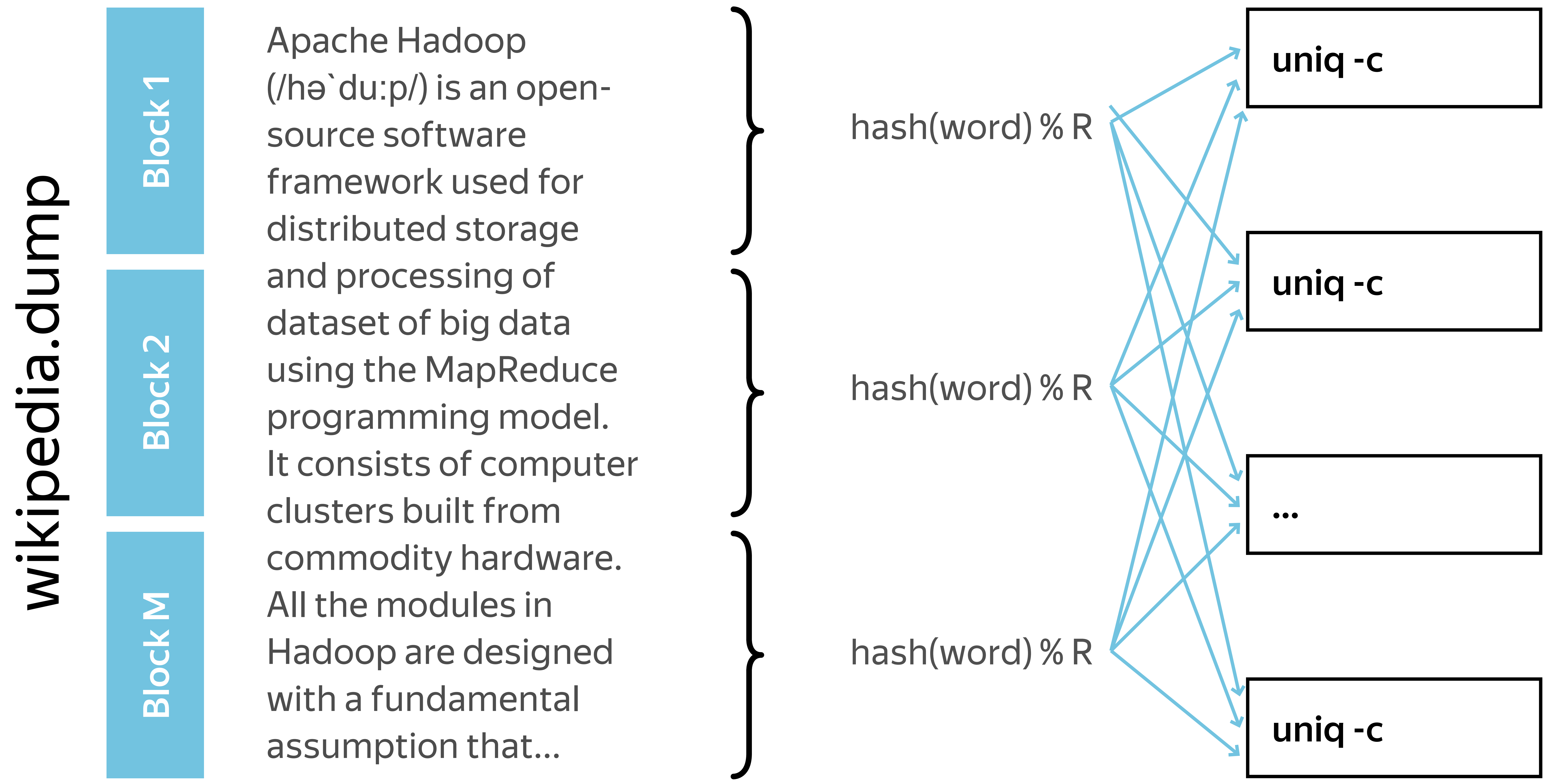
Yandex

MapReduce

Combiner

WordCount

```
cat wikipedia.dump | tr ' ' '\n'  |    sort     |       uniq -c
```


wikipedia.dump
Block 1
Block 2
Block M
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...
hash(word) % R
hash(word) % R
hash(word) % R
uniq -c
uniq -c
...
uniq -c

wikipedia.dump -> map () -> word   shuffle & sort   reduce()

$$\underbrace{\texttt{<article id>}}_{\texttt{key}}\ \texttt{<tab>}\ \underbrace{\texttt{<article content>}}_{\texttt{value}}$$

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

WIKIPEDIA
The Free Encyclopedia

$$\underbrace{\texttt{<article id>}}_{\texttt{key}}\ \texttt{<tab>}\ \underbrace{\texttt{<article content>}}_{\texttt{value}}$$

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

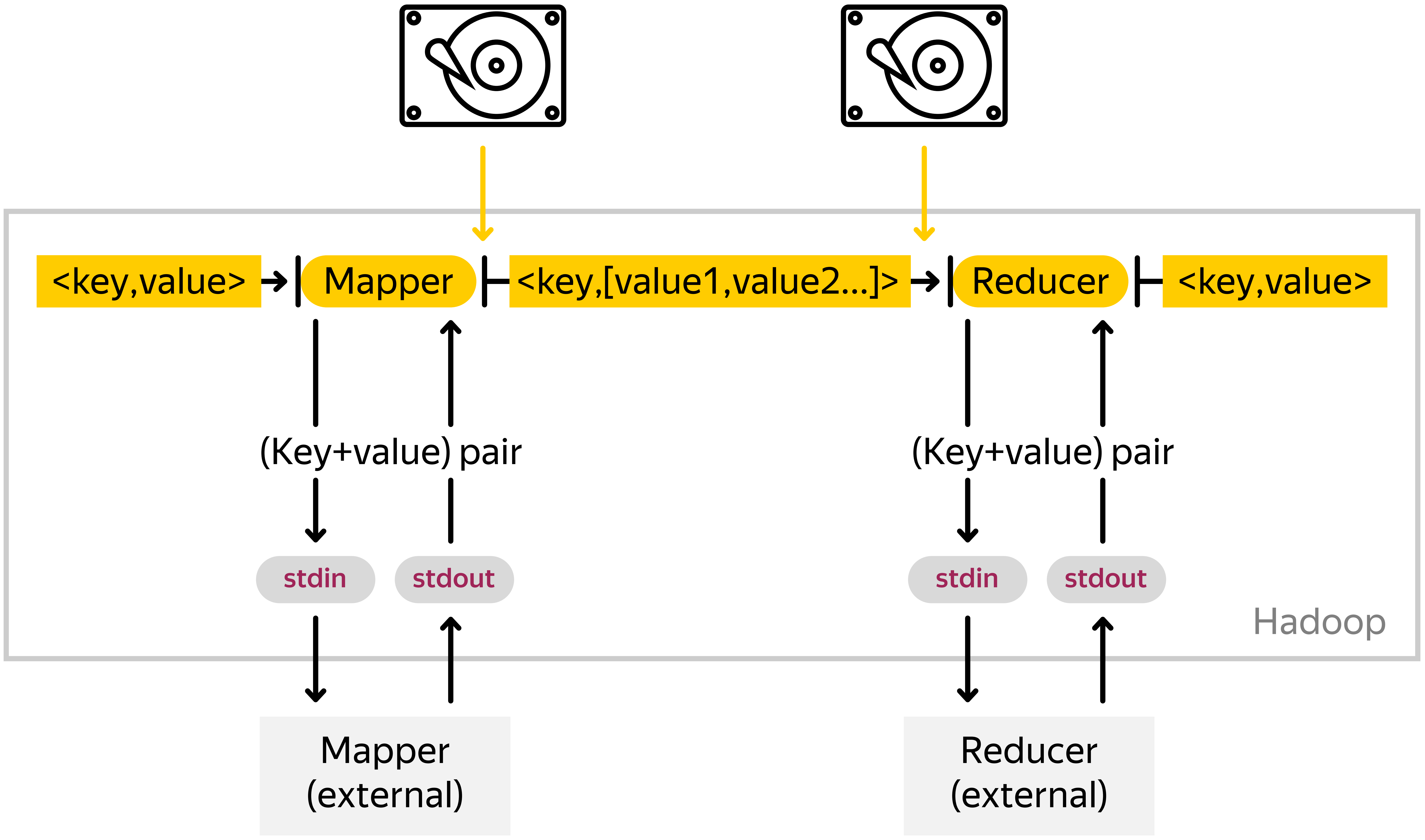
Mapper (Python): reporter_mapper.py

```python
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")
```

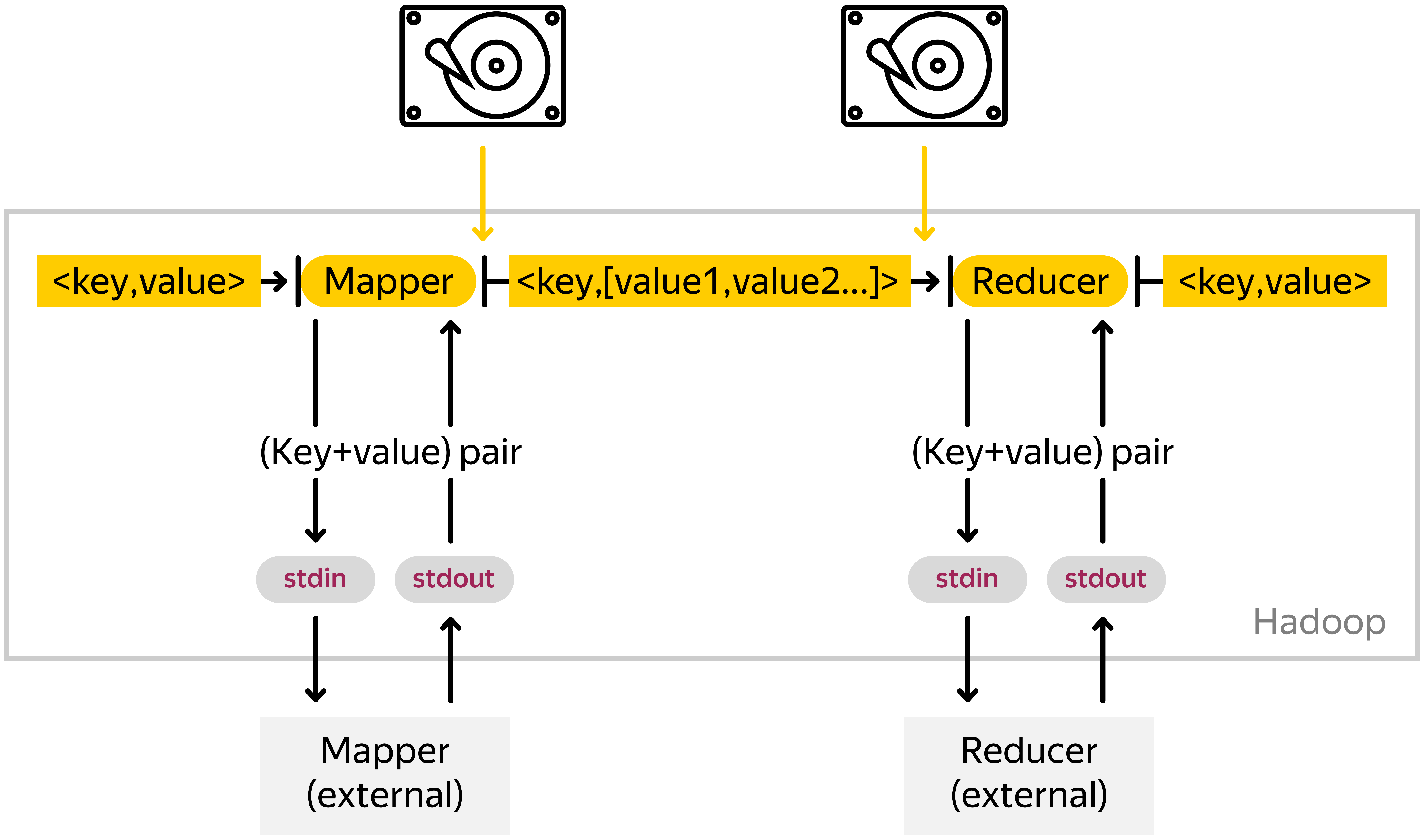
output: (word, 1), (word, 1), (a, 1), ...

Shuffle & **Sort**



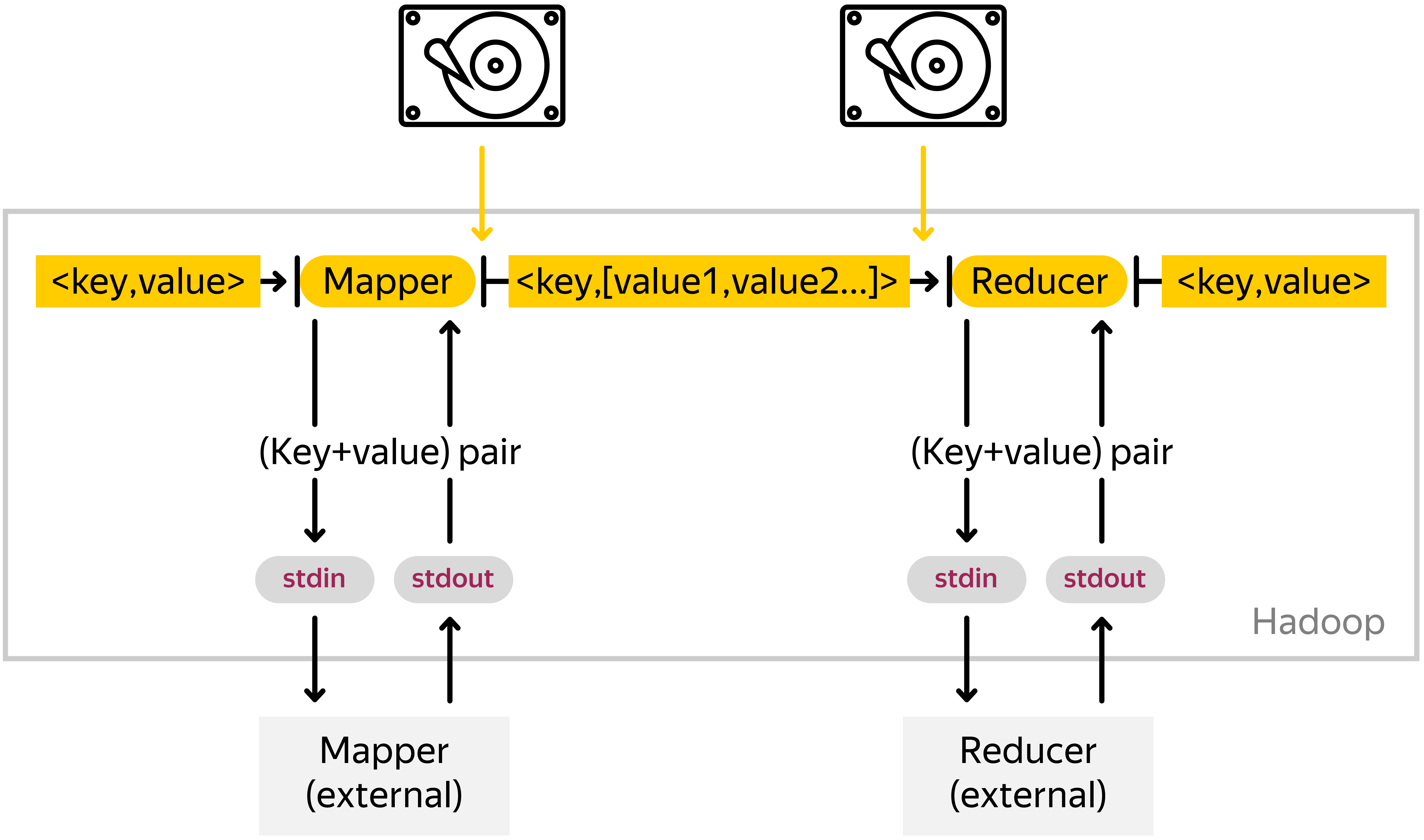
output: (word, 1), (word, 1), (a, 1), ...

Shuffle & **Sort**



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

Shuffle & **Sort**



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

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

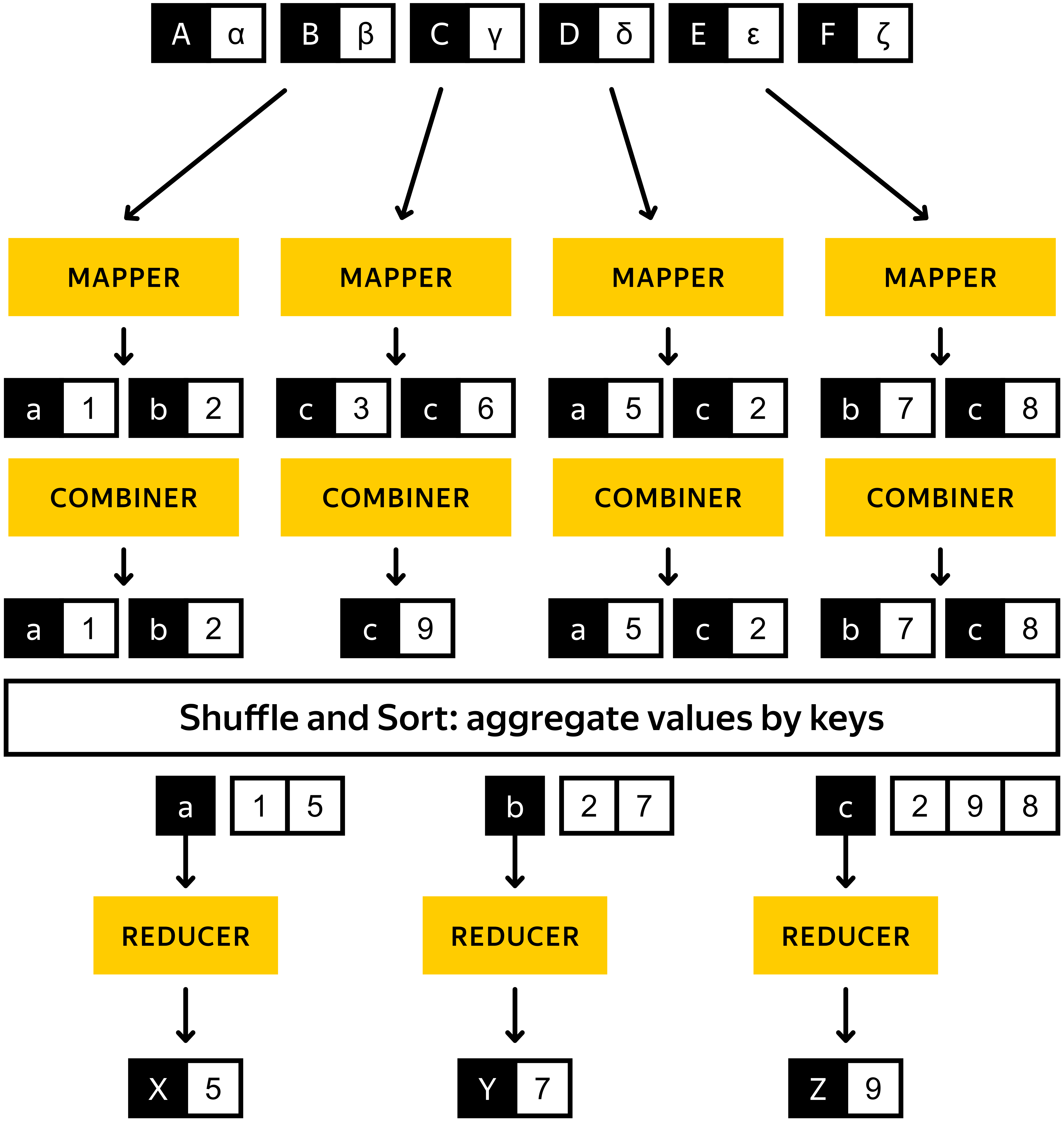
Mapper (Python): reporter_mapper.py

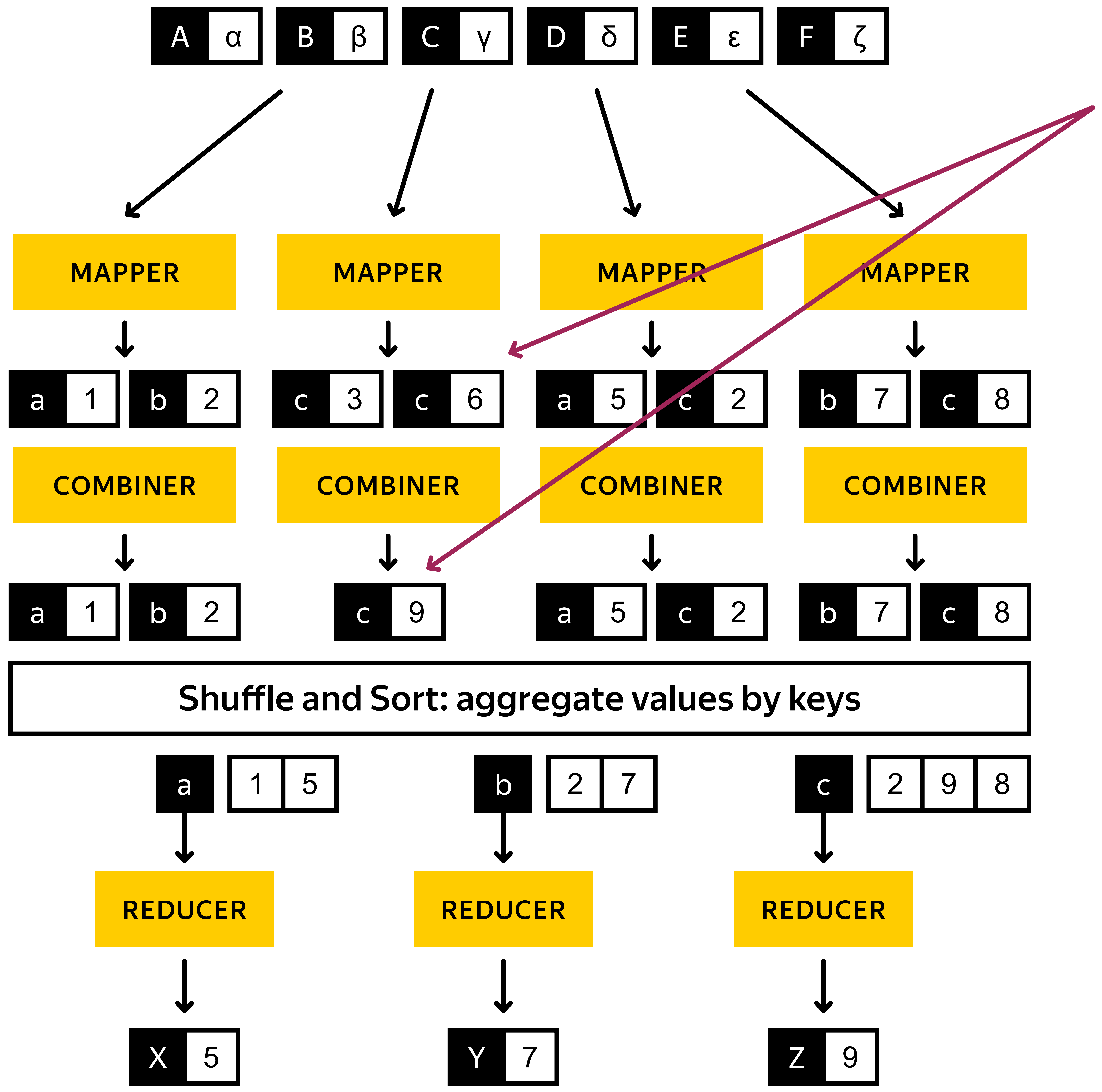
```
from __future__ import print_function
from collections import Counter
import sys

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")
```

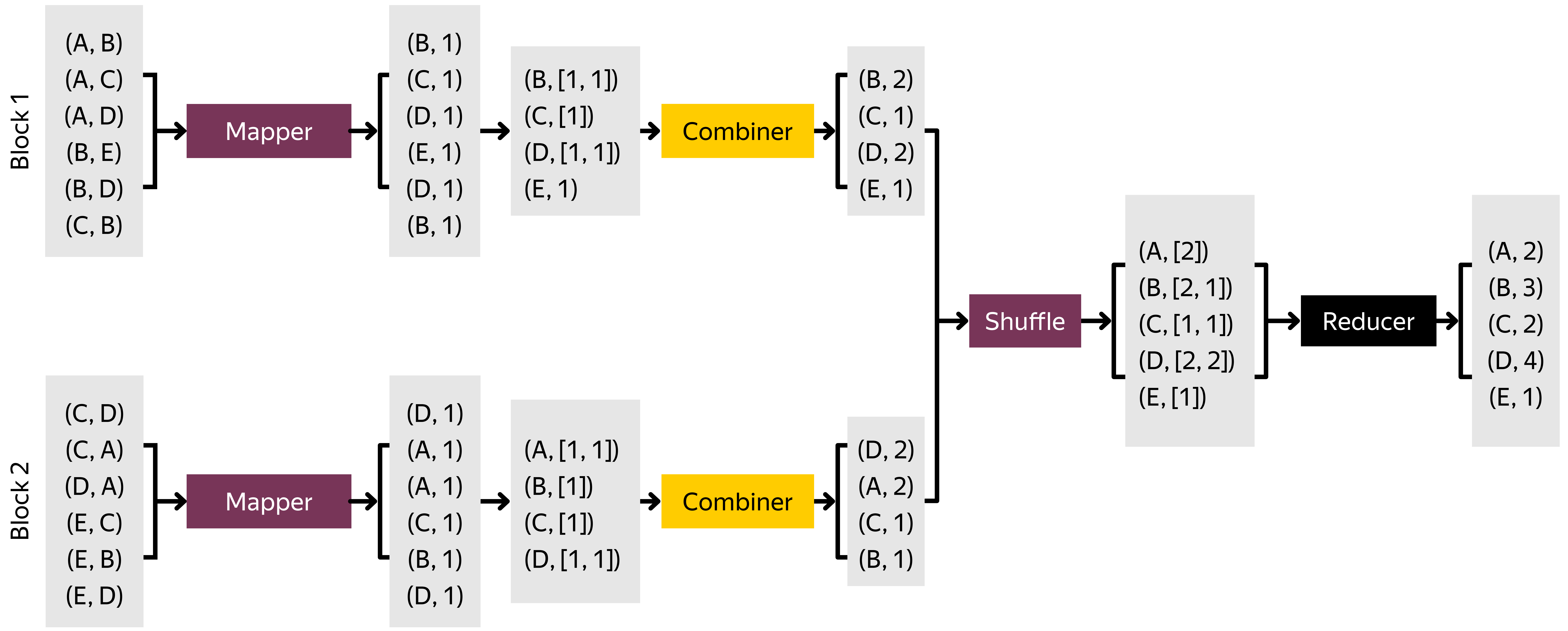
output: (a, 1), (b, 1), (c, 1), (d, 2), (e, 1), (word, 4), ...

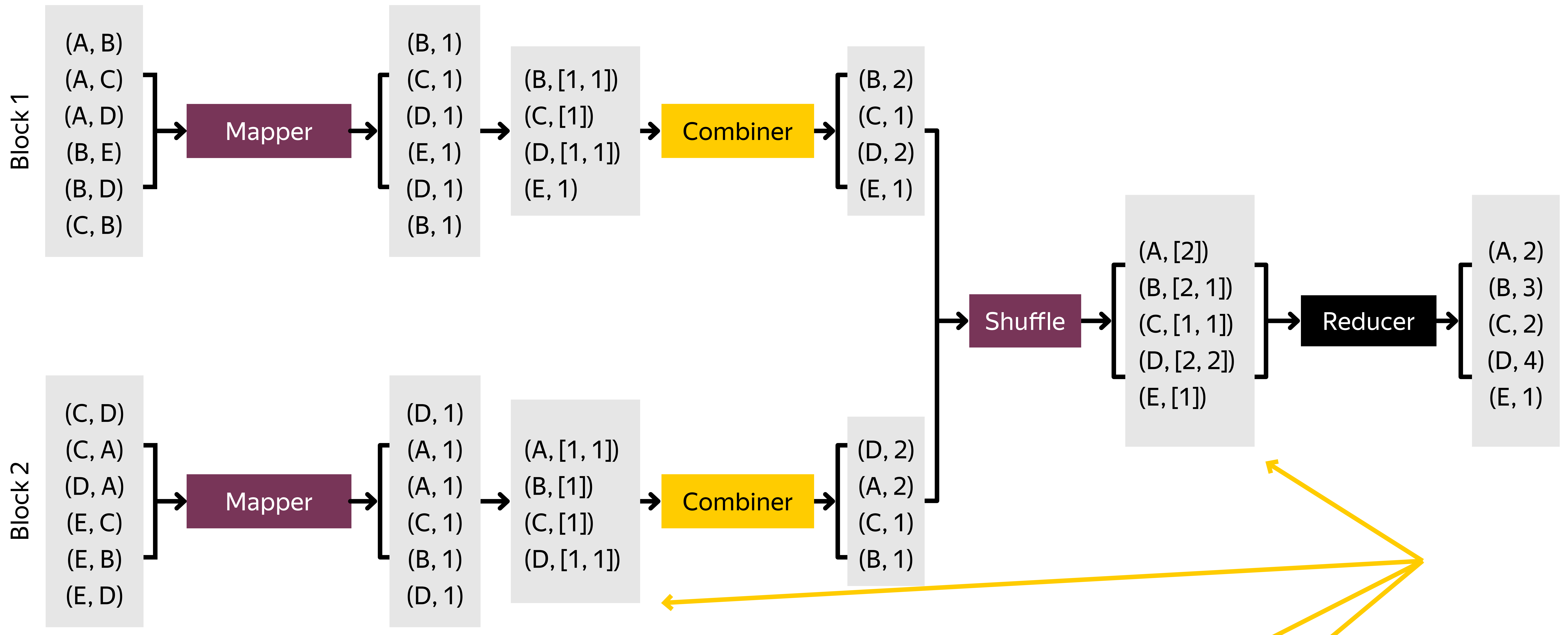
	without Combiner	with Combiner
Wall time (sec)	**935**	**528**
CPU time (sec)	**9790**	**6584**
Local FS Read (MB)	**3006**	**1324**
Local FS Write (MB)	**4527**	**1963**
Peek Map phys. memory (MB)	**526**	**606**
Peek Map virt. memory (MB)	**2131**	**2144**
Peek Reduce phys. memory (MB)	**2744**	**631**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


A α
B β
C γ
D δ
E ε
F ζ
MAPPER
MAPPER
MAPPER
MAPPER
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
COMBINER
COMBINER
COMBINER
COMBINER
a 1
b 2
c 9
a 5
c 2
b 7
c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
REDUCER
REDUCER
REDUCER
X 5
Y 7
Z 9

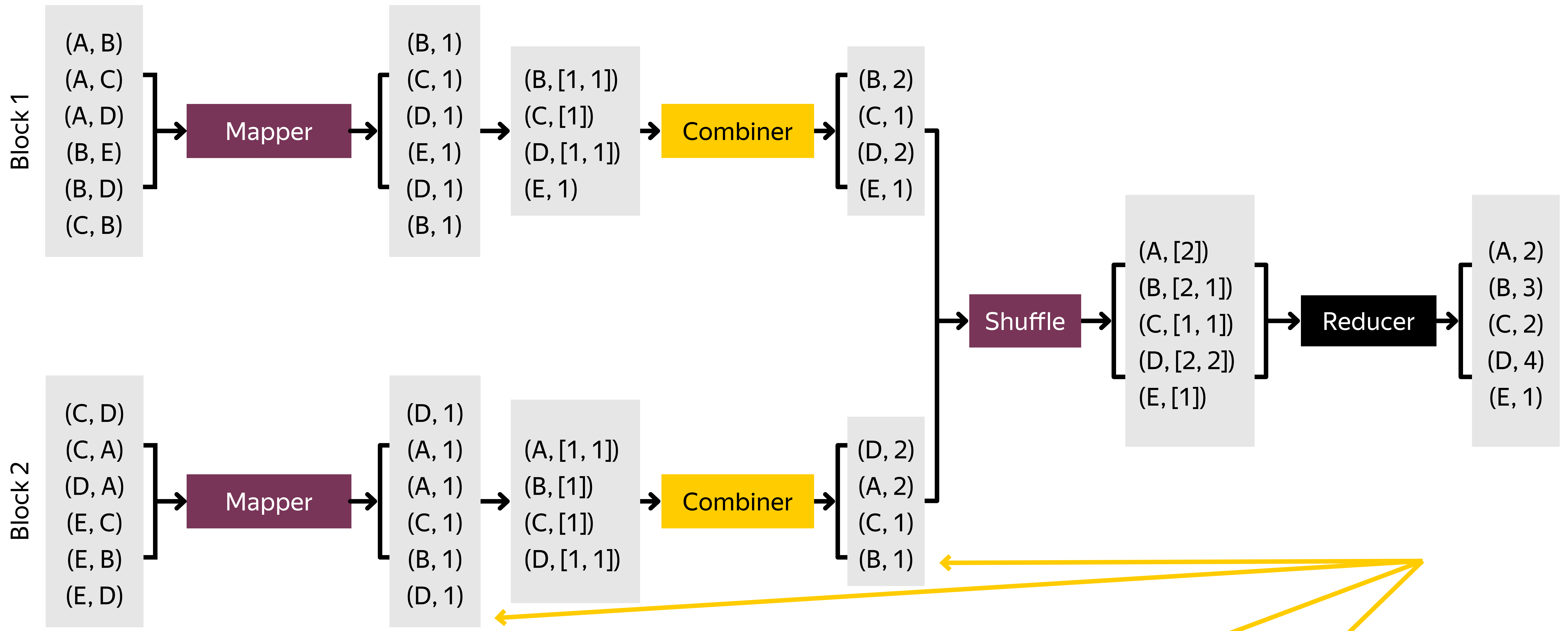


A α
B β
C γ
D δ
E ε
F ζ
MAPPER
MAPPER
MAPPER
MAPPER
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
COMBINER
COMBINER
COMBINER
COMBINER
a 1
b 2
c 9
a 5
c 2
b 7
c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
REDUCER
REDUCER
REDUCER
X 5
Y 7
Z 9


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)



- 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), ...]

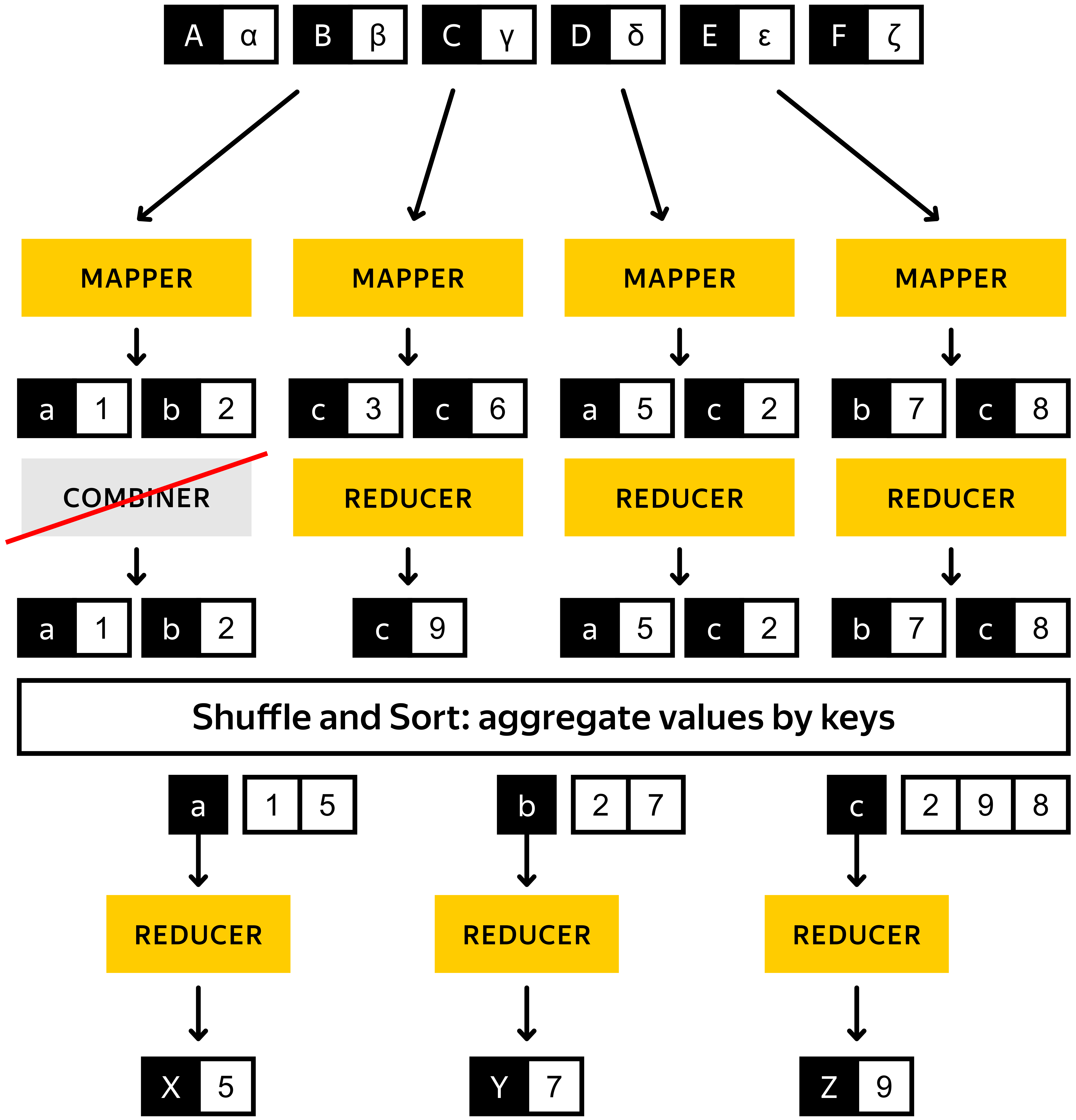


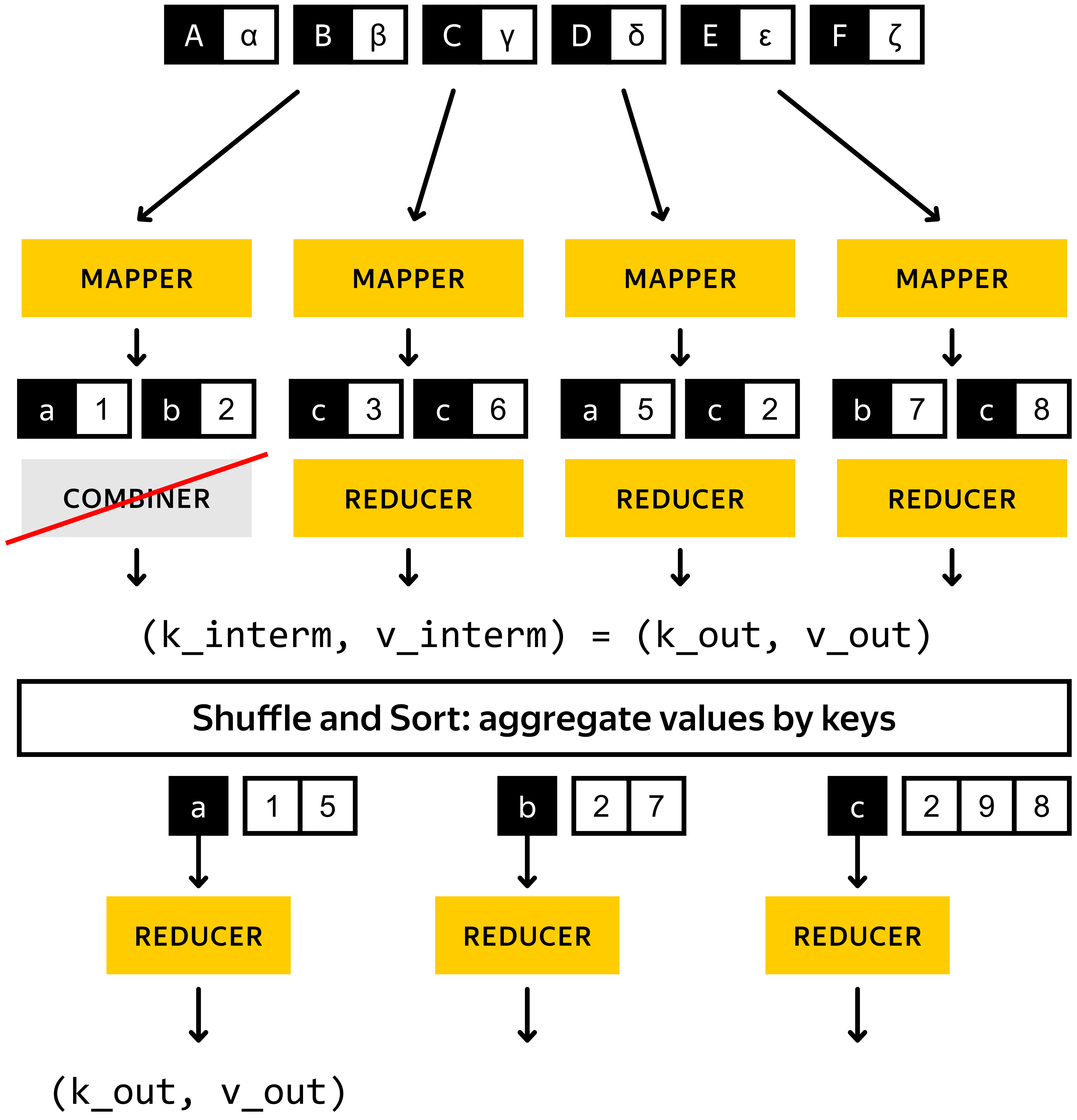
- read: [(k_in, v_in), ...]
- map: (k_in, v_in) —> **[(k_interm, v_interm), ...]**
- combiner: **(k_interm, [(v_interm, ...)])** —> **[(k_interm, v_interm), ...]**
- Shuffle & Sort: sort and group by k_interm
- reduce: **(k_interm, [(v_interm, ...)])** —-> [(k_out, v_out), ...]

```
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
```

```
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
```


A α
B β
C γ
D δ
E ε
F ζ
MAPPER
MAPPER
MAPPER
MAPPER
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
~~COMBINER~~
REDUCER
REDUCER
REDUCER
a 1
b 2
c 9
a 5
c 2
b 7
c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
REDUCER
REDUCER
REDUCER
X 5
Y 7
Z 9


A α
B β
C γ
D δ
E ε
F ζ
MAPPER
MAPPER
MAPPER
MAPPER
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
~~COMBINER~~
REDUCER
REDUCER
REDUCER
(k_interm, v_interm) = (k_out, v_out)
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
REDUCER
REDUCER
REDUCER
(k_out, v_out)

WIKIPEDIA
The Free Encyclopedia

$$\underbrace{\texttt{<article id>}}_{\texttt{key}}\ \texttt{<tab>}\ \underbrace{\texttt{<article content>}}_{\texttt{value}}$$

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

$$\Big\downarrow \text{ mean}$$

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

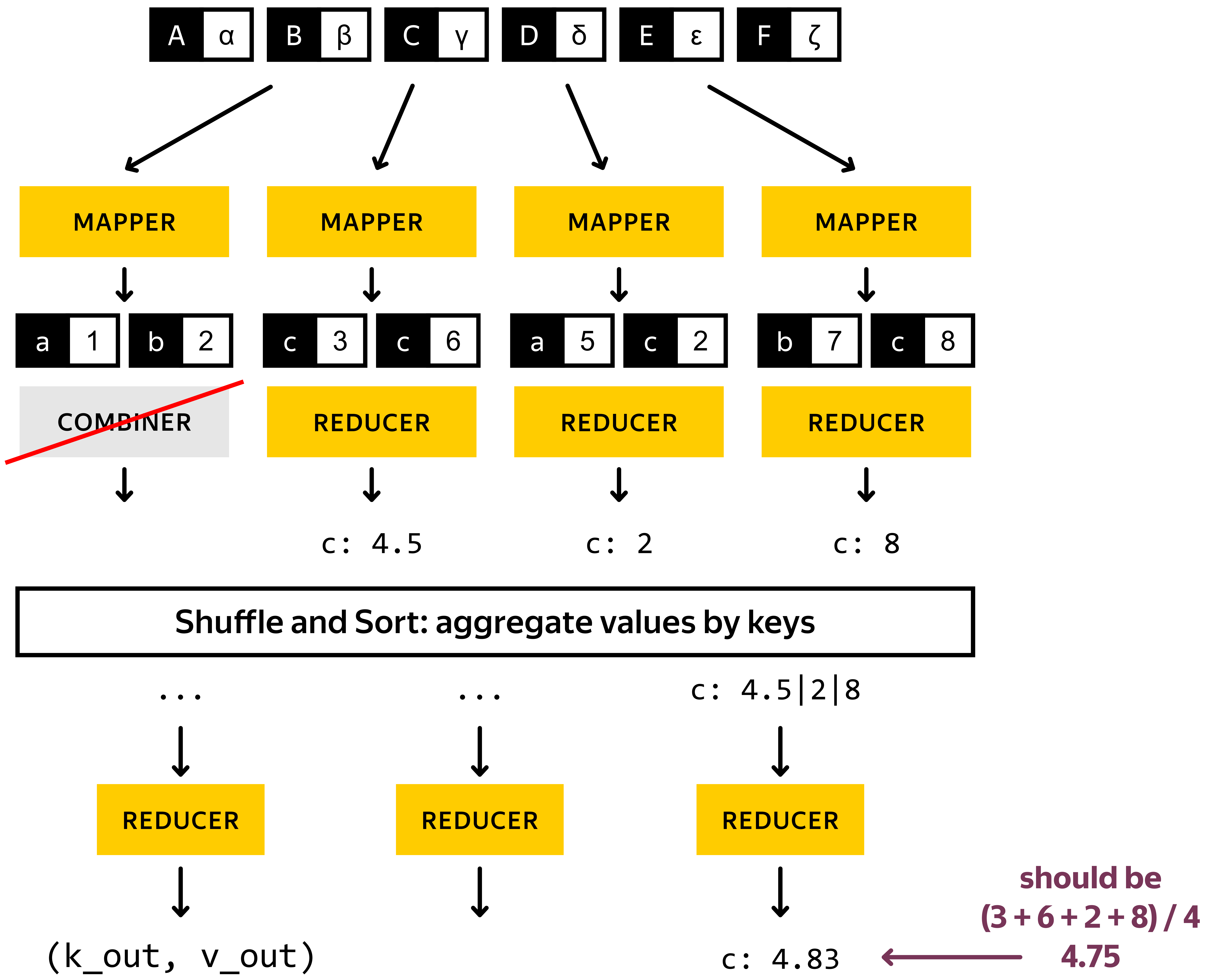
```python
from __future__ import print_function
import sys

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")
```

MEAN


A α
B β
C γ
D δ
E ε
F ζ
MAPPER
MAPPER
MAPPER
MAPPER
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
~~COMBINER~~
REDUCER
REDUCER
REDUCER
c: 4.5
c: 2
c: 8
Shuffle and Sort: aggregate values by keys
...
...
c: 4.5|2|8
REDUCER
REDUCER
REDUCER
(k_out, v_out)
c: 4.83
should be
(3 + 6 + 2 + 8) / 4
4.75

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")
```

Mapper (Python): combiner.py

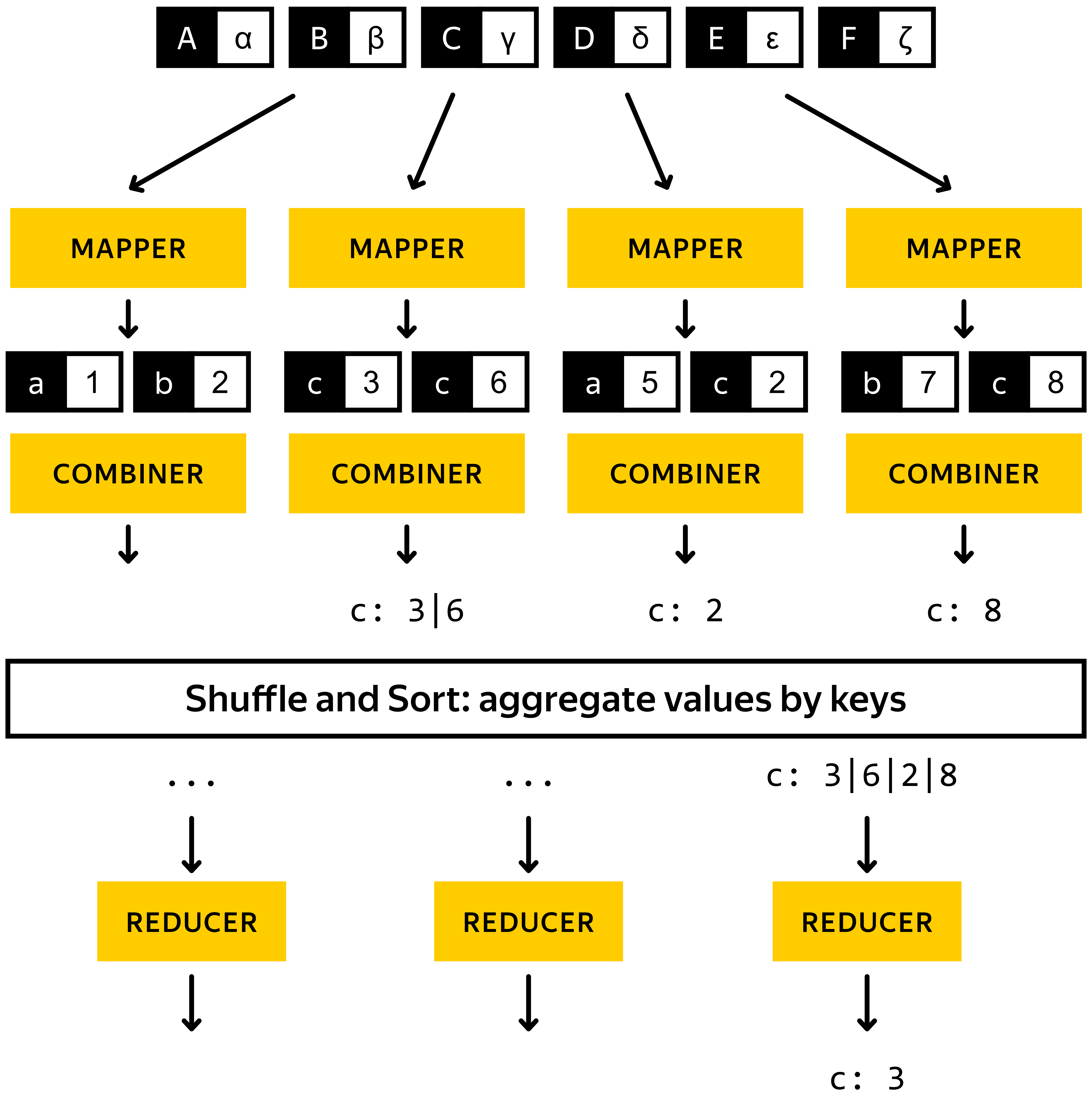
```python
from __future__ import print_function
import sys

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")
```

MEDIAN



A α
B β
C γ
D δ
E ε
F ζ
MAPPER
MAPPER
MAPPER
MAPPER
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
COMBINER
COMBINER
COMBINER
COMBINER
c: 3|6
c: 2
c: 8
Shuffle and Sort: aggregate values by keys
...
...
c: 3|6|2|8
REDUCER
REDUCER
REDUCER
c: 3

Summary

Summary

- Based on Mapper and Reducer you **can derive** MapReducer Combiner signature

Summary

- Based on Mapper and Reducer you **can derive** MapReducer Combiner signature
- You know **how to call** MapReduce Combiner in streaming applications

Summary

- Based on Mapper and Reducer you **can derive** MapReducer Combiner signature
- You know **how to call** MapReduce Combiner in streaming applications
- You know **how to write** MapReduce Combiner streaming scripts

Summary

- Based on Mapper and Reducer you **can derive** MapReducer Combiner signature
- You know **how to call** MapReduce Combiner in streaming applications
- You know **how to write** MapReduce Combiner streaming scripts
- You **can identify situations** where you do not need to use Combiner

BigDATAteam