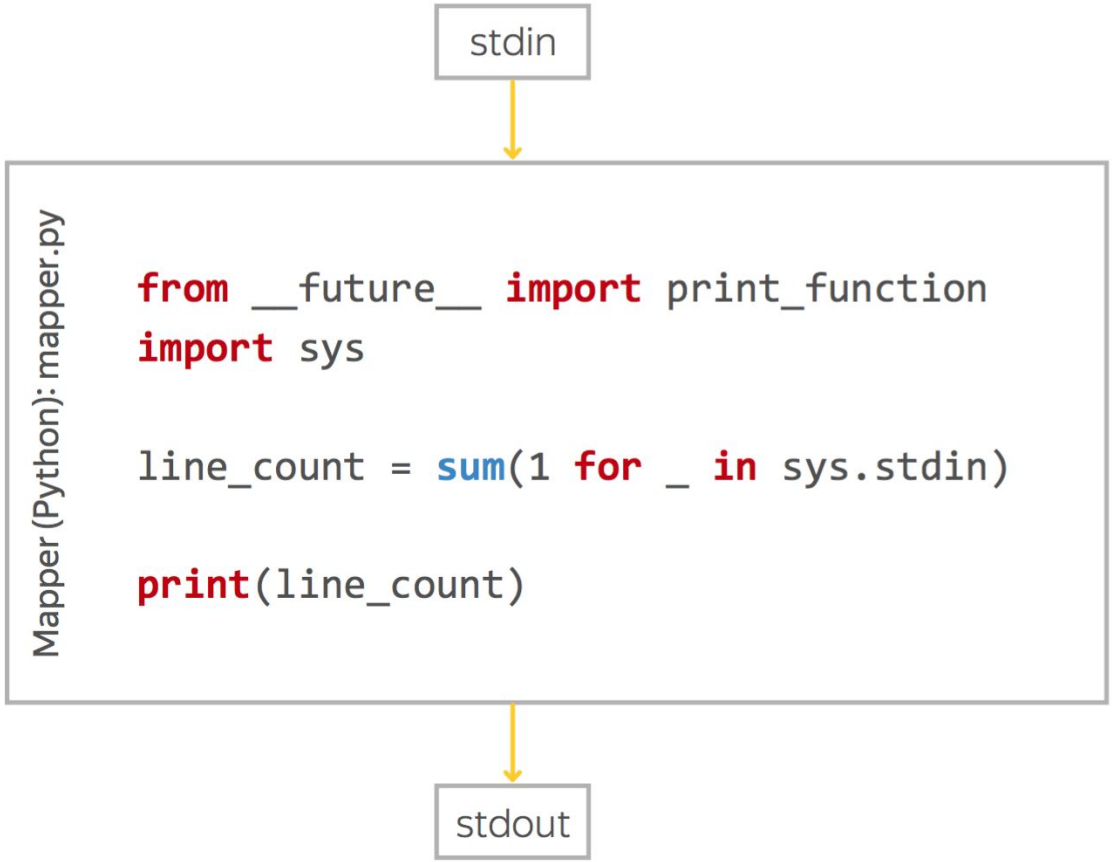
```
HADOOP_STREAMING_JAR="/path/to/hadoop-streaming.jar"
yarn jar $HADOOP_STREAMING_JAR \
    -files mapper.py, reducer.sh \
    -mapper 'python mapper.py' \
    -reducer './reducer.sh' \
    -numReduceTasks 1 \
    -input /data/wiki/en_articles \
    -output wc_mr_with_reducer
```

The general command line syntax is
bin/hadoop command [**genericOptions**] [commandOptions]
-conf <configuration file>
-D <property=value>
-fs <local|namenode:port>
-jt <local|resourcemanager:port>
-files <comma separated list of files>
-libjars <comma separated list of jars>
-archives <comma separated list of archives>

MapReduce Examples

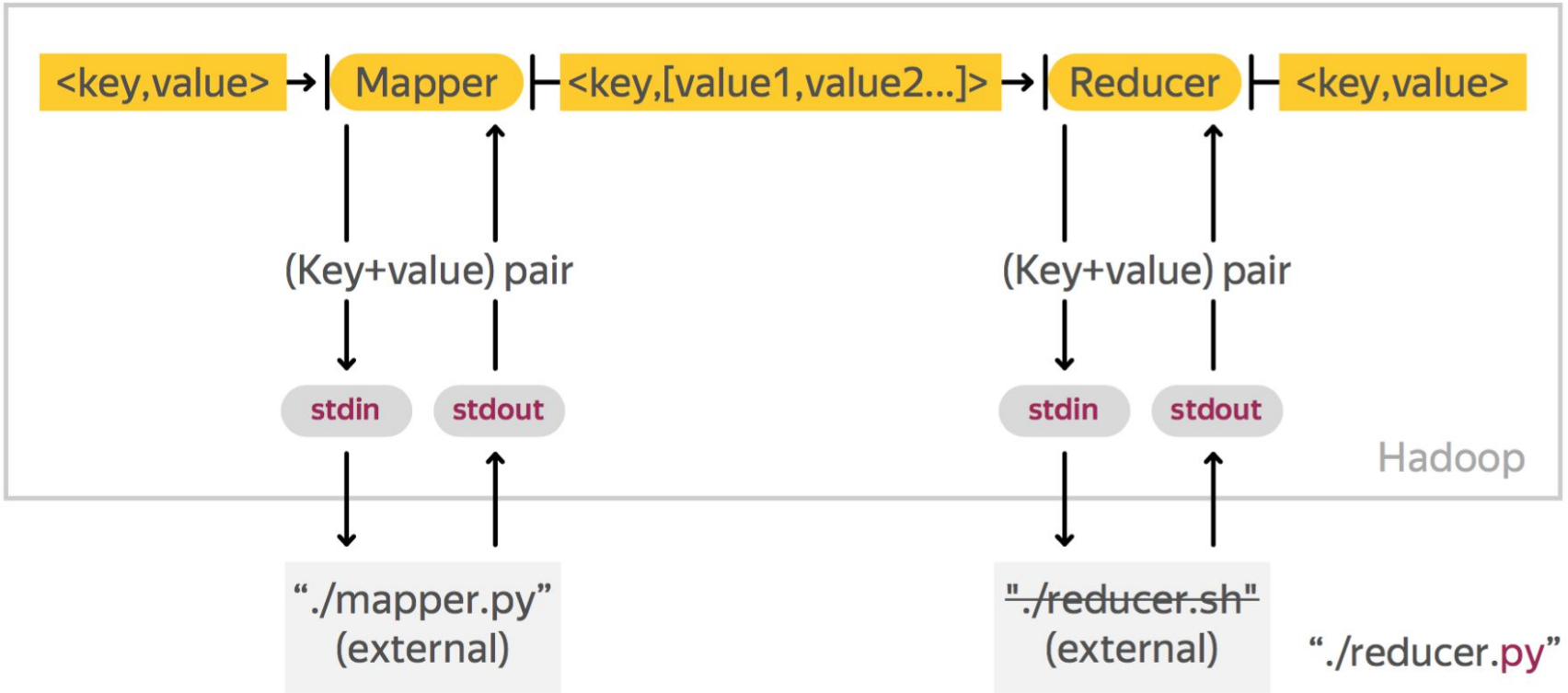


MapReduce Examples

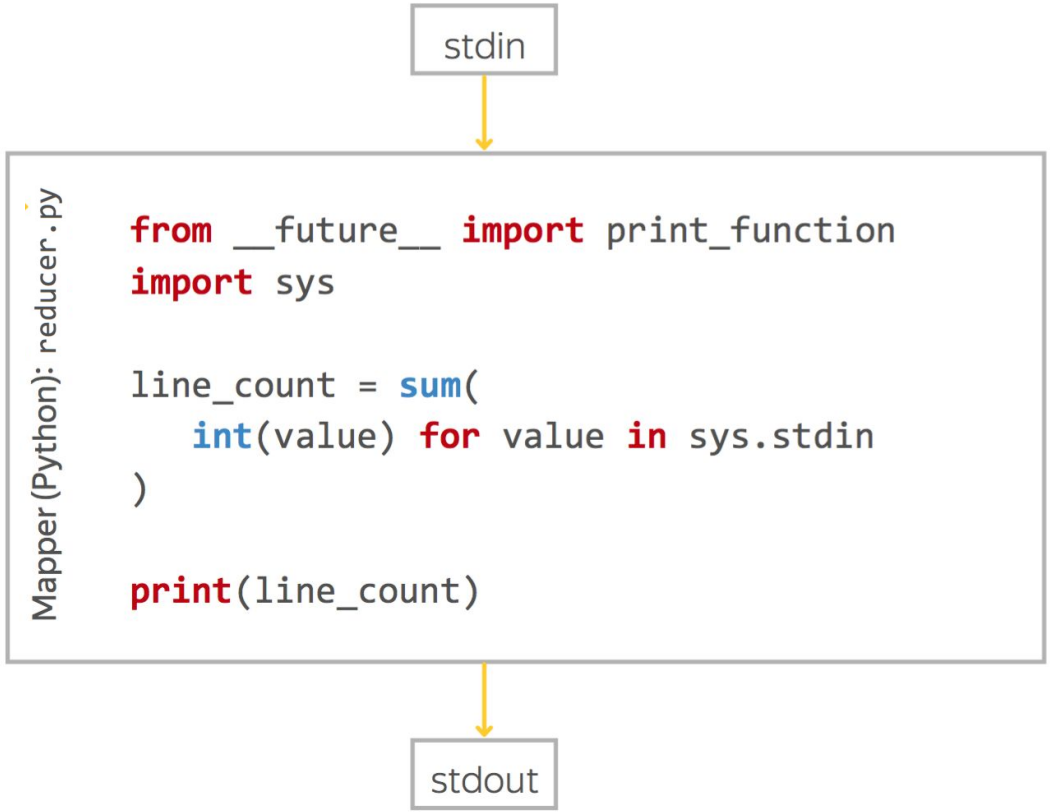


BIGDATA TEAM

MapReduce Examples



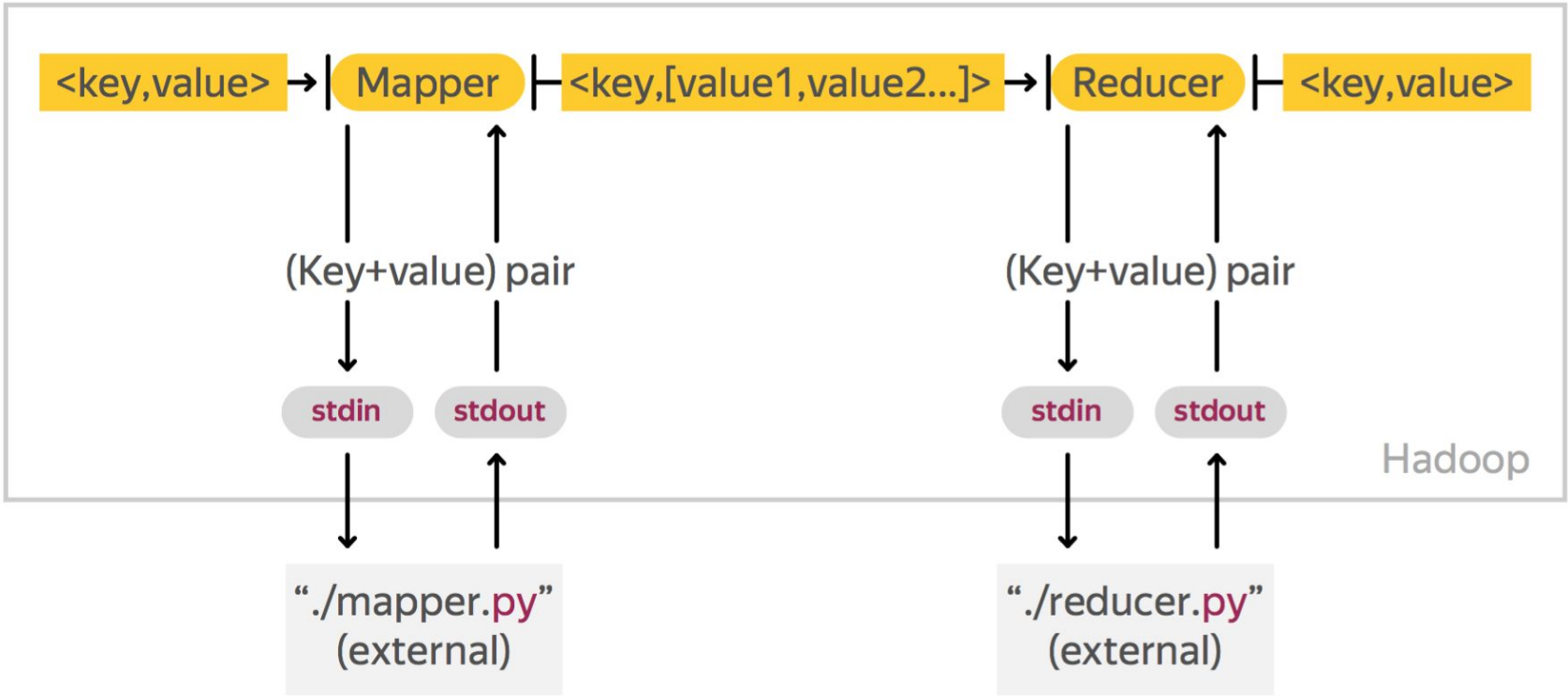
MapReduce Examples



MapReduce Examples

```
HADOOP_STREAMING_JAR="/path/to/hadoop-streaming.jar"
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py,reducer.py  \
          -mapper 'python mapper.py' \
          -reducer './reducer.py' \
          -numReduceTasks 1 \
          -input /data/wiki/en_articles \
          -output wc_mr_with_reducer
```

MapReduce Examples




KEEP
CALM
AND
TRY
CODING

MapReduce Word Count

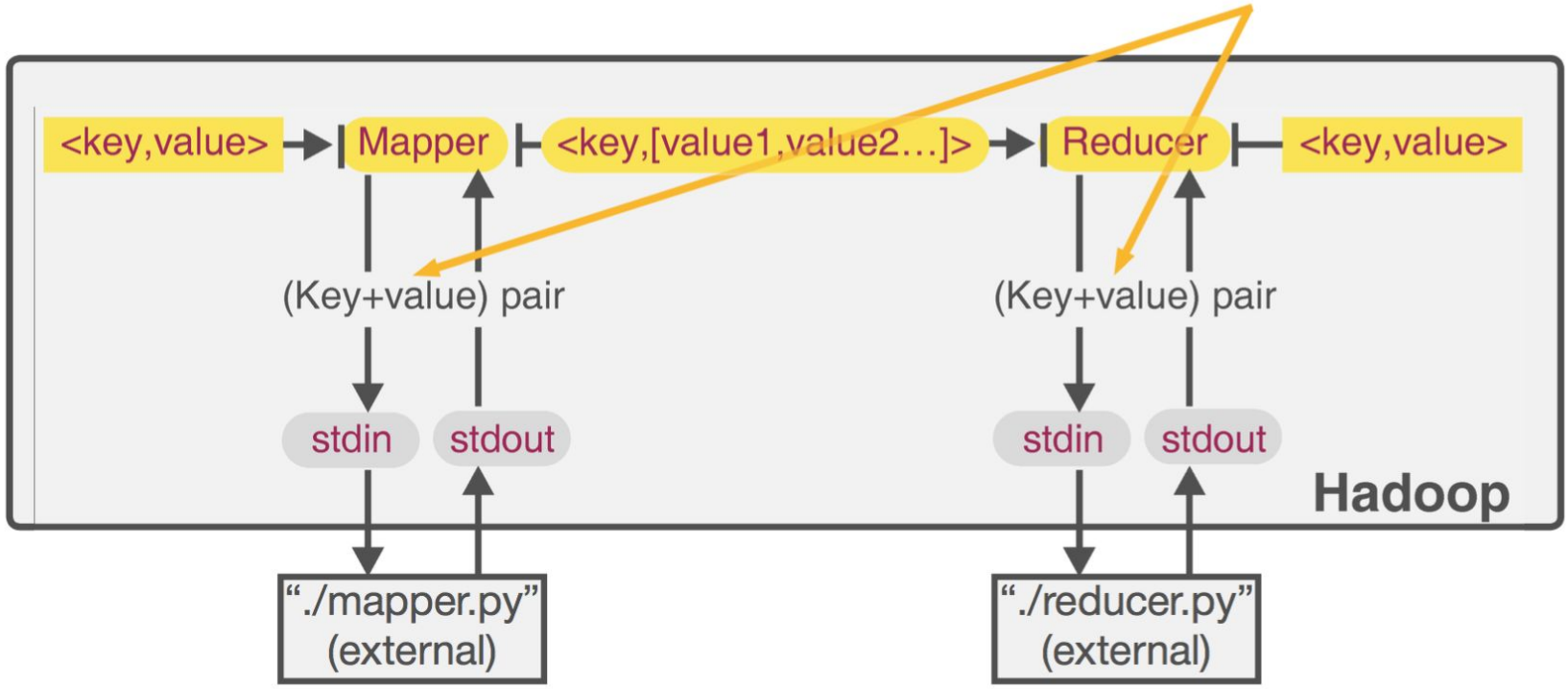
Word Count

Apache Hadoop (/hə`du:p/) is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework...

```
'the': 3, 'of': 3, 'hadoop': 2, …
```

Word Count


<key,value>
Mapper
<key,[value1,value2…]>
Reducer
<key,value>
(Key+value) pair
(Key+value) pair
stdin
stdout
stdin
stdout
Hadoop
"./mapper.py"
(external)
"./reducer.py"
(external)

Word Count

<article id> <tab> <article content>

key value

```
from __future__ import print_function
import sys

for line in sys.stdin:
  article_id, content = line.split("\t", 1)
  words = content.split()
  for word in words:
      print(word, 1, sep="\t")
```

Word Count

```
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py \
          -mapper 'python mapper.py' \
          -numReduceTasks 0 \
          -input /data/wiki/en_articles \
          -output word_count
```

```
$ hdfs dfs -text /data/wiki/en_articles/* | head -c 80
12 <tab> Anarchism          Anarchism is often defined as a political
philosophy which …
```

```
$ hdfs dfs -ls -h word_count
Found 3 items
-rw-r--r--   3 adral adral   0 2017-03-22 11:40 word_count/_SUCCESS
-rw-r--r--   3 adral adral   47.8 M 2017-03-22 11:40 word_count/part-00000
-rw-r--r--   3 adral adral   47.9 M 2017-03-22 11:40 word_count/part-00001
```

Word Count

```
$ hdfs dfs -text
    word_count/part-… | head -5
```

part-00000	part-00001
Basel 1	Anarchism 1
Basel 1	Anarchism 1
(1	is 1
) 1	often 1
or 1	defined 1
...	...

Word Count

```
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py \
          -mapper 'python mapper.py' \
          -numReduceTasks 1 \
          -input /data/wiki/en_articles \
          -output word_count
```

```
$ hdfs dfs -text word_count/part-00000 | head
! 1
! 1
! 1
! 1
! 1
…
```

Word Count

<article id> <tab> <article content>

key value

```
from __future__ import print_function
import re
import sys

for line in sys.stdin:
 article_id, content = line.split("\t", 1)
 words = re.split("\W+", content)
 for word in words:
     if word:
        print(word, 1, sep="\t")
```

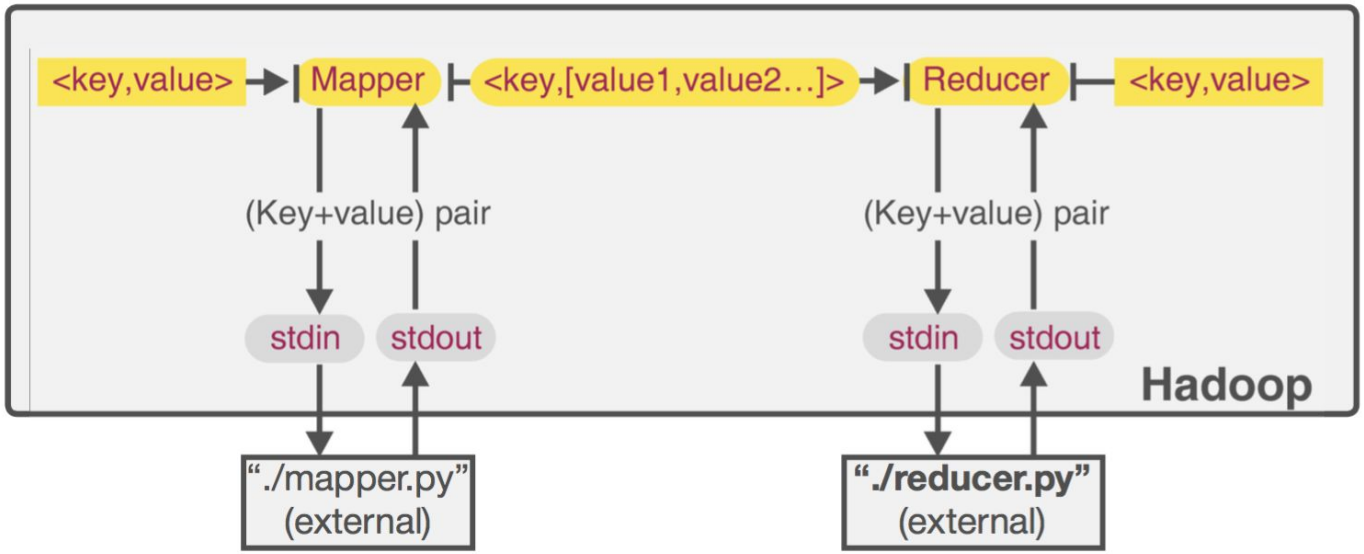
Word Count

```
yarn jar $HADOOP_STREAMING_JAR \
            -files mapper.py \
            -mapper 'python mapper.py' \
            -numReduceTasks 1 \
            -input /data/wiki/en_articles \
            -output word_count
```

```
$ hdfs dfs -text word_count/part-00000 | head -4
0  1
0  1
0  1
0  1
```

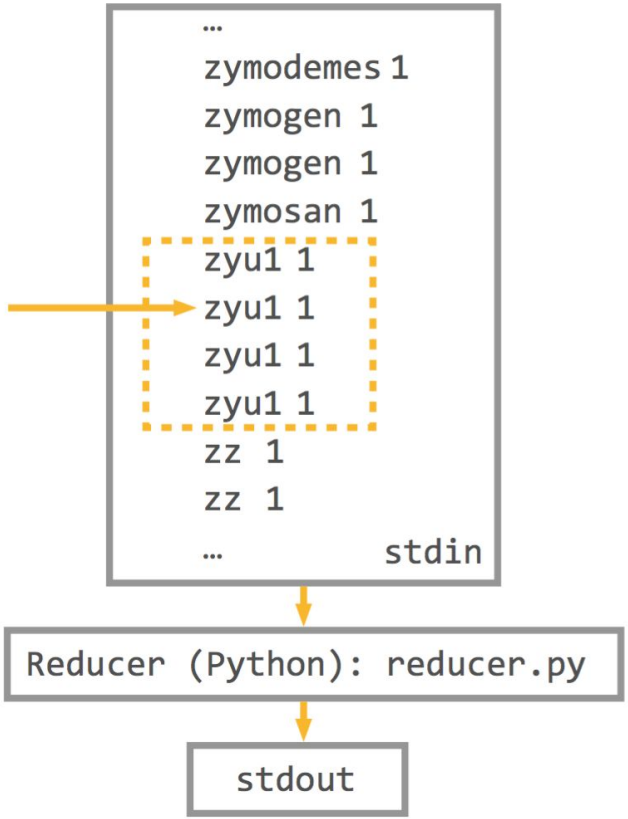
```
$ hdfs dfs -tail word_count/part-00000 | tail -4
zyu1 1
zyu1 1
zz  1
zz  1
```

Word Count



- define input format
- **aggregate sorted data by key**
- process data
- define output format

Word Count

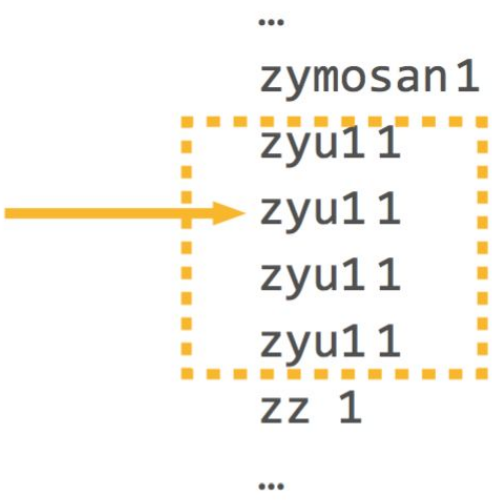

…
zymodemes 1
zymogen 1
zymogen 1
zymosan 1
zyu1 1
zyu1 1
zyu1 1
zyu1 1
zz 1
zz 1
…
stdin
Reducer (Python): reducer.py
stdout

Word Count

```
from __future__ import print_function
import sys

current_word = None
word_count = 0

for line in sys.stdin:
    word, counts = line.split("\t", 1)
    counts = int(counts)
```

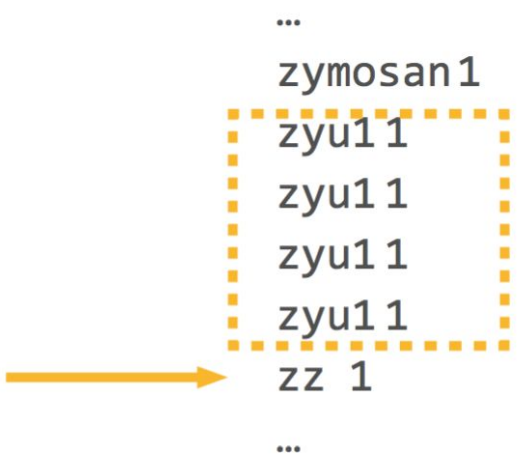
Word Count

```
from __future__ import print_function
import sys

current_word = None
word_count = 0

for line in sys.stdin:
    word, counts = line.split("\t", 1)
    counts = int(counts)
    if word == current_word:
        word_count += counts
    else:
```



Word Count

```
from __future__ import print_function
import sys

current_word = None
word_count = 0

for line in sys.stdin:
    word, counts = line.split("\t", 1)
    counts = int(counts)
    if word == current_word:
        word_count += counts
    else:
        if current_word:
            print(current_word, word_count, sep="\t")
        current_word = word
        word_count = counts
```



Word Count

```
from __future__ import print_function
import sys

current_word = None
word_count = 0

for line in sys.stdin:
    word, counts = line.split("\t", 1)
    counts = int(counts)
    if word == current_word:
        word_count += counts
    else:
        if current_word:
            print(current_word, word_count, sep="\t")
        current_word = word
        word_count = counts

if current_word:
    print(current_word, word_count, sep="\t")
```

Word Count

part-00000	part-00005
...	...
zuang 1	zsu 1
zucchini 5	zuchetto 1
zuerst 1	zure 1
zumase 2	zuurstof 1
zyu 1 4	zz 2


KEEP
CALM
AND
TRY
CODING

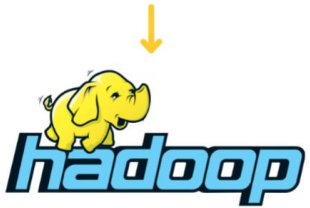
Tea / Coffee Break

Distributed Cache

Filter by Vocabulary

<article id> <tab> <article content>

key value




```
...
zymodemes1
zymogen 2
zymosan 1
zyu1 4
zz 2
```

Filter by Vocabulary

```
from __future__ import print_function
import re
import sys

def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

vocabulary = read_vocabulary("vocabulary.txt")

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.split("\W+", content)
    for word in words:
        if word in vocabulary:
            print(word, 1, sep="\t")
```

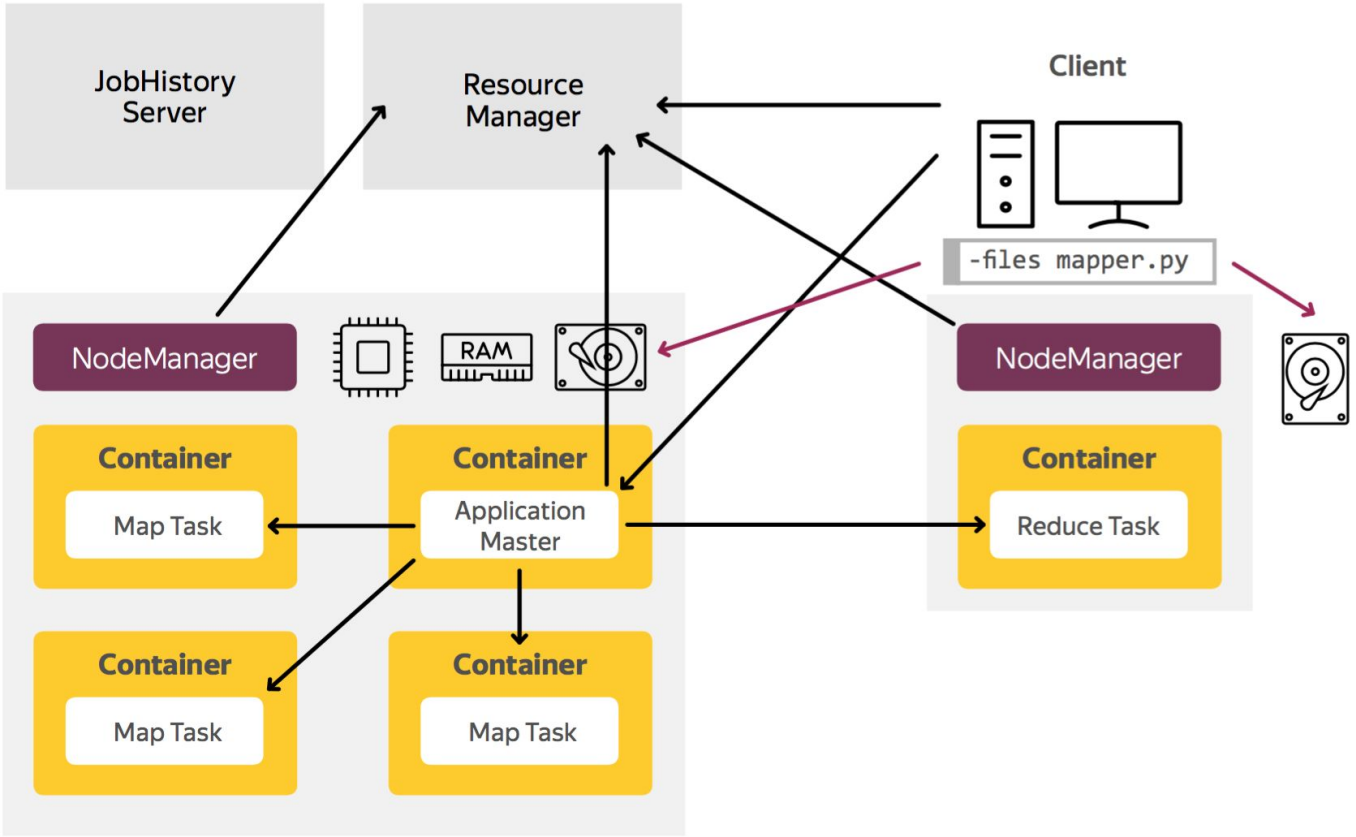
Filter by Vocabulary

```
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py,reducer.py,vocabulary.txt \
          -mapper 'python mapper.py' \
          -reducer 'python reducer.py' \
          -input /data/wiki/en_articles \
          -output word_count
```

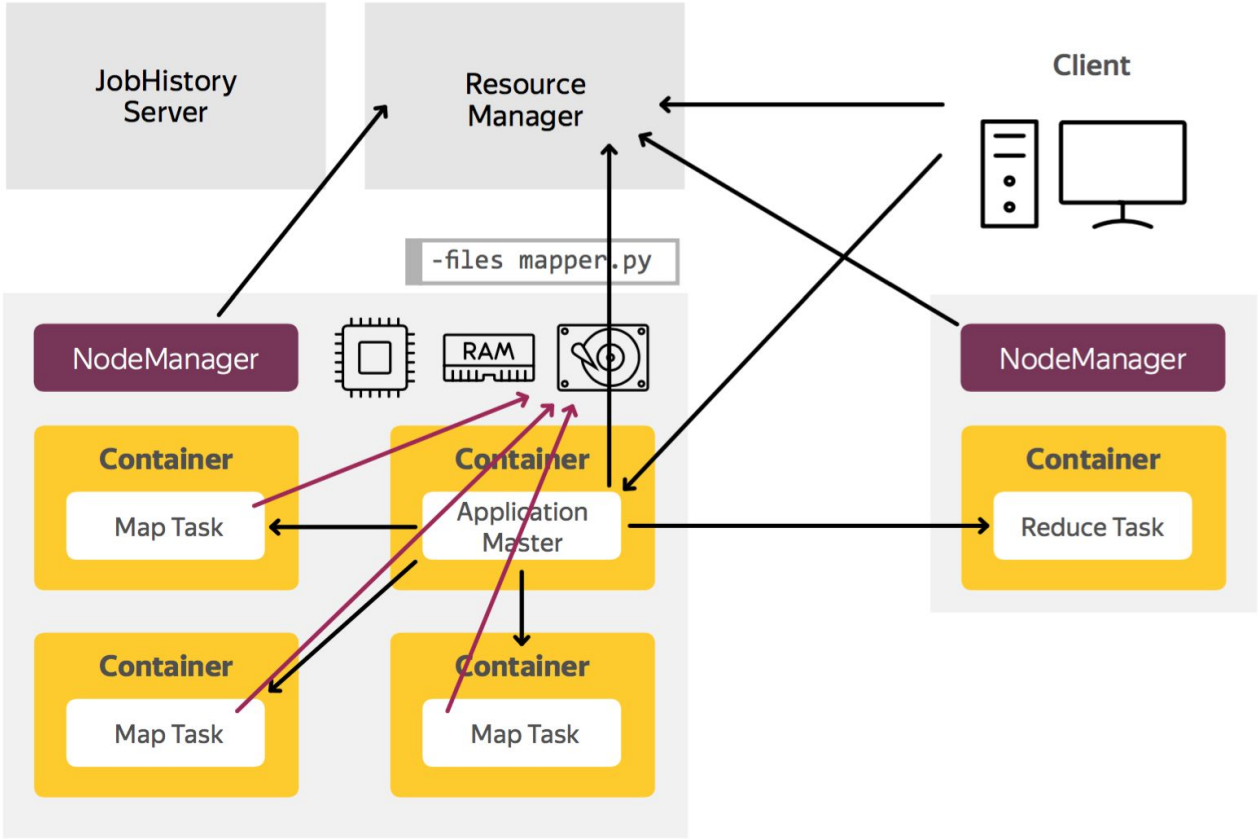
Mapper (Python): mapper.py

```
def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

vocabulary = read_vocabulary("vocabulary.txt")
```

Data Distribution

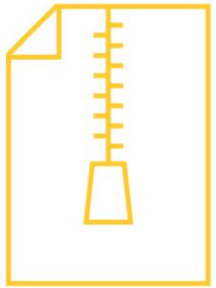


Data Usage


JobHistory Server
Resource Manager
Client
-files mapper.py
NodeManager
RAM
NodeManager
Container
Map Task
Container
Application Master
Container
Reduce Task
Container
Map Task
Container
Map Task

Distributed Cache Options

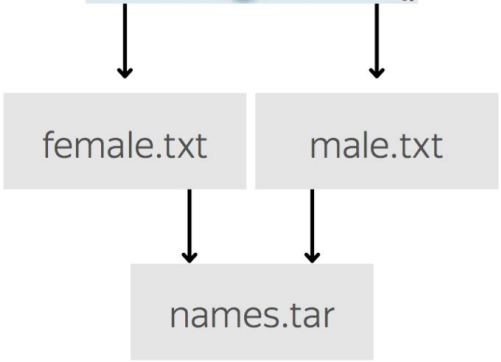
-files



-archives

-libjars

Birth Names Example




```
$ tar -cf names.tar male.txt female.txt
```

Birth Names Example

```
$ tar -cf names.tar male.txt female.txt
```

names.tar


```
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py,reducer.py,names.tar \
          -archives names.tar \
          -mapper 'python mapper.py' \
          -reducer 'python reducer.py' \
          -input /data/wiki/en_articles \
          -output word_count
```

Birth Names Example

```
from __future__ import print_function
import re
import sys

def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

male_names = read_vocabulary("names.tar/male.txt")
female_names = read_vocabulary("names.tar/female.txt")

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.split("\W+", content)
    for word in words:
        if word in male_names or word in female_names:
            print(word, 1, sep="\t")
```

Birth Names Example


```
$ hdfs dfs -text word_count/* | sort -nrk2,2
James   2284
Thomas  1941
Jack    786
Harry   504
Muhammad        444
Oliver  250
Charlie 250
Jacob   234
Emily   128
Isabella        99
Sophia  92
Noah    80
Sophie  64
Lily    31
Olivia  27
Ethan   27
Ella    25
Amelia  25
Isabelle        18
Chloe   9
```


KEEP
CALM
AND
TRY
CODING

BIGDATA TEAM

Tea / Coffee Break

Мир MapReduce

Combiner Partitioner Comparator

Мир MapReduce

Combiner Partitioner Comparator

BIGDATA TEAM

Постановка задачи

<article id> <tab> <article content>

key value

input: word word a word b c d word d e...

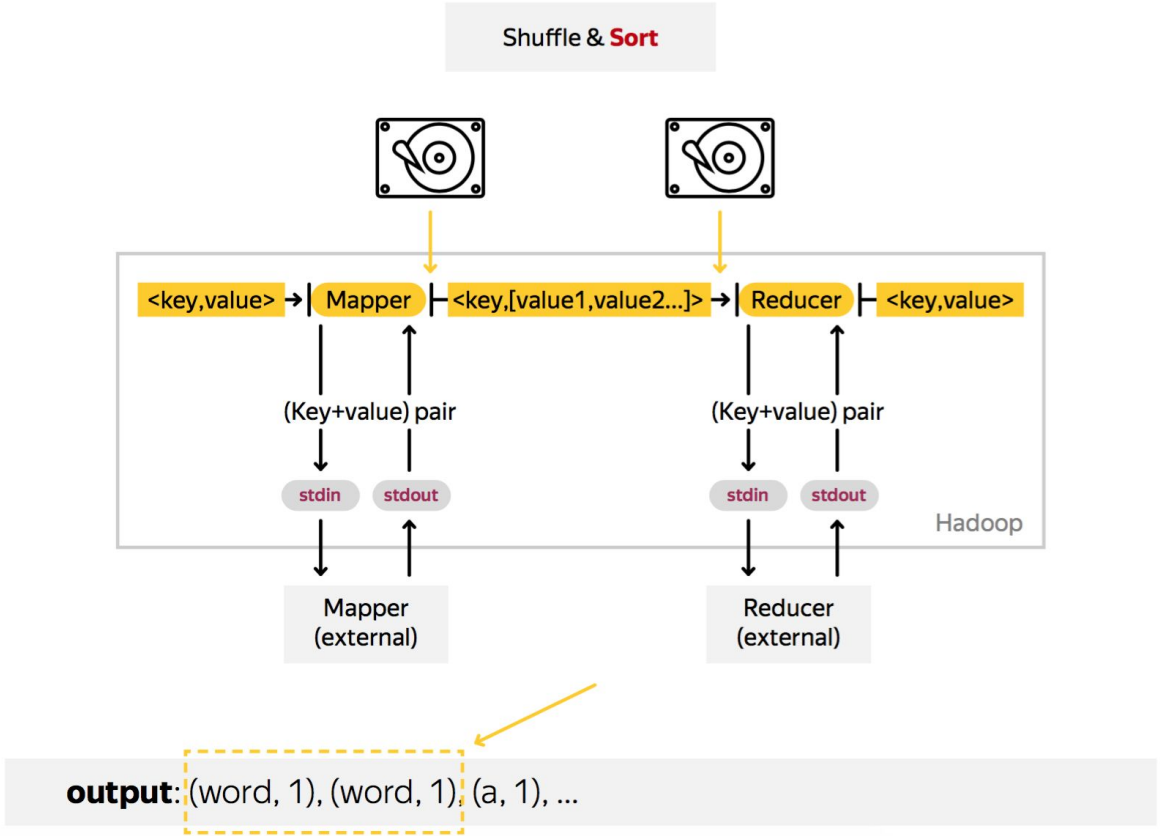
Mapper (Python): reporter_mapper.py

```
from __future__ import print_function
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = content.split()
    for word in words:
            print(word, 1, sep="\t")
```

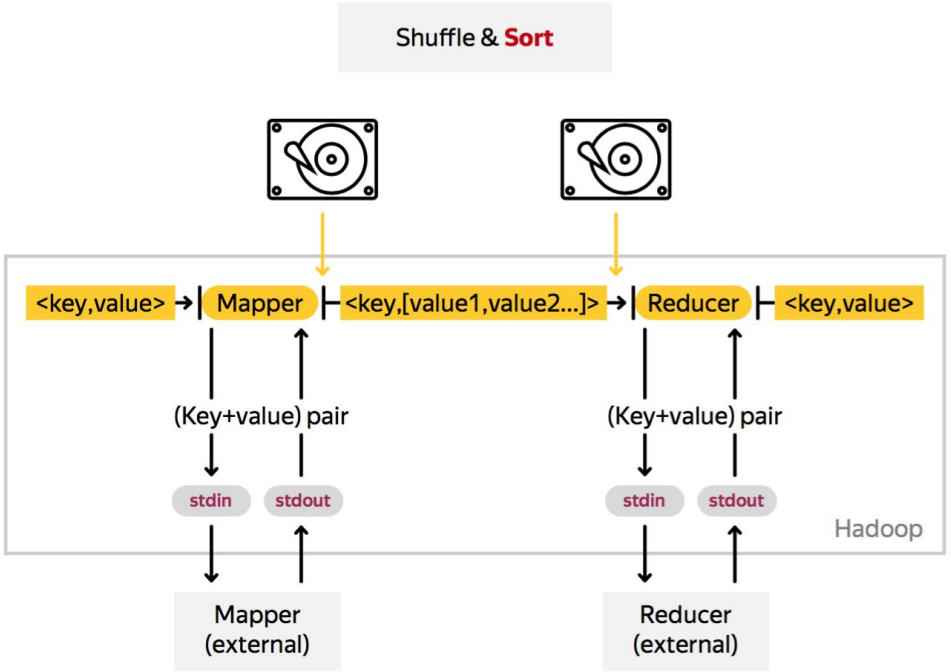
output: (word, 1), (word, 1), (a, 1), ...

BIGDATA TEAM

Постановка задачи



Решение



output: ~~(word, 1), (word, 1), (a, 1), ...~~

(word, 2), (a, 1), ...

Решение

input: word word a word b c d word d e...

Mapper (Python): reporter_mapper.py

```
from __future__ import print_function
from collections import Counter
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = content.split()
    counts = Counter(words)
    for word, word_count in counts.items():
        print(word, word_count, sep="\t")
```

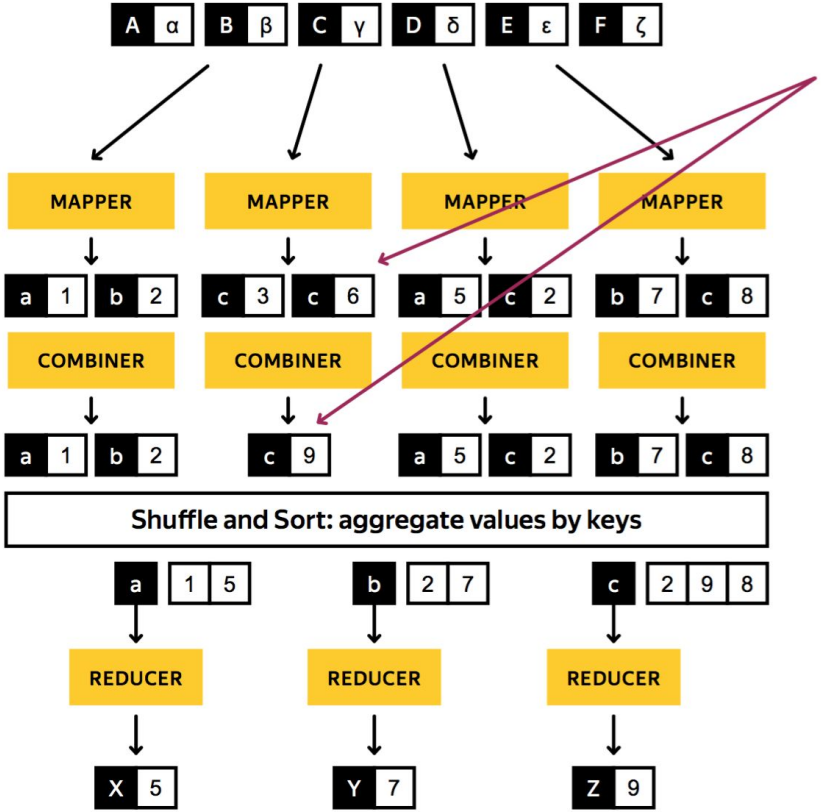
output: (a, 1), (b, 1), (c, 1), (d, 2), (e, 1), (word, 4), ...

Сравнение

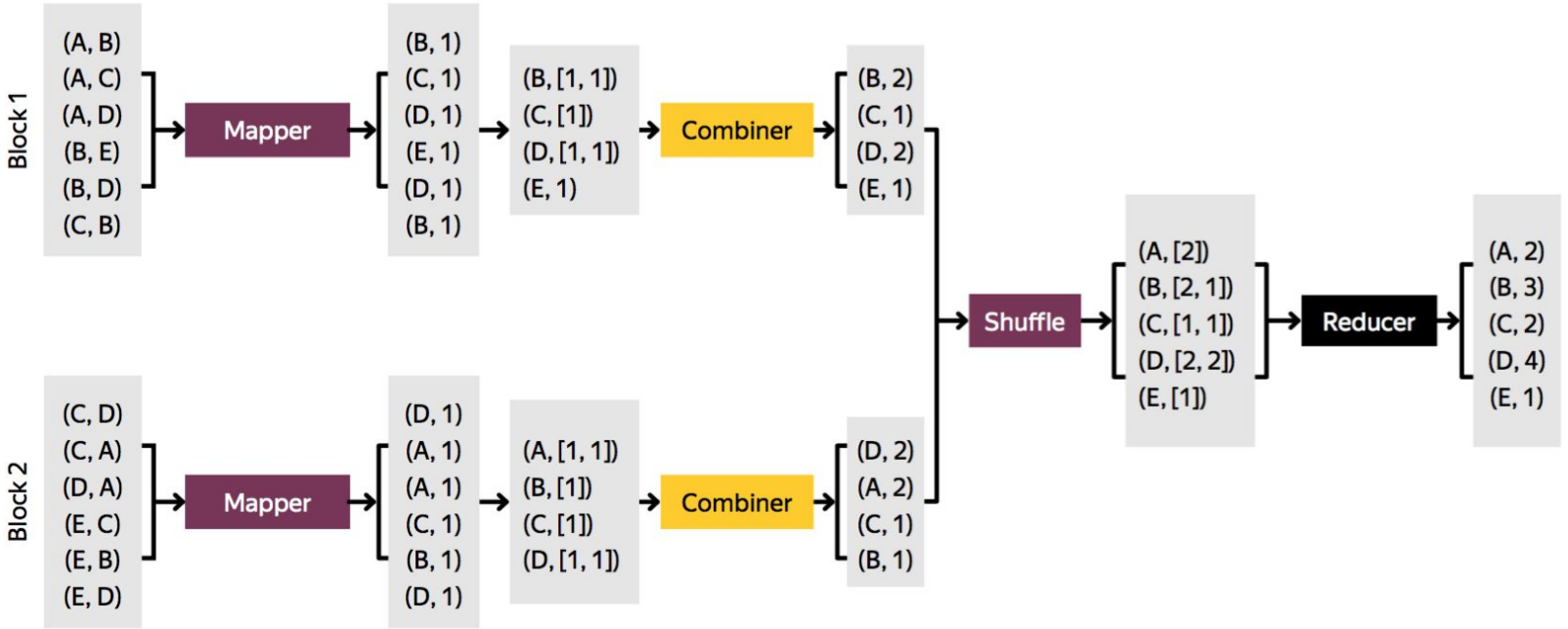
	without Combiner	with Combiner
Wall time (sec)	**935**	**528**
CPU time (sec)	**9790**	**6584**
Local FS Read (MB)	**3006**	**1324**
Local FS Write (MB)	**4527**	**1963**
Peek Map phys. memory (MB)	**526**	**606**
Peek Map virt. memory (MB)	**2131**	**2144**
Peek Reduce phys. memory (MB)	**2744**	**631**
Peek Reduce virt. memory (MB)	**3196**	**3194**

BIGDATA TEAM

Уточненная модель MapReduce

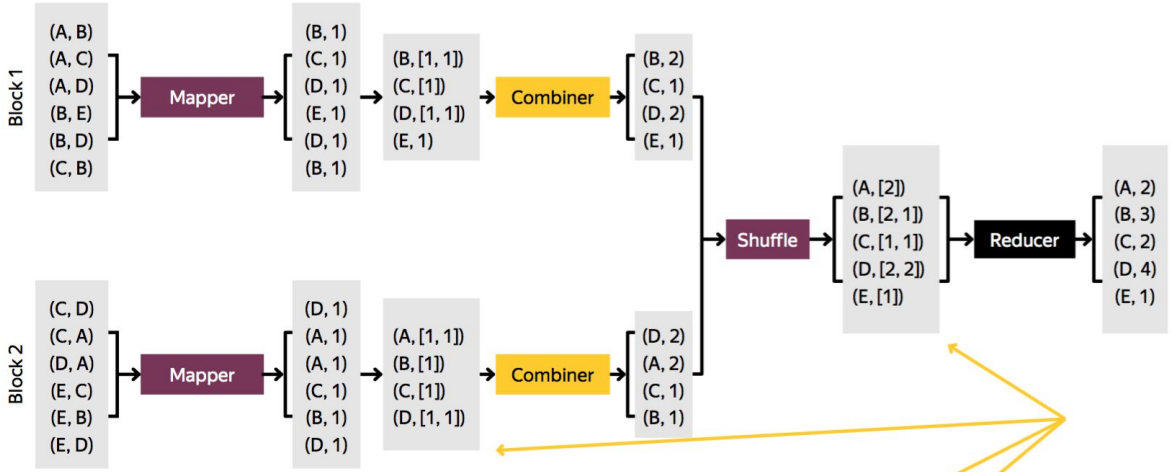


Combiner


Block 1
(A, B)
(A, C)
(A, D)
(B, E)
(B, D)
(C, B)
Mapper
(B, 1)
(C, 1)
(D, 1)
(E, 1)
(D, 1)
(B, 1)
(B, [1, 1])
(C, [1])
(D, [1, 1])
(E, 1)
Combiner
(B, 2)
(C, 1)
(D, 2)
(E, 1)
Block 2
(C, D)
(C, A)
(D, A)
(E, C)
(E, B)
(E, D)
Mapper
(D, 1)
(A, 1)
(A, 1)
(C, 1)
(B, 1)
(D, 1)
(A, [1, 1])
(B, [1])
(C, [1])
(D, [1, 1])
Combiner
(D, 2)
(A, 2)
(C, 1)
(B, 1)
Shuffle
(A, [2])
(B, [2, 1])
(C, [1, 1])
(D, [2, 2])
(E, [1])
Reducer
(A, 2)
(B, 3)
(C, 2)
(D, 4)
(E, 1)

BIGDATA TEAM

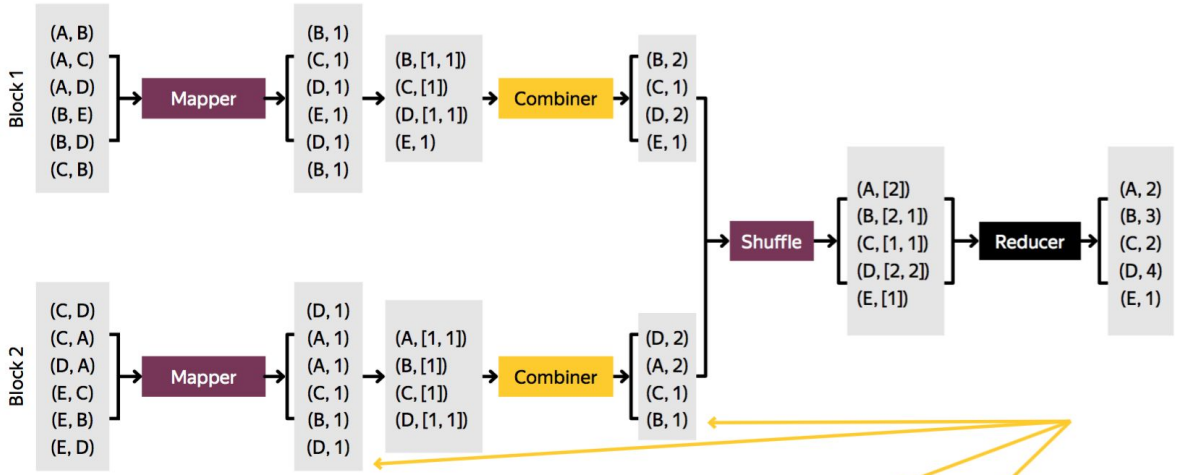
Формальная модель



- read: [(k_in, v_in), ...]
- map: (k_in, v_in) —> [(k_interm, v_interm), ...]
- combiner: **(k_interm, [(v_interm, ...)])** —> [(k_interm, v_interm), ...]
- Shuffle & Sort: sort and group by k_interm
- reduce: **(k_interm, [(v_interm, ...)])** —-> [(k_out, v_out), ...]

BIGDATA TEAM

Формальная модель



- read: [(k_in, v_in), ...]
- map: (k_in, v_in) —> **[(k_interm, v_interm), ...]**
- combiner: **(k_interm, [(v_interm, ...)])** —> **[(k_interm, v_interm), ...]**
- Shuffle & Sort: sort and group by k_interm
- reduce: **(k_interm, [(v_interm, ...)])** —-> [(k_out, v_out), ...]

Использование

```
yarn jar $HADOOP_STREAMING_JAR \
                -files mapper.py,reducer.py \
                -mapper 'python mapper.py' \
                -reducer 'python reducer.py' \
                -combiner 'python reducer.py' \
                -numReduceTasks 2 \
                -input griboedov.txt \
                -output word_count
```

Статистика использования

```
yarn jar $HADOOP_STREAMING_JAR \
              -files mapper.py,reducer.py \
              -mapper 'python mapper.py' \
              -reducer 'python reducer.py' \
              -combiner 'python reducer.py' \
              -numReduceTasks 2 \
              -input griboedov.txt \
              -output word_count
```

```
Map-Reduce Framework
    Map input records=2681
    Map output records=182
    Map output bytes=1218
    Map output materialized bytes=955
    Input split bytes=126
    Combine input records=182
    Combine output records=99
    Reduce input groups=99
    Reduce shuffle bytes=955
    Reduce input records=99
    Reduce output records=99
```

Combiner: Mean

<article id> <tab> <article content>

key value

input: word word a word b c d word d e...

mean

output: (word, 3.5), (a, 0.5), ...

Mean Mapper

input: word word a word b c d word d e...

Mapper (Python): mapper.py

```
from __future__ import print_function
from collections import Counter
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = content.split()
    counts = Counter(words)
    for word, word_count in counts.items():
        print(word, word_count, sep="\t")
```

output: (a, 1), (d, 2), (word, 4), ...

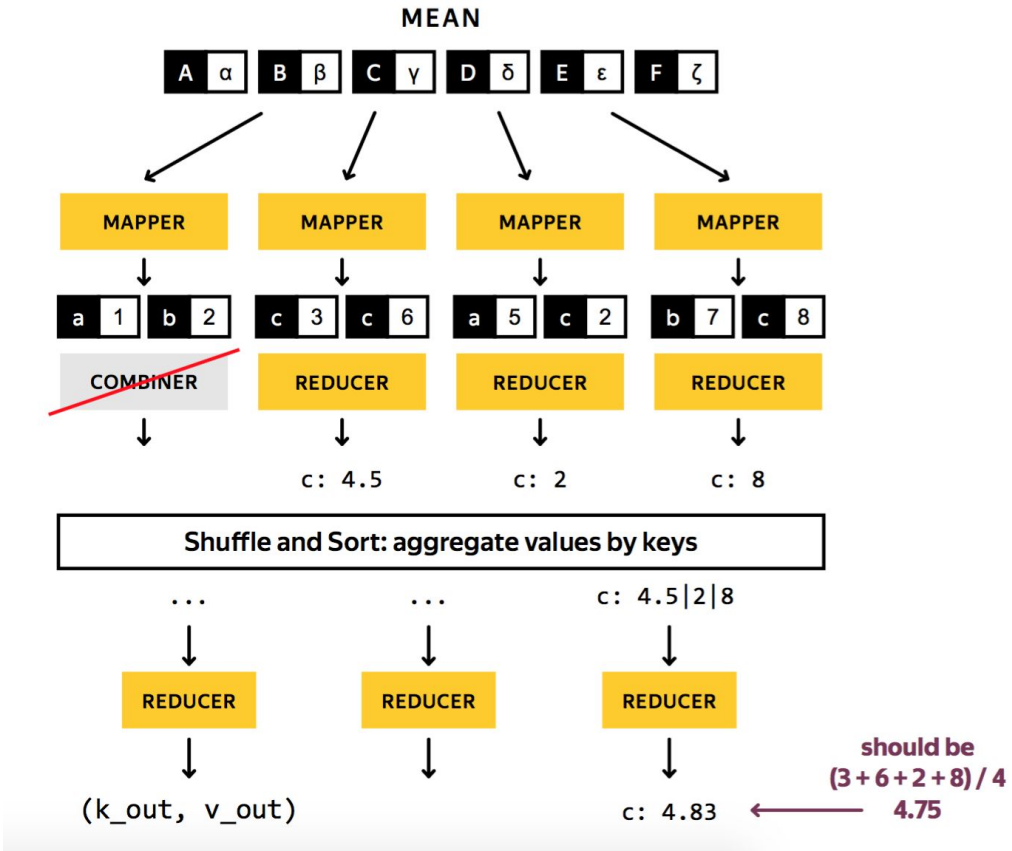
Mean Reducer

Mapper (Python): reducer.py

```
from __future__ import print_function
import sys

current_word = None
word_count, article_count = 0, 0

for line in sys.stdin:
    word, counts = line.split("\t", 1)
    counts = int(counts)
    if word == current_word:
        word_count += counts
        article_count += 1
    else:
        if current_word:
            print(current_word, word_count / article_count, sep="\t")
        current_word = word
        word_count, article_count = counts, 1

if current_word:
    print(current_word, word_count  / article_count, sep="\t")
```

BIGDATA TEAM

Постановка задачи



Mean Mapper Updated

input: word word a word b c d word d e...

Mapper (Python): mapper.py

```
from __future__ import print_function
from collections import Counter
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = content.split()
    counts = Counter(words)
    for word, word_count in counts.items():
        print(word, word_count, sep="\t")
        print(word, 1, word_count, sep="\t")
```

~~**output**: (a, 1), (d, 2), (word, 4), ...~~

output: (a, (1, 1)), (d, (1, 2)), (word, (1, 4)), ...

Mean Reducer

Mapper (Python): reducer.py

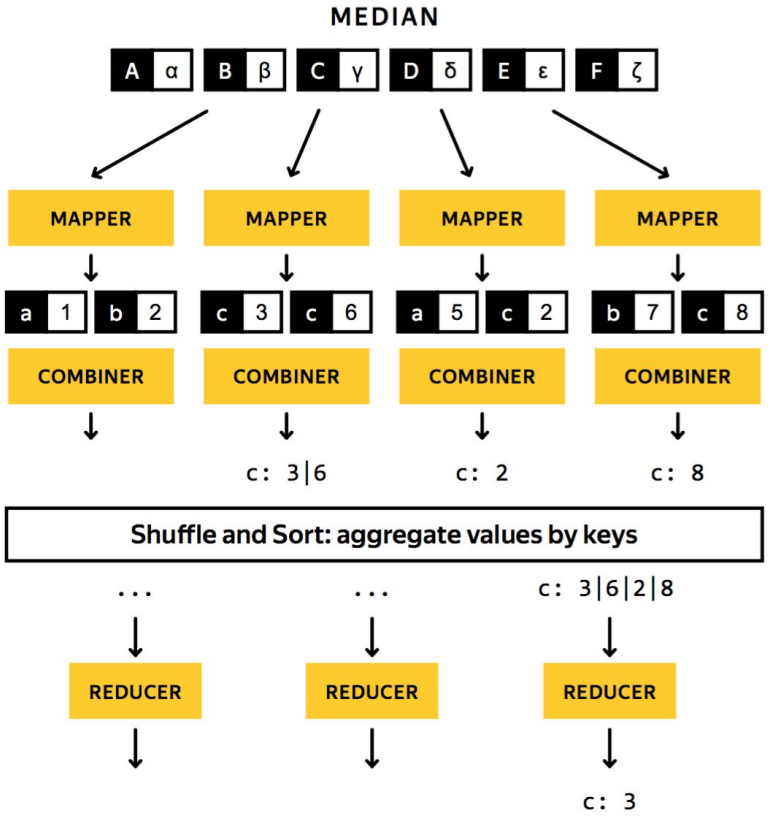
```
from __future__ import print_function
import sys

current_word = None
word_count, article_count = 0, 0

for line in sys.stdin:
    word, articles, counts = line.split("\t", 2)
    articles, counts = int(articles), int(counts)
    if word == current_word:
        word_count += counts
        article_count += articles
    else:
        if current_word:
            print(current_word, word_count / article_count, sep="\t")
            current_word = word
        word_count = counts
        article_count = articles

if current_word:
    print(current_word, word_count  / article_count, sep="\t")
```

Combiner: **Spot** the Problem

Mapper (Python): combiner.py

```
from __future__ import print_function
import sys

current_word = None
word_count, article_count = 0, 0

for line in sys.stdin:
    word, articles, counts = line.split("\t", 2)
    articles, counts = int(articles), int(counts)
    if word == current_word:
        word_count += counts
        article_count += articles
    else:
        if current_word:
            assert len(current_word.rstrip()) > 0
            print(current_word, word_count / article_count, sep="\t")
        current_word = word
        word_count = counts
        article_count = articles

if current_word:
    print(current_word, word_count  / article_count, sep="\t")
```

Combiner: Median



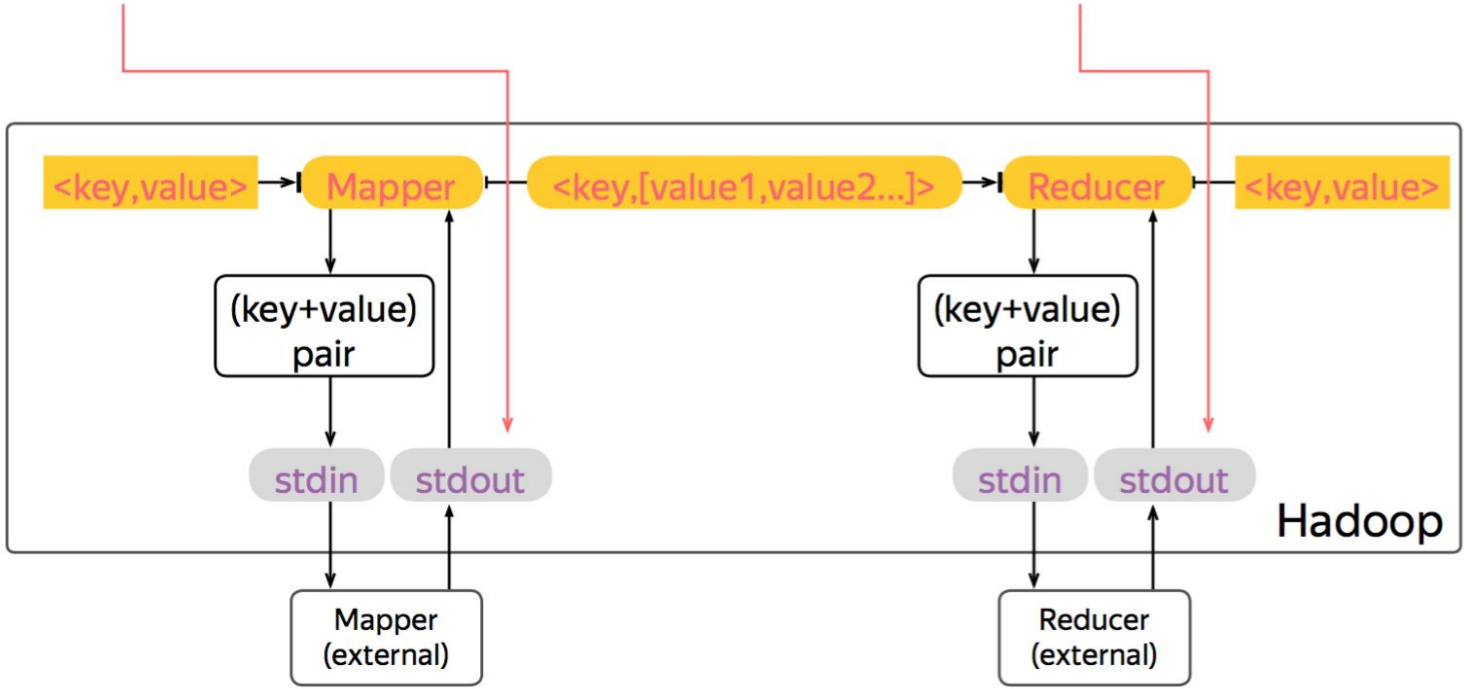
Мир MapReduce

Combiner Partitioner Comparator

Partitioner Flags

-D stream.map.output.field.separator=.
-D stream.num. map.output.key.fields=1

-D stream.reduce.output.field.separator=.
-D stream.num. reduce.output.key.fields=2



input

```
$ cat subnet.txt
1.2.3.4
2.3.4.5
3.4.5.6
4.5.6.7
```

```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D stream.map.output.field.separator=. \
        -D stream.num.map.output.key.fields=1 \
        -D stream.reduce.output.field.separator=. \
        -D stream.num.reduce.output.key.fields=2 \
        -files identity_mr.py \
        -mapper 'python identity_mr.py' \
        -reducer 'python identity_mr.py' \
        -numReduceTasks 2 \
        -input subnet.txt \
        -output subnet_out \
```

output

```
cat subnet_out/*
2    3.4    5
4    5.6    7
1    2.3    4
3    4.5    6
```

input

```
$ cat subnet.txt
1.2.3.4
2.3.4.5
3.4.5.6
4.5.6.7
```

```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D stream.map.output.field.separator=. \
        -D stream.num.map.output.key.fields=1 \
        -D stream.reduce.output.field.separator=. \
        -D stream.num.reduce.output.key.fields=2 \
        -files identity_mr.py \
        -mapper 'python identity_mr.py' \
        -reducer 'python identity_mr.py' \
        -numReduceTasks 2 \
        -input subnet.txt \
        -output subnet_out \
```

output

```
cat subnet_out/*
2    3.4    5
4    5.6    7
1    2.3    4
3    4.5    6
```

Streaming.jar Partitioner

```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D map.output.key.field.separator=. \
        -D mapreduce.partition.keypartitioner.options=-k1.2,1.2 \
        -files identity_mr.py \
        -mapper 'python identity_mr.py' \
        -reducer 'python identity_mr.py' \
        -numReduceTasks 2 \
        -input subnet.txt \
        -output subnet_out \
        -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

Streaming.jar Partitioner

input

```
$ cat subnet.txt
1a.2.3.4
2a.3.4.5
3b.4.5.6
4b.5.6.7
```

man sort | grep KEYDEF

```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D map.output.key.field.separator=. \
        -D mapreduce.partition.keypartitioner.options=-k1.2,1.2 \
        -files identity_mr.py \
        -mapper 'python identity_mr.py' \
        -reducer 'python identity_mr.py' \
        -numReduceTasks 2 \
        -input subnet.txt \
        -output subnet_out \
        -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

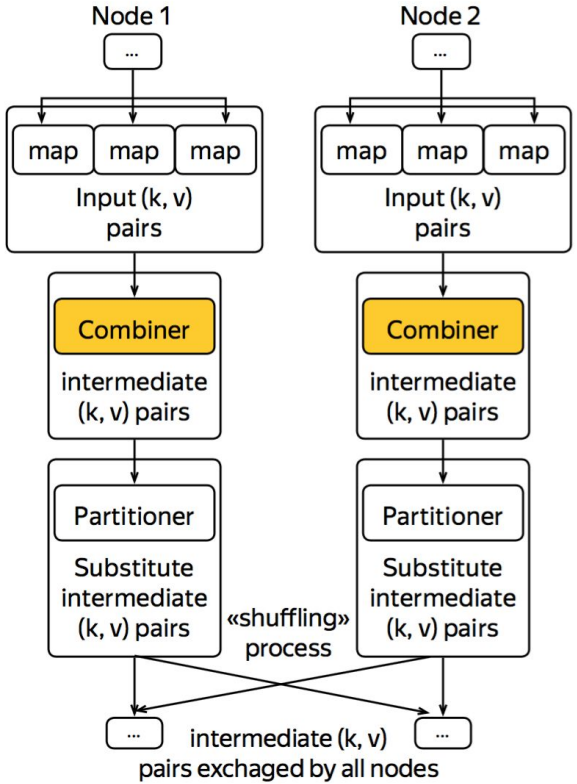
output

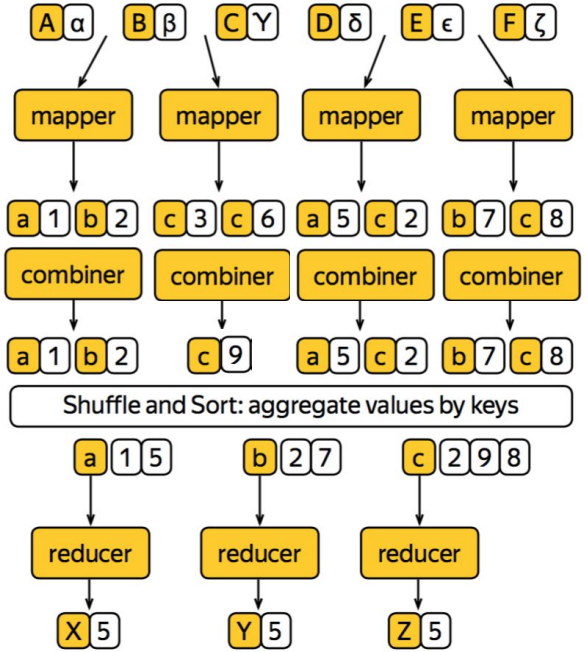
```
cat subnet_out/*
3b.4.5.6
4b.5.6.7
1a.2.3.4
2a.3.4.5
```

```
ls -lth subnet_out
total 8.0K
0 Apr 1 14:59 _SUCCESS
20 Apr 1 14:59 part-00001
20 Apr 1 14:59 part-00000
```

BIGDATA TEAM

Уточненная модель MapReduce

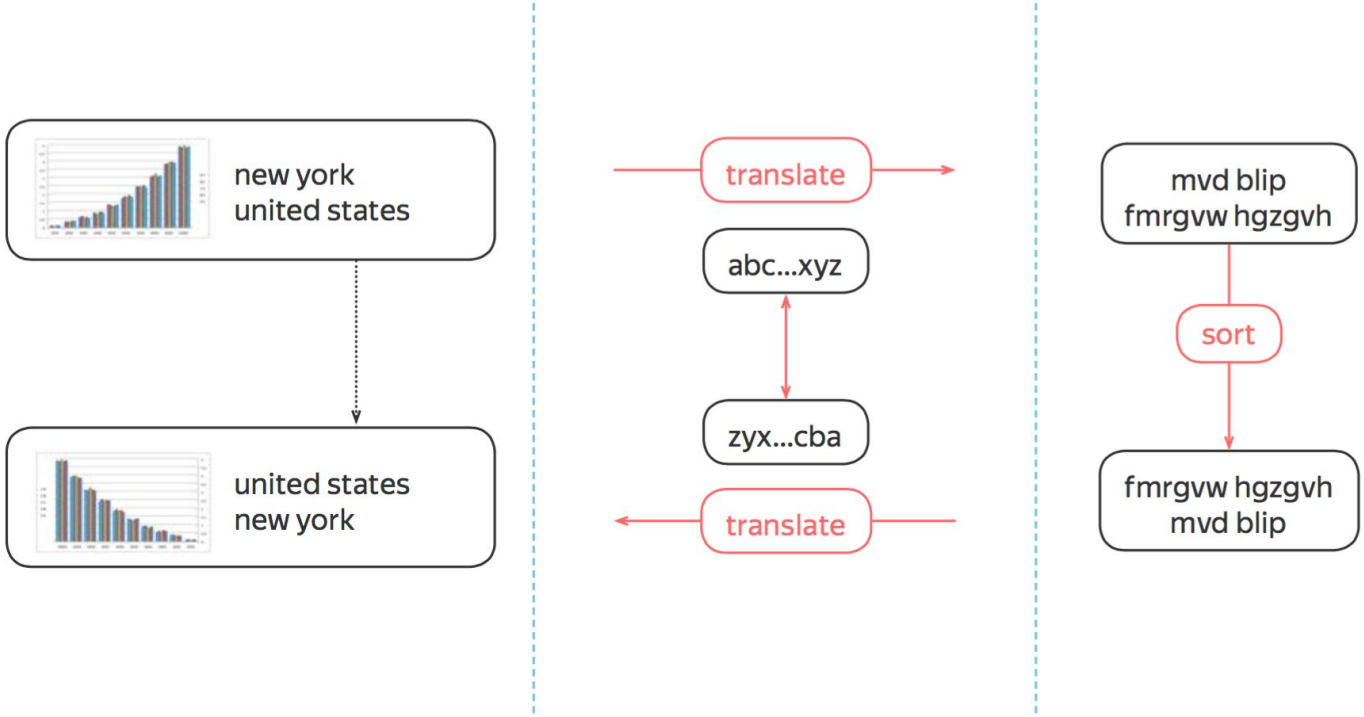




World of MapReduce

Combiner Partitioner Comparator

Problem Statement



Comparator Usage

input

1.4.3.4
2.4.4.5
3.2.5.6
4.1.6.7
5.12.6.7

```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.KeyFieldBasedComparator \
        -D mapreduce.map.output.key.field.separator=. \
        -D mapreduce.partition.keycomparator.options="-k2,2 -k3,3r" \
        -files identity_mr.py \
        -mapper 'python identity_mr.py' \
        -reducer 'python identity_mr.py' \
        -numReduceTasks 1 \
        -input subnet.txt \
        -output subnet_out \
```

output

4.1.6.7
5.12.6.7
3.2.5.6
2.4.4.5
1.4.3.4

Резюме

- Вы знаете как написать MapReduce Streaming приложение с помощью Bash и Python
- Вы умеете использовать Distributed Cache для передачи файлов и архивов, а также знаете сколько копий будет распространено по кластеру
- Зная вход и выход Mapper и Reducer Вы можете определить сигнатуру Combiner
- Вы знаете когда стоит, а когда не стоит использовать Combiner
- Вы знаете, каким образом использовать Partitioner и Comparator в MapReduce Streaming приложения

Thank you! Questions?

Feedback: http://rebrand.ly/mf2019q2_feedback_03_mro

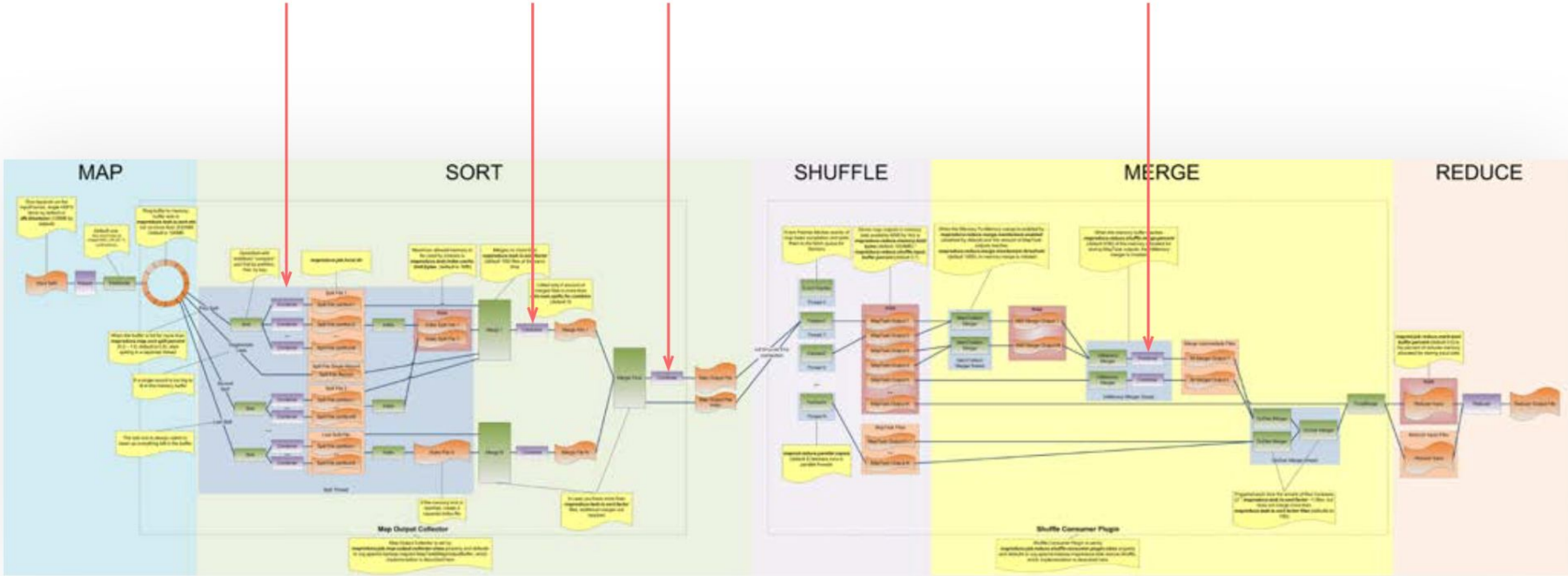
Dral Alexey, aadral@bigdatateam.org
CEO at BigData Team, http://bigdatateam.org/
https://www.linkedin.com/in/alexey-dral
https://www.facebook.com/bigdatateam/

Документация по Hadoop Streaming

https://hadoop.apache.org/docs/r1.2.1/streaming.html

BIGDATA TEAM

Combiner: реальность



https://0x0fff.com/hadoop-mapreduce-comprehensive-description/