

MapReduce Optimization

Алексей Драль, aadral@bigdatateam.org

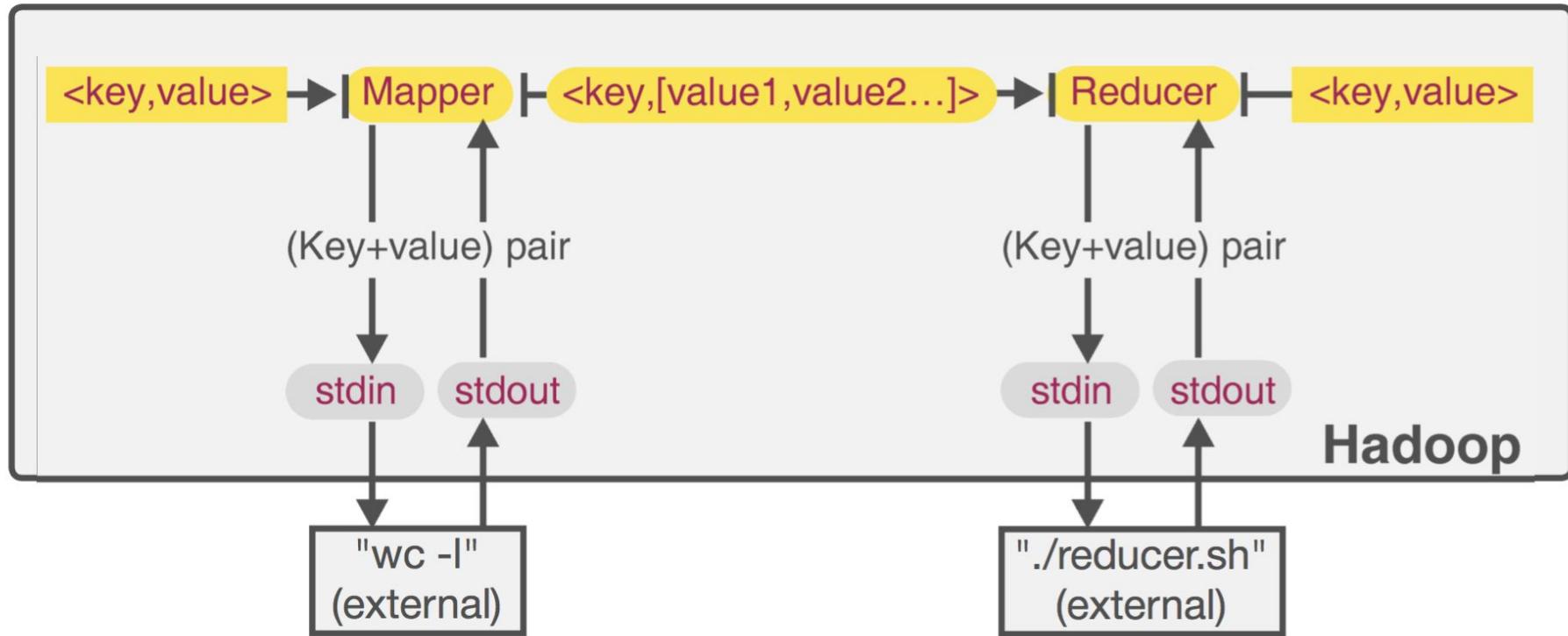
CEO at BigData Team, <http://bigdatateam.org>

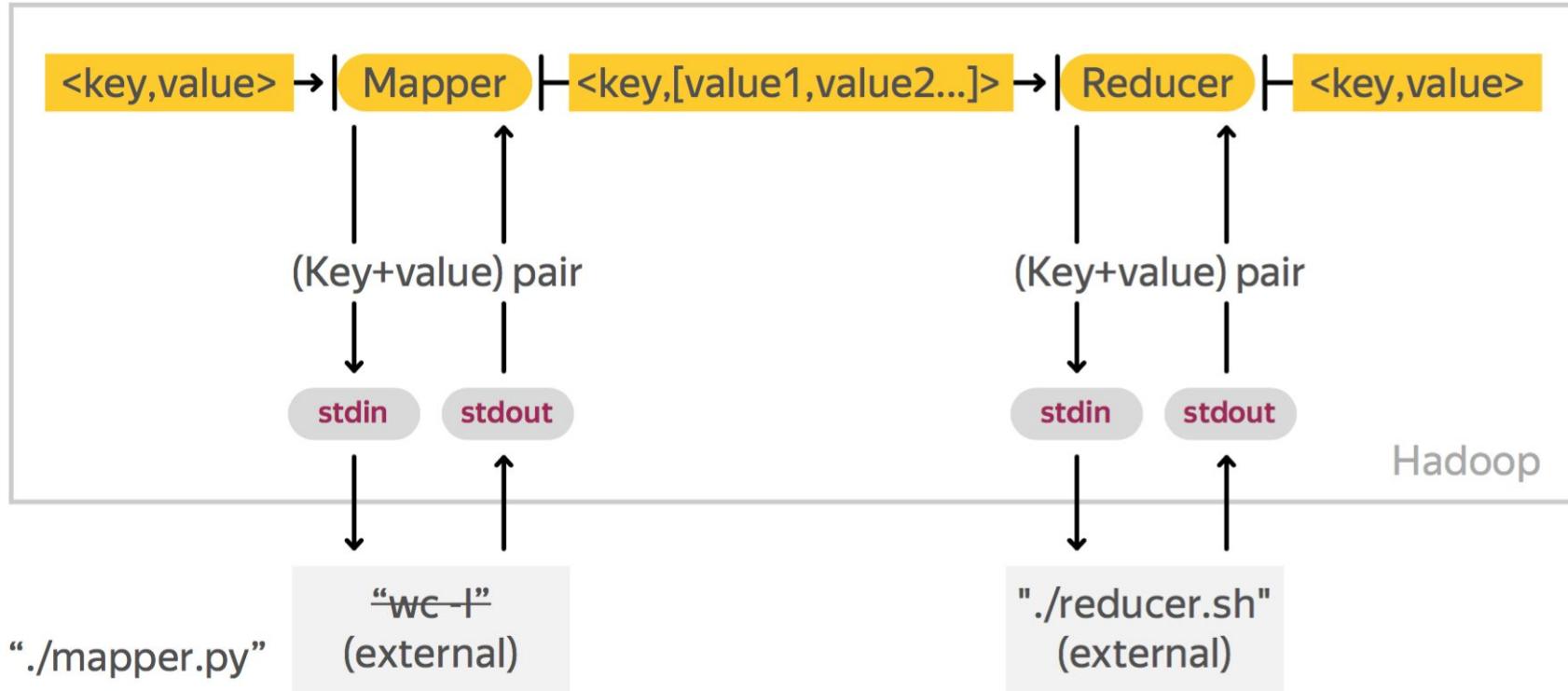
<https://www.facebook.com/bigdatateam>



04.03.2021, Moscow, Russia

- ▶ Проверка знаний (Kahoot!)
- ▶ MapReduce LineCount и WordCount WarmUp
- ▶ Распределенный кеш (Distributed Cache)
- ▶ Workshop
- ▶ -- перерыв, Q&A
- ▶ Combiner, Partitioner, Comparator







Mapper (Python): mapper.py

```
from __future__ import print_function
import sys

line_count = sum(1 for _ in sys.stdin)

print(line_count)
```



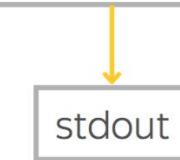


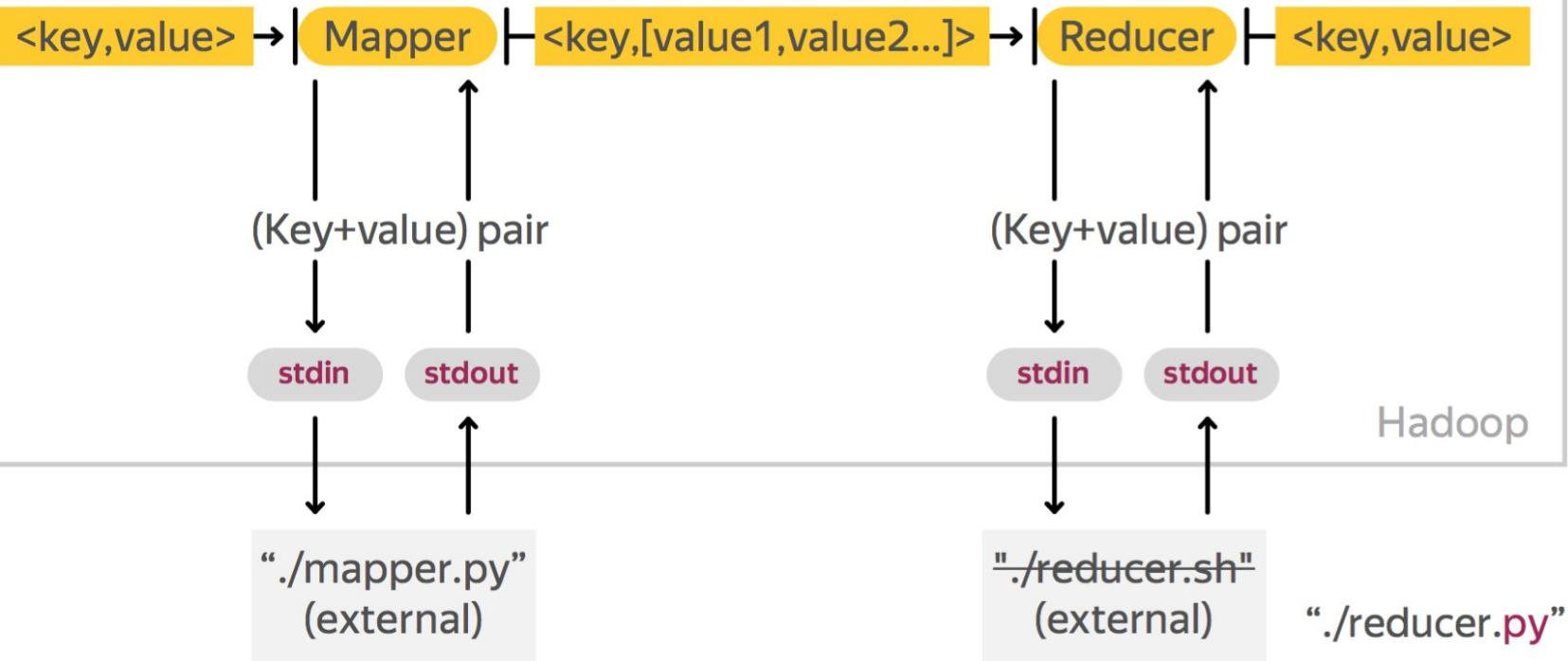
Mapper (Python): mapper.py

```
from __future__ import print_function
import sys

line_count = 0
for line in sys.stdin:
    line_count += 1

print(line_count)
```







Mapper(Python): reducer.py

```
from __future__ import print_function
import sys

line_count = sum(
    int(value) for value in sys.stdin
)

print(line_count)
```

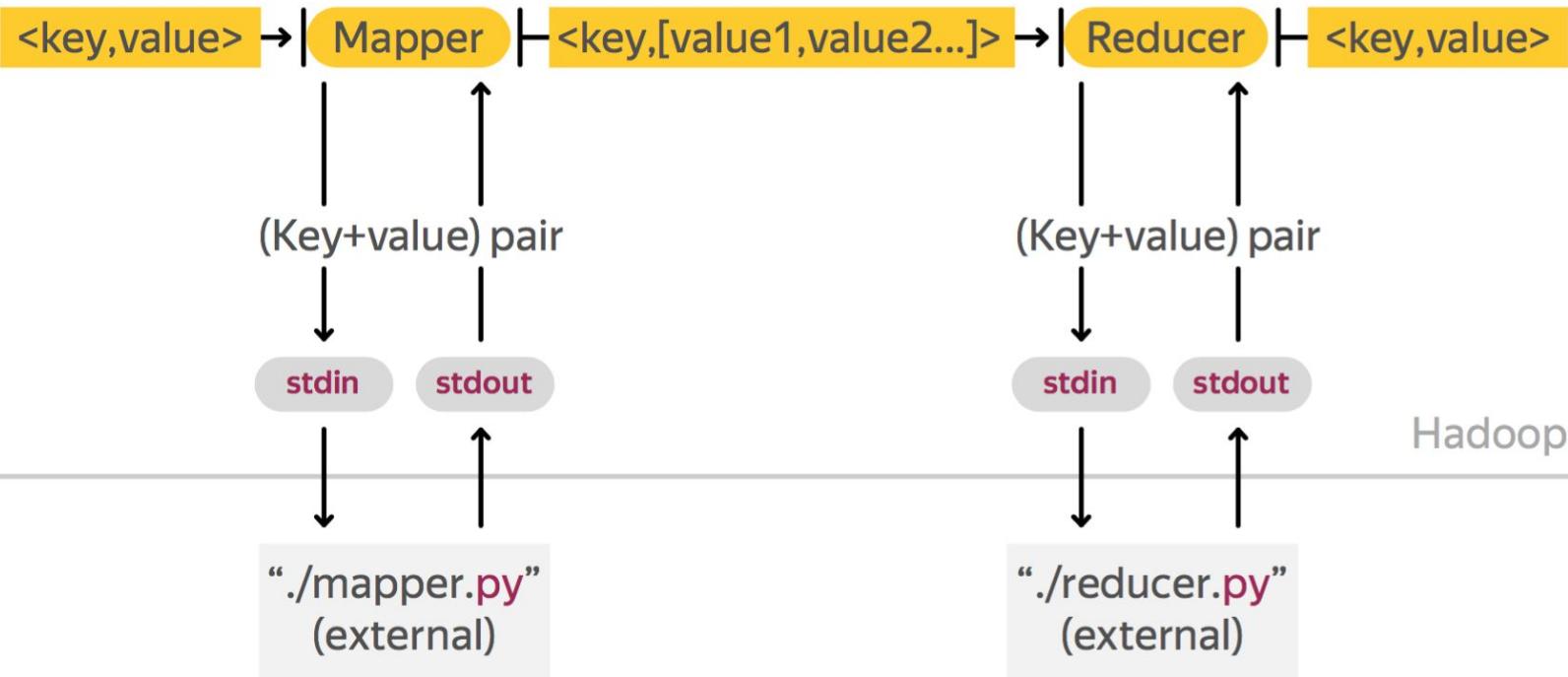




```
HADOOP_STREAMING_JAR="/path/to/hadoop-streaming.jar"  
yarn jar $HADOOP_STREAMING_JAR \  
  -files mapper.py,reducer.py \  
  -mapper 'python mapper.py' \  
  -reducer './reducer.py' \  
  -numReduceTasks 1 \  
  -input /data/wiki/en_articles \  
  -output wc_mr_with_reducer
```



Py-friendly Line Count





**BIGDATA
TEAM**

MapReduce Word Count



Apache Hadoop (/həˈdu: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...

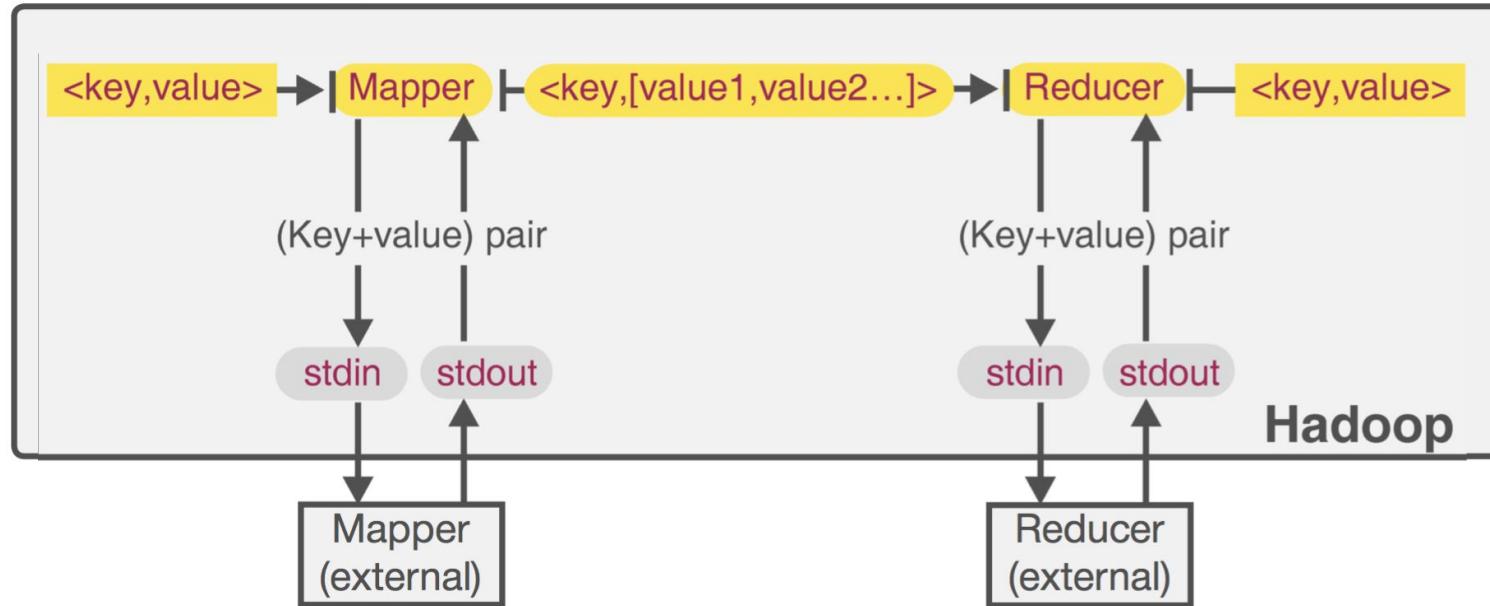


'the': 3, 'of': 3, 'hadoop': 2, ...



WIKIPEDIA
The Free Encyclopedia





Mapper:

- ▶ Как данные читаем (input format)
- ▶ Как данные обрабатываем
- ▶ Как данные выводим (output format)

Тоже, что и Mapper, плюс:

- ▶ Как агрегируем по ключам отсортированные данные



WIKIPEDIA

WordCount Mapper

<article_id> <tab> <article_content>

↑
key ↑
 value

mapper.py

```
#!/usr/bin/env python3
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    # ...
```

mapper.py

```
#!/usr/bin/env python3
import sys

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

mapper.py

```
#!/usr/bin/env python3
import sys

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



```
yarn jar $HADOOP_STREAMING_JAR \
  -files mapper.py \
  -mapper "python3 mapper.py" \
  -numReduceTasks 0 \
  -input /data/wiki/en_articles_part \
  -output word_count
```

```
$ hdfs dfs -ls -h word_count
```

Found 3 items

-rw-r--r--	3	aadral	hdfs	0	2021-03-02 10:24	word_count/_SUCCESS
-rw-r--r--	3	aadral	hdfs	83.6 M	2021-03-02 10:24	word_count/part-00000
-rw-r--r--	3	aadral	hdfs	12.1 M	2021-03-02 10:24	word_count/part-00001



```
$ hdfs dfs -tail word_count/part-00001 | head -5
```

ia,	1
Paris,	1
2005.	1
John	1
Newton,	1



mapper.py

```
#!/usr/bin/env python3
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")
```



```
yarn jar $HADOOP_STREAMING_JAR \
  -files mapper.py \
  -mapper "python3 mapper.py" \
  -numReduceTasks 0 +
  -input /data/wiki/en_articles_part \
  -output word_count
```

```
$ hdfs dfs -ls -h word_count
```

Found 3 items

-rw-r--r--	3	aadral	hdfs	0	2021-03-02 10:46	word_count/_SUCCESS
-rw-r--r--	3	aadral	hdfs	94.1 M	2021-03-02 10:46	word_count/part-00000



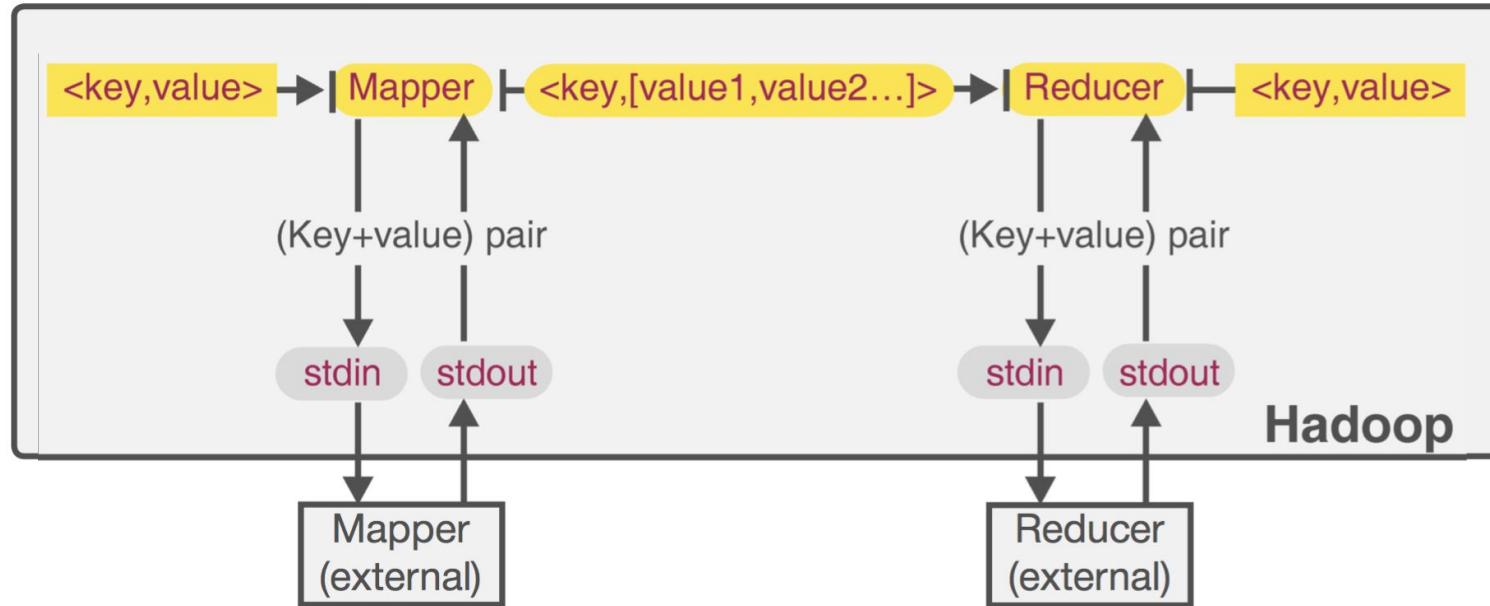
```
$ hdfs dfs -text word_count/part-00000 | head -5
```

```
0 1  
0 1  
0 1  
0 1  
0 1
```

```
text: Unable to write to output stream.
```

```
$ hdfs dfs -tail word_count/part-00000 | tail -5
```

```
□ 1  
□ 1  
□ 1  
□ 1  
□ 1
```



Mapper:

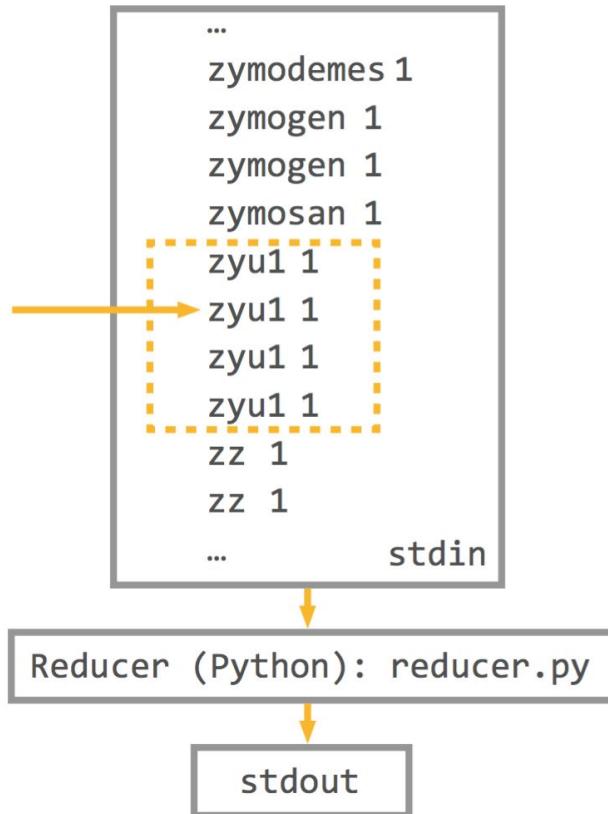
- ▶ Как данные читаем (input format)
- ▶ Как данные обрабатываем
- ▶ Как данные выводим (output format)

Тоже, что и Mapper, плюс:

- ▶ Как агрегируем по ключам отсортированные данные



Word Count Reducer





reducer.py

```
#!/usr/bin/env python3
import sys

current_word = None
word_count = 0

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



reducer.py

```
# ...

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

...	
zymosan	1
zyu1	1
zz	1
...	
stdin	



reducer.py

```
# ...
if word == current_word:
    word_count += counts
else:
    if current_word:
        print(current_word, word_count, sep="\t")
    current_word = word
    word_count = counts
# ...
```



...	
zymosan	1
zyu1	1
zz	1
...	
stdin	



reducer.py

```
# ...
for line in sys.stdin:
    # ...
if current_word:
    print(current_word, word_count, sep="\t")
```

```
...
zymosan 1
zyu1     1
zyu1     1
zyu1     1
zyu1     1
zyu1     1
zz       1
...
stdin
```





reducer.py

```
#!/usr/bin/env python3
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")
```





```
yarn jar $HADOOP_STREAMING_JAR \
  -files mapper.py,reducer.py \
  -mapper "python3 mapper.py" \
  -reducer "python3 reducer.py" \
  -input /data/wiki/en_articles_part \
  -output word_count
```

```
$ hdfs dfs -ls -h word_count
```

Found 3 items

-rw-r--r--	3	aadral	hdfs	0	2021-03-02 11:21	word_count/_SUCCESS
-rw-r--r--	3	aadral	hdfs	3.4 M	2021-03-02 11:21	word_count/part-00000

```
$ hdfs dfs -text word_count/part-00000 | head -5
```

```
0          14891
00         844
000        8186
0000       55
00000      5
```

text: Unable to write to output stream.



**BIGDATA
TEAM**

Где мы сейчас:

ТО САМОЕ ОЩУЩЕНИЕ

**КОГДА ЗАЛОЖИЛ
ФУНДАМЕНТ BIG DATA**

risovach.ru

Можно мне чуточку

Оптимизации

memesmix.net

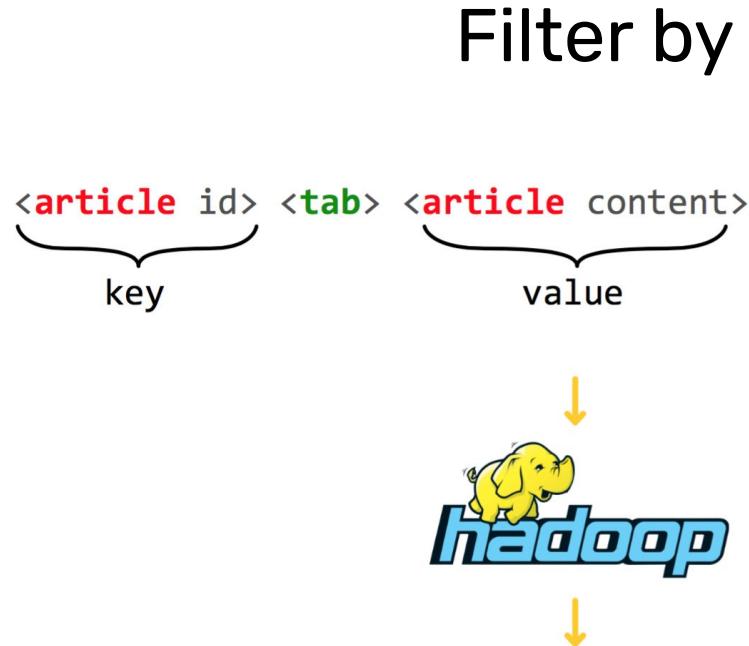


**BIGDATA
TEAM**

Distributed Cache



WIKIPEDIA
The Free Encyclopedia



TOP 10 NAMES	
GIRLS	BOYS
1 Olivia	1 Muhammad
2 Sophia	2 Oliver
3 Lily	3 Jack
4 Emily	4 Noah
5 Amelia	5 Jacob
6 Chloe	6 Harry
7 Isabelle	7 Charlie
8 Sophie	8 Ethan
9 Ella	9 James
10 Isabella	10 Thomas

Figures for 2015 Source: BabyCentre

...
zymodemes 1
zymogen 2
zymosan 1
zyu1 4
zz 2



```
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")
```



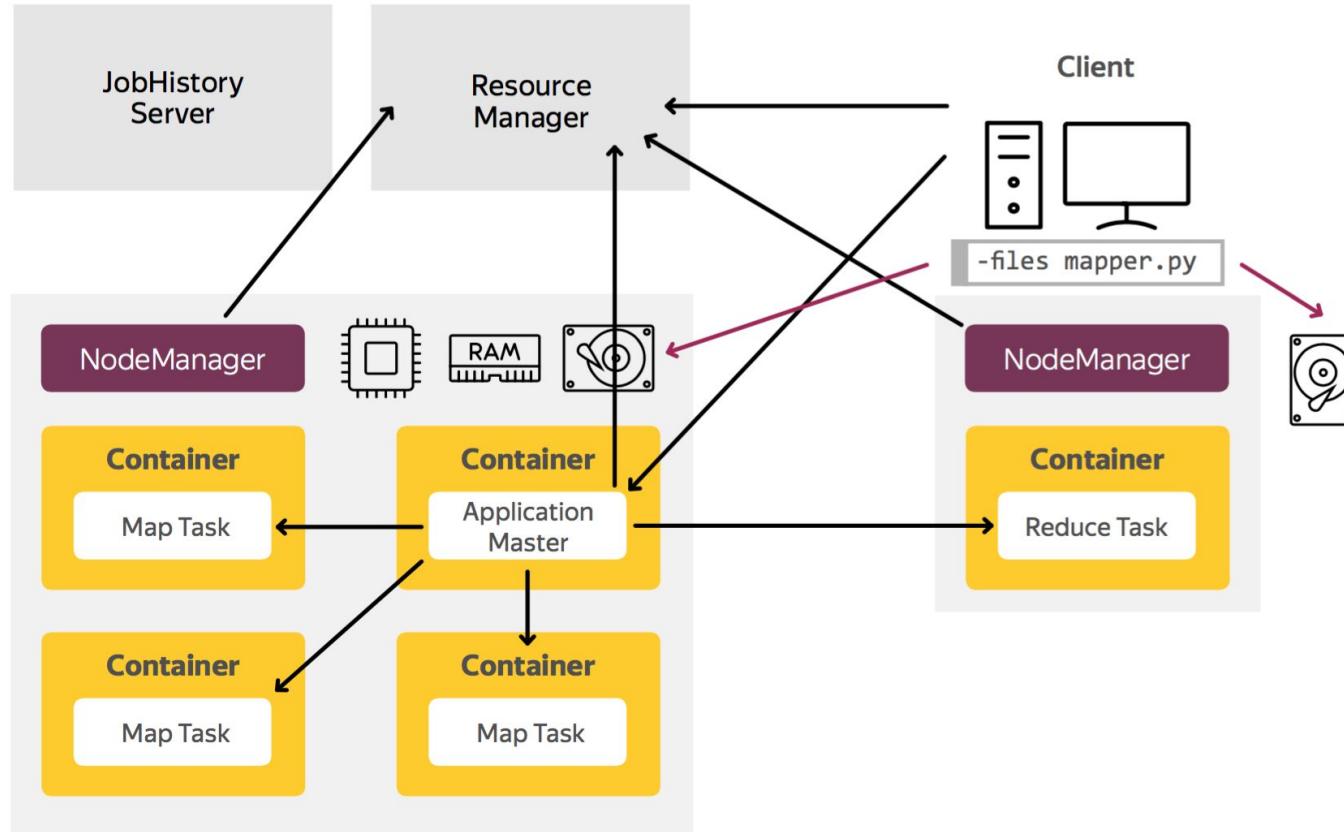
Filter by Vocabulary

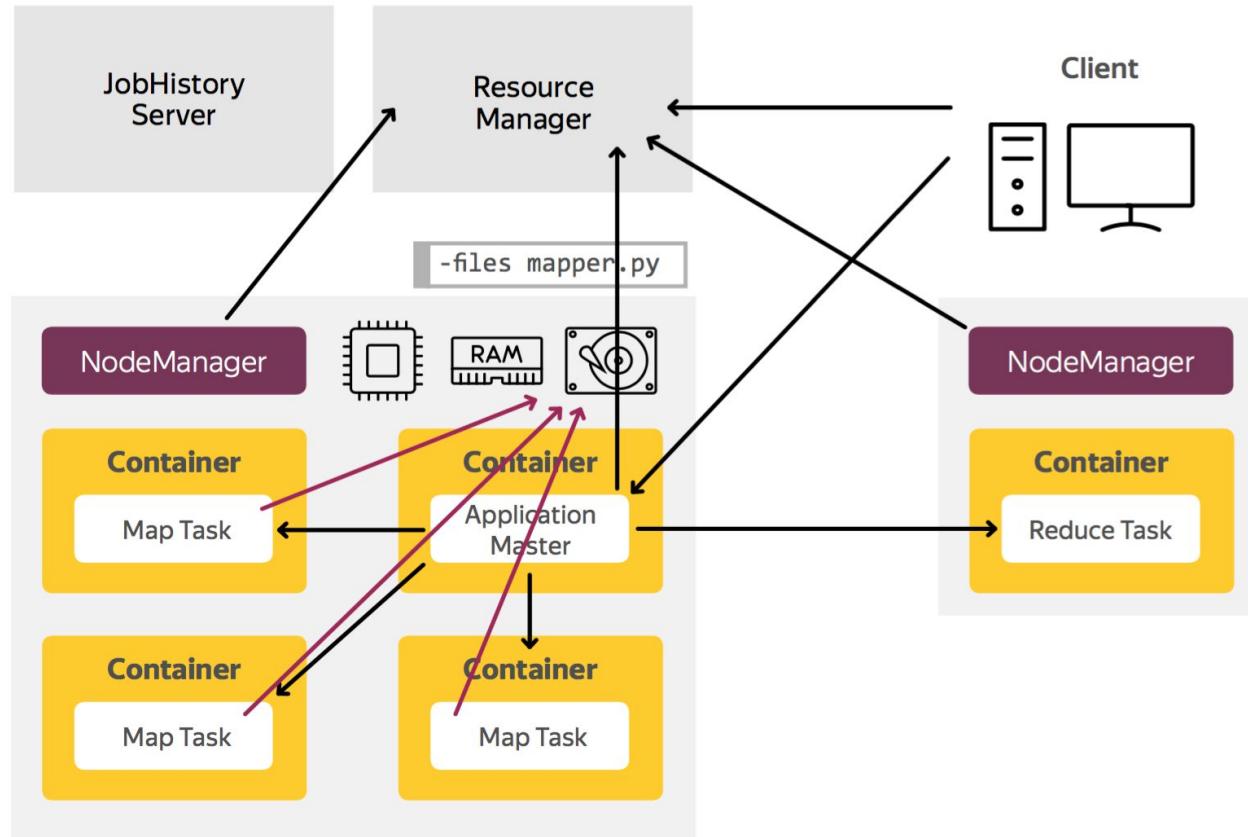
```
yarn jar $HADOOP_STREAMING_JAR \
  -files mapper.py,reducer.py,vocabulary.txt \
  -mapper 'python mapper.py' \
  -reducer 'python reducer.py' \
  -input /data/wiki/en_articles \
  -output word_count
```

Mapper (Python): mapper.py

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

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



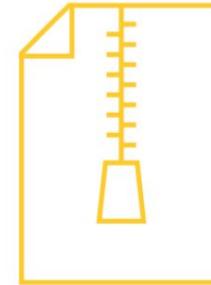




Distributed Cache Options



-files



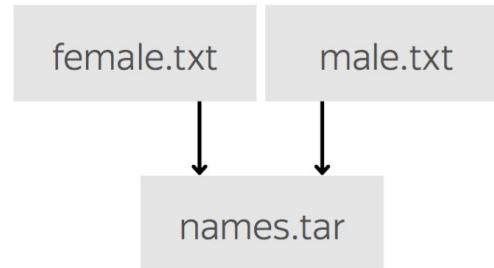
-archives



-libjars



Birth Names Example



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



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

names.tar

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





```
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





Combiner

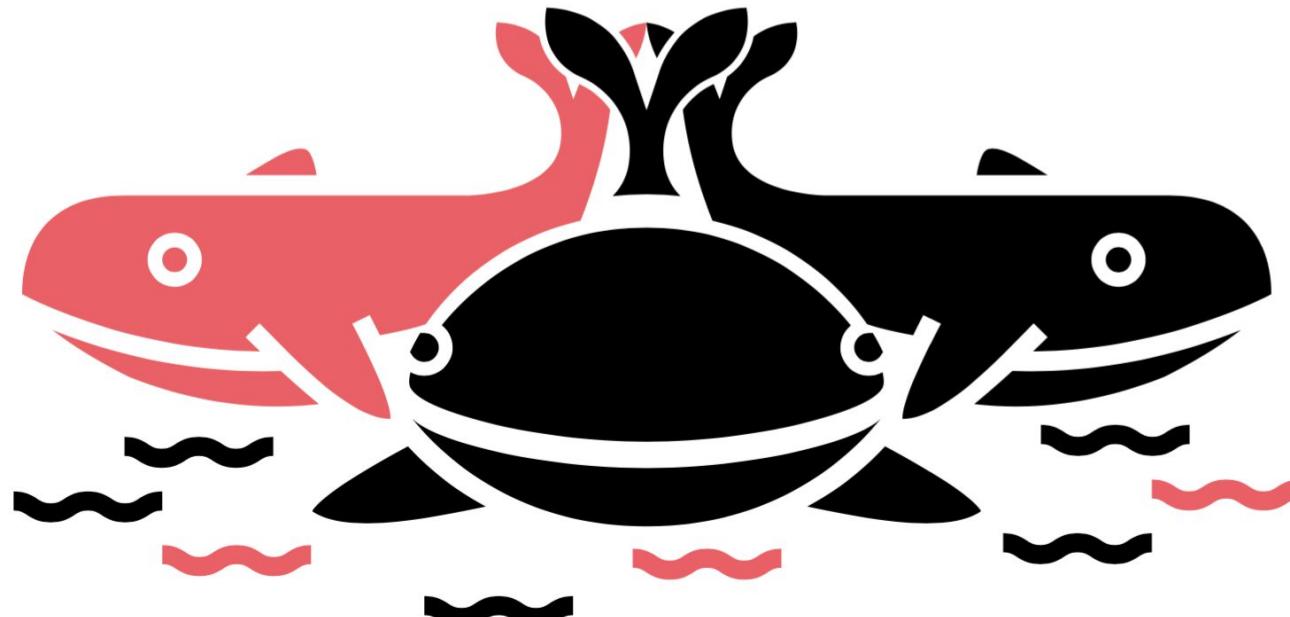
Partitioner

Comparator



BIGDATA
TEAM

Мир MapReduce



Combiner

Partitioner

Comparator



WIKIPEDIA
The Free Encyclopedia



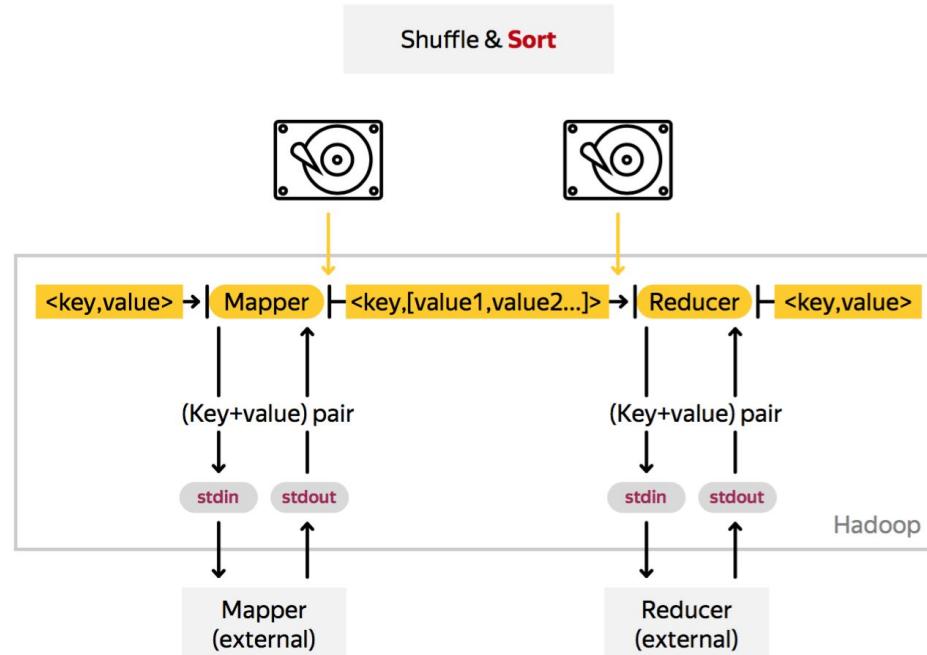
input: word word a word b c d word d e...

Mapper (Python): reporter_mapper.py

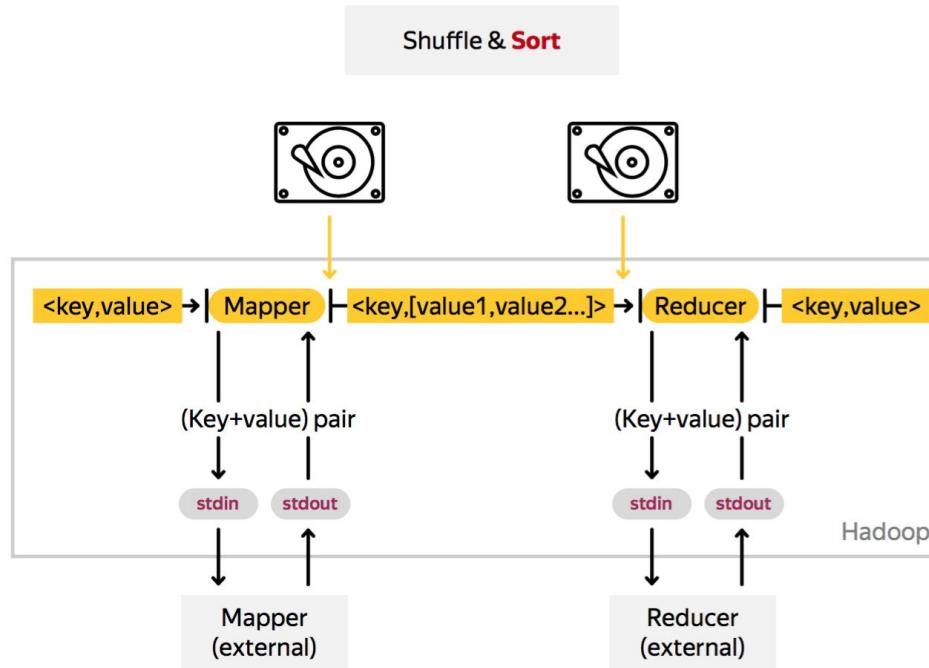
```
from __future__ import print_function
import sys

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), ...



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



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



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")
```

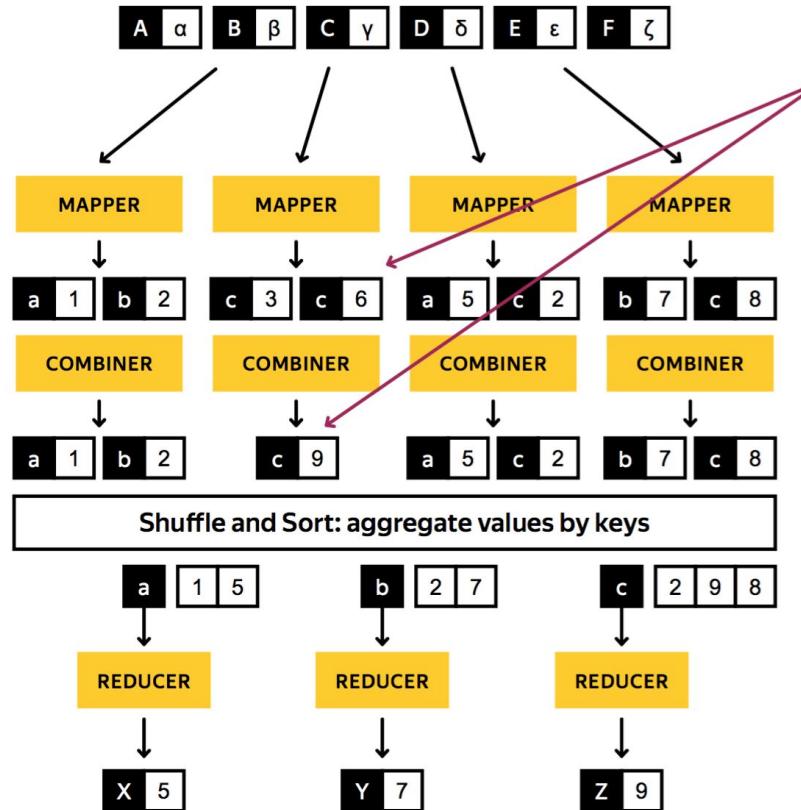
output: (a, 1), (b, 1), (c, 1), (d, 2), (e, 1), (word, 4), ...

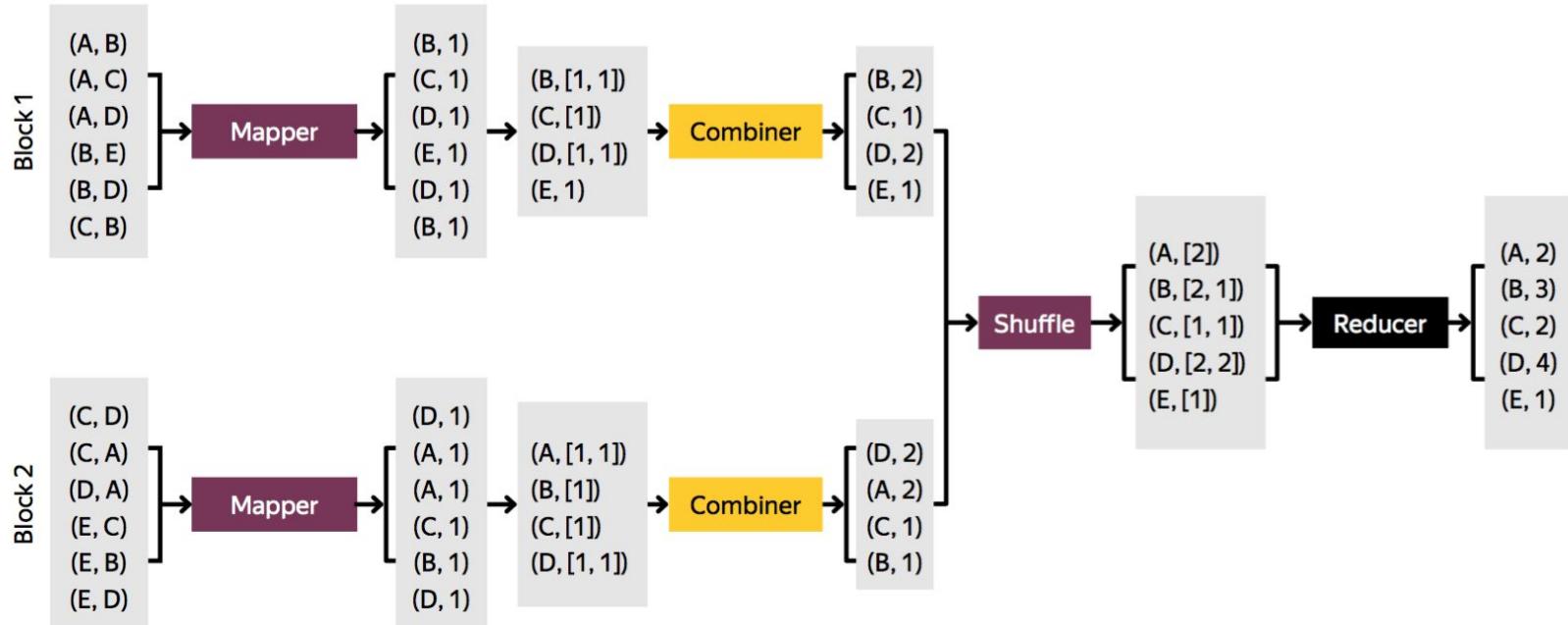


	without Combiner	with Combiner
Wall time (sec)	935	528
CPU time (sec)	9790	6584
Local FS Read (MB)	3006	1324
Local FS Write (MB)	4527	1963
Peak Map phys. memory (MB)	526	606
Peak Map virt. memory (MB)	2131	2144
Peak Reduce phys. memory (MB)	2744	631
Peak Reduce virt. memory (MB)	3196	3194



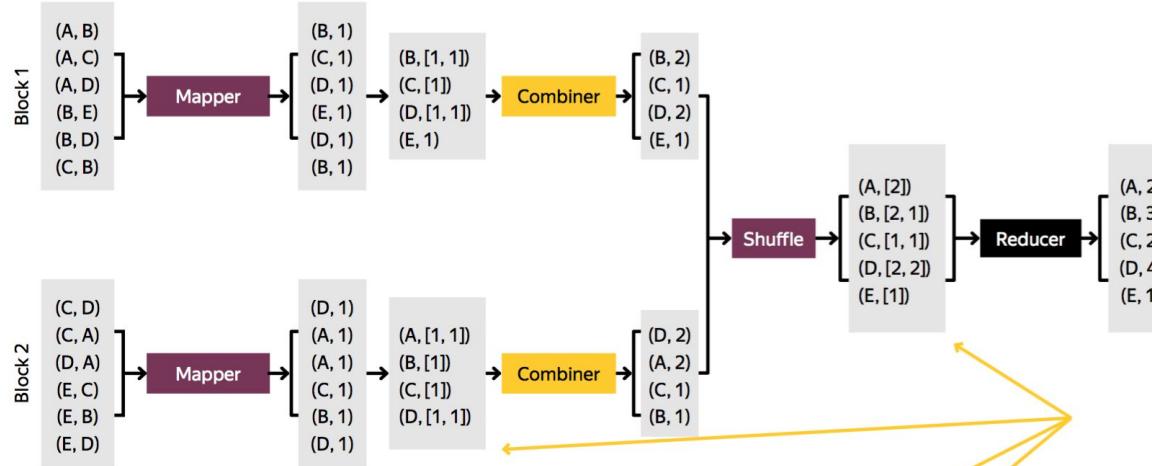
Уточненная модель MapReduce







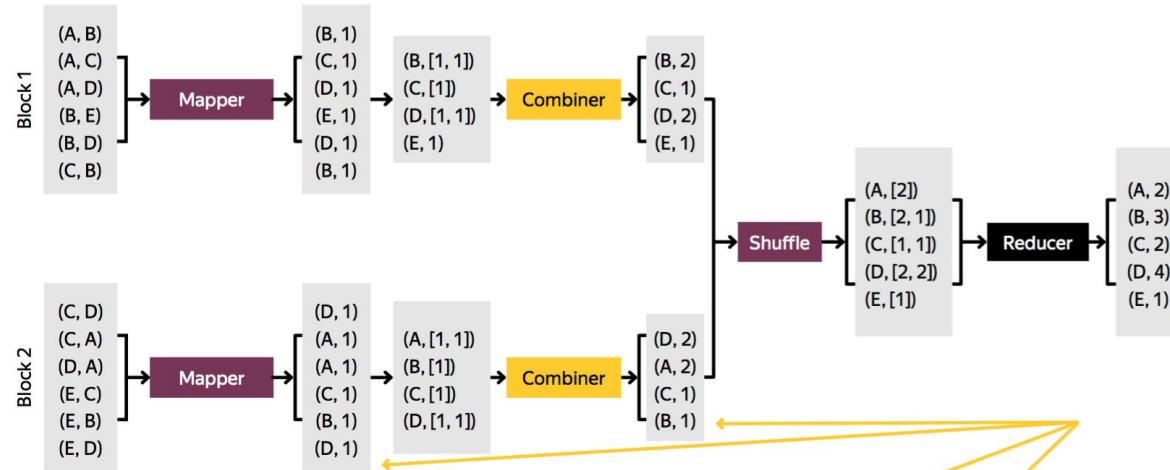
Формальная модель



- **read:** $[(k_{in}, v_{in}), \dots]$
- **map:** $(k_{in}, v_{in}) \rightarrow [(k_{interm}, v_{interm}), \dots]$
- **combiner:** $(k_{interm}, [v_{interm}, \dots]) \rightarrow [(k_{interm}, v_{interm}), \dots]$
- **Shuffle & Sort:** sort and group by k_{interm}
- **reduce:** $(k_{interm}, [v_{interm}, \dots]) \rightarrow [(k_{out}, v_{out}), \dots]$



Формальная модель



- read: $[(k_{in}, v_{in}), \dots]$
- map: $(k_{in}, v_{in}) \rightarrow [(k_{interm}, v_{interm}), \dots]$
- combiner: $(k_{interm}, [(v_{interm}, \dots)]) \rightarrow [(k_{interm}, v_{interm}), \dots]$
- Shuffle & Sort: sort and group by k_{interm}
- reduce: $(k_{interm}, [(v_{interm}, \dots)]) \rightarrow [(k_{out}, v_{out}), \dots]$



```
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 gribodov.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
```





WIKIPEDIA
The Free Encyclopedia

<article id> <tab> <article content>
key value

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



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

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), ...

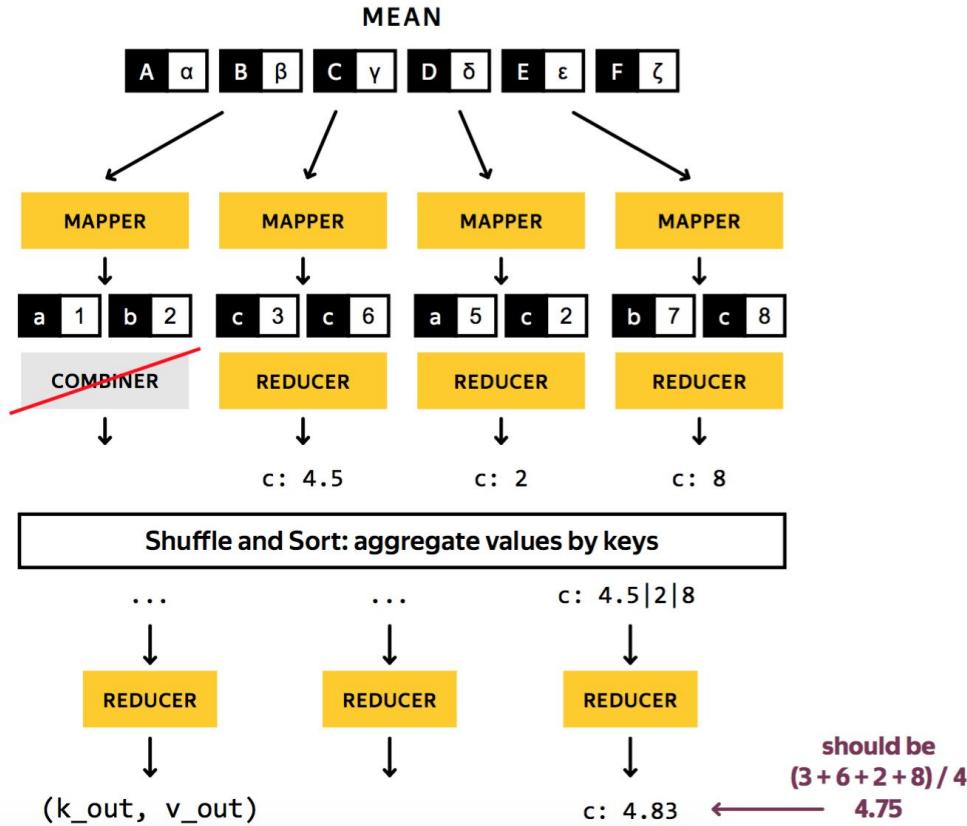
Mapper (Python): reducer.py

```
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")
```





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)), ...



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

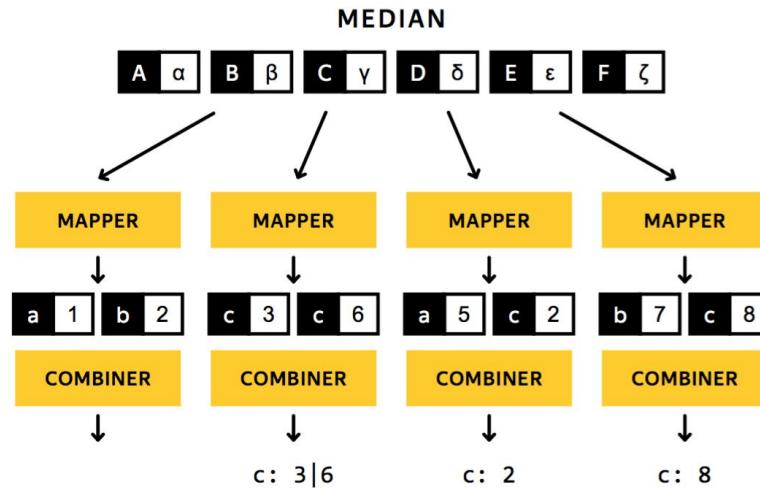
Mapper (Python): combiner.py

```
from __future__ import print_function
import sys

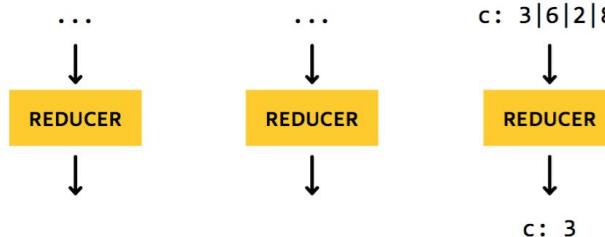
current_word = None
word_count, article_count = 0, 0

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")
```



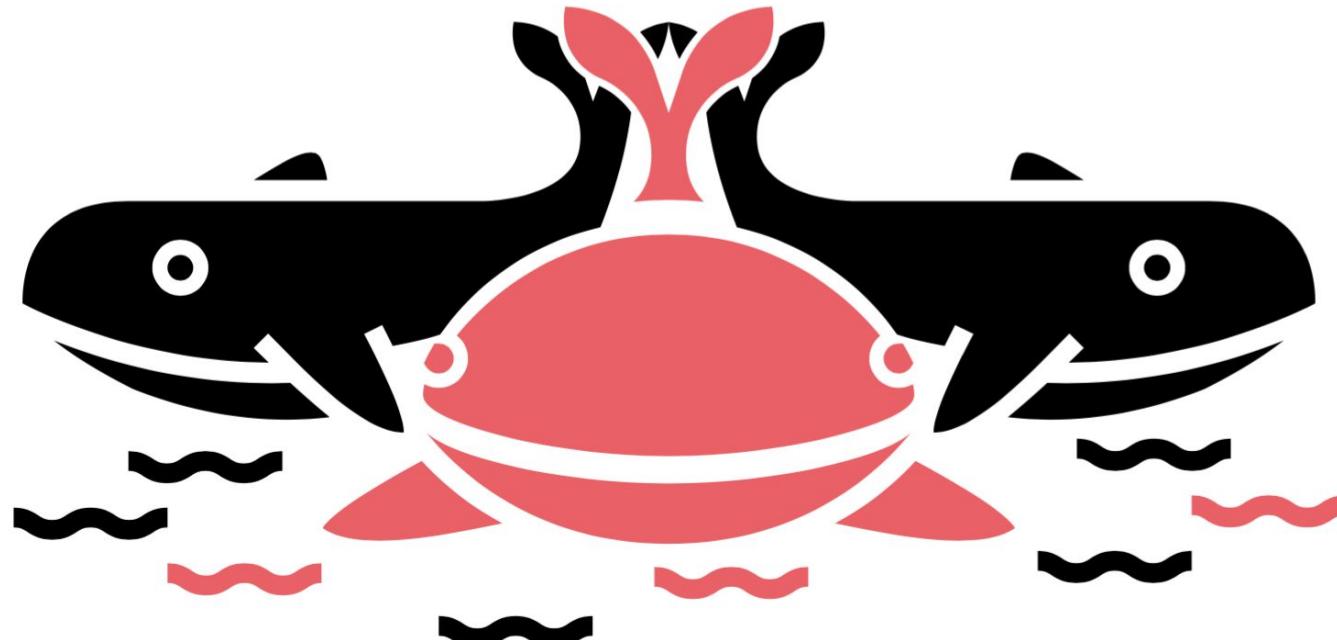
Shuffle and Sort: aggregate values by keys





BIGDATA
TEAM

Мир MapReduce



Combiner

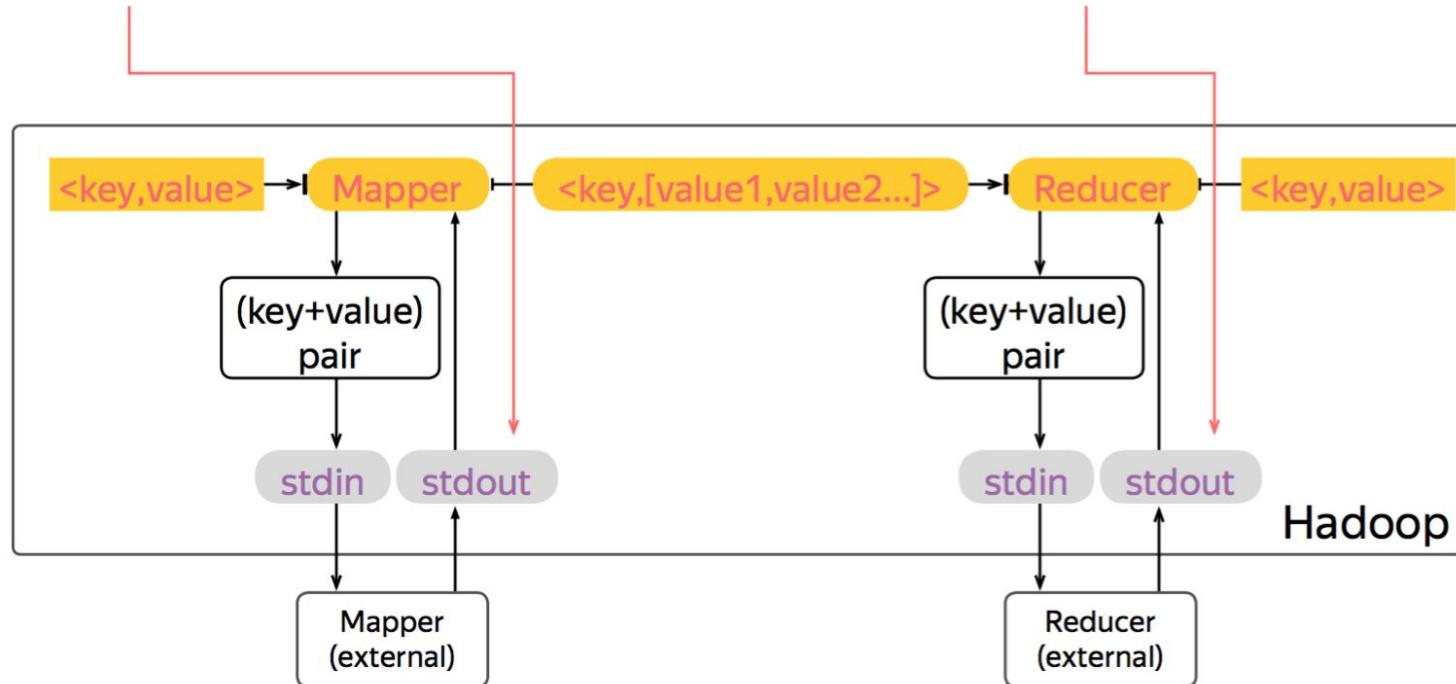
Partitioner

Comparator



-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



*reducer и combiner



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

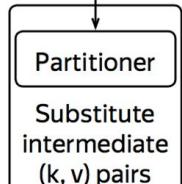
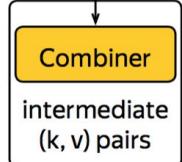
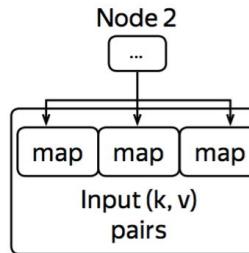
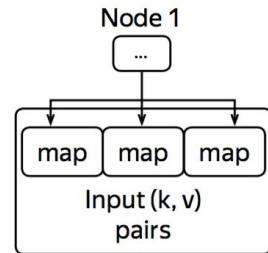
```
cat subnet_out/*
3b.4.5.6
4b.5.6.7
1a.2.3.4
2a.3.4.5
```

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
```

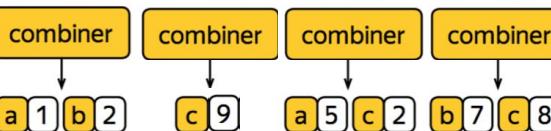
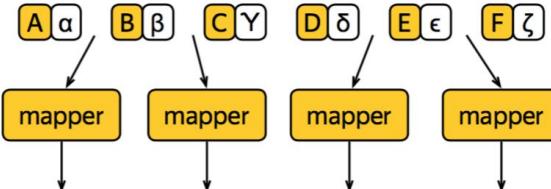


Уточненная модель MapReduce

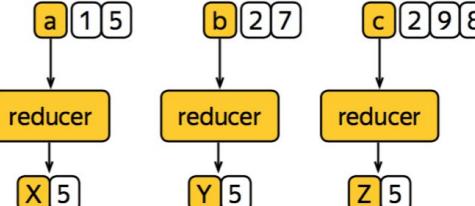


«shuffling» process

... intermediate (k, v) pairs exchanged by all nodes ...



Shuffle and Sort: aggregate values by keys





Combiner

Partitioner

Comparator



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



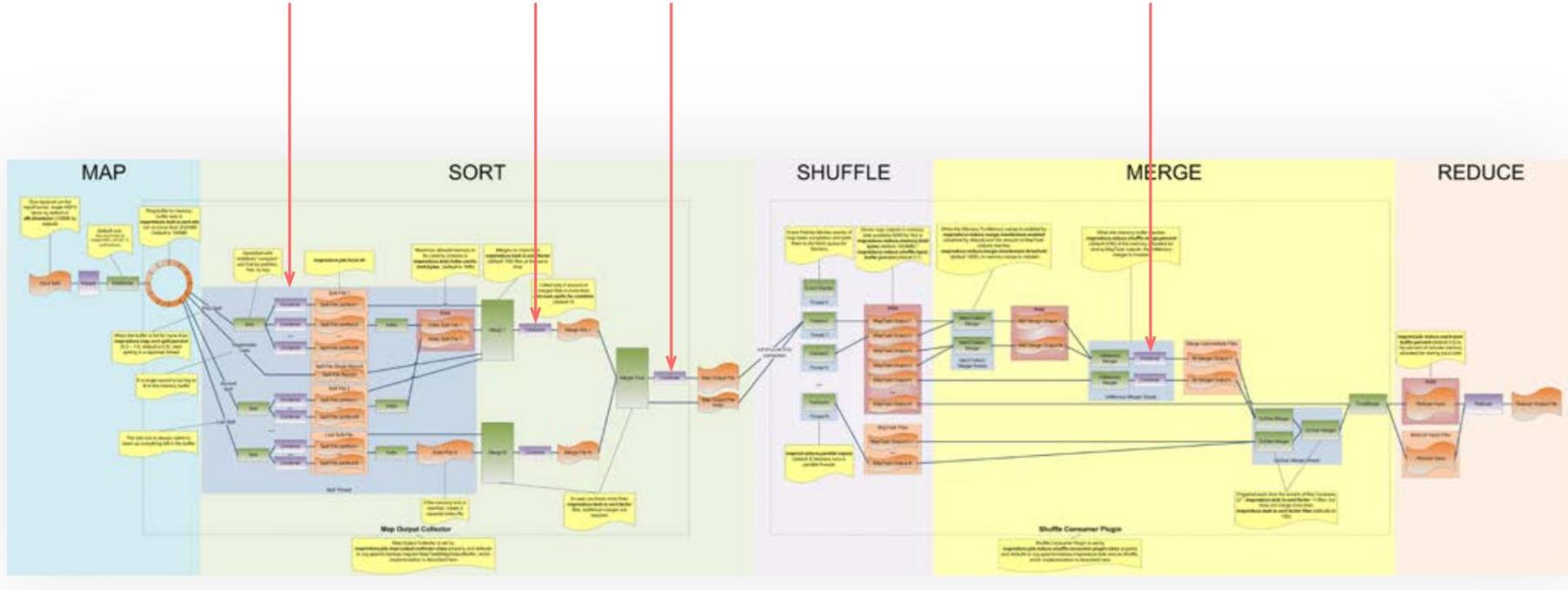
- ▶ Вы знаете как написать MapReduce Streaming приложение с помощью Bash и Python
- ▶ Вы умеете использовать Distributed Cache для передачи файлов и архивов, а также знаете сколько копий будет распространено по кластеру
- ▶ Зная вход и выход Mapper и Reducer Вы можете определить сигнатуру Combiner
-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

Документация по Hadoop Streaming

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



Combiner: реальность



Спасибо! Вопросы?

Feedback: http://rebrand.ly/x5bd2021q1_feedback_03_mro

Алексей Драль, aadral@bigdatateam.org

CEO at BigData Team, <http://bigdatateam.org>

<https://www.linkedin.com/in/alexey-dral>

<https://www.facebook.com/bigdatateam>