Yandex

MapReduce

Partitioner

World of MapReduce

Combiner Partitioner Comparator

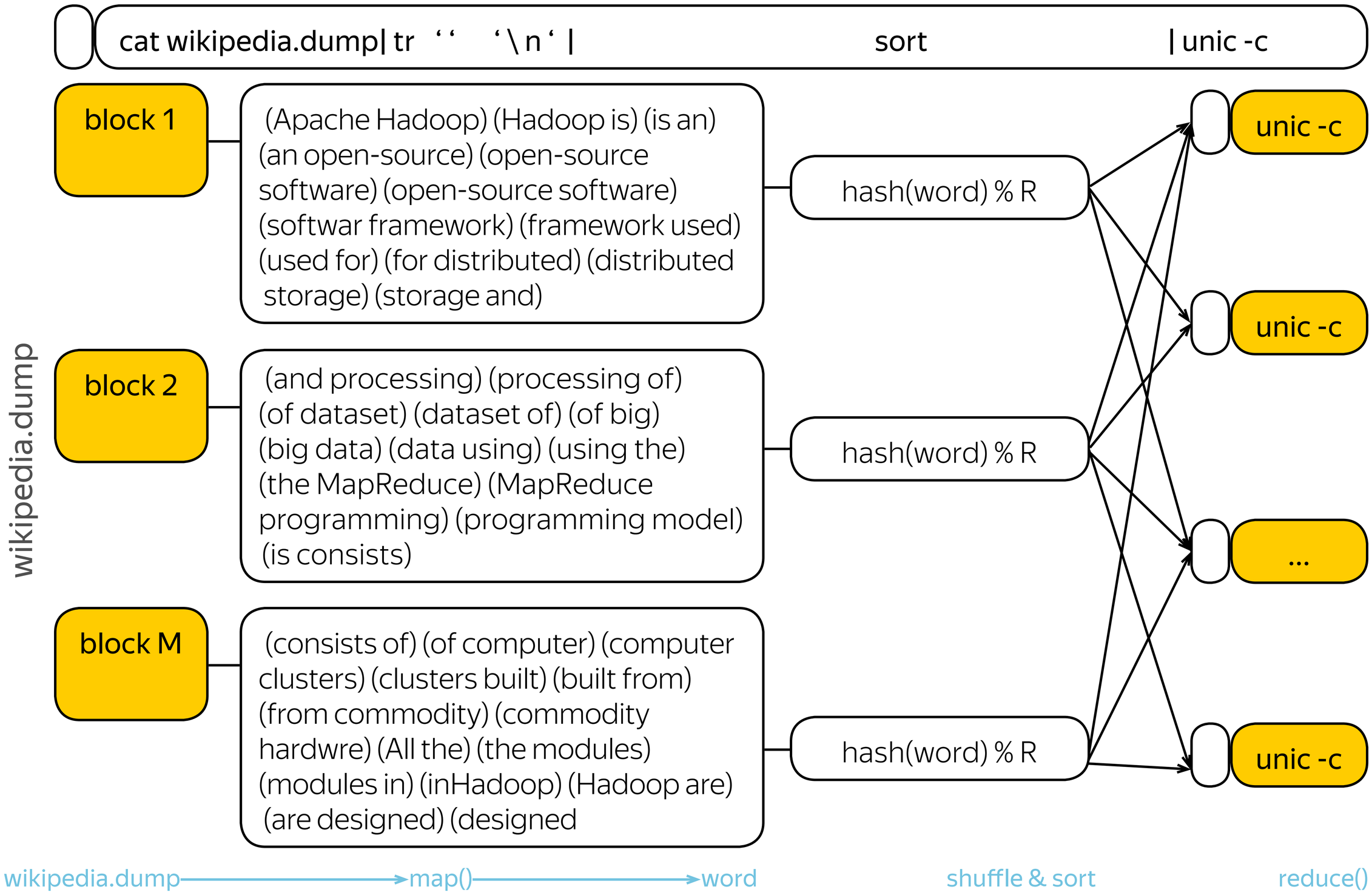
World of MapReduce

Combiner Partitioner Comparator

World of MapReduce

Combiner Partitioner Comparator

WordCount


cat wikipedia.dump| tr ‘ ‘ ‘ \ n ‘ | sort | unic -c
wikipedia.dump
block 1
(Apache Hadoop) (Hadoop is) (is an) (an open-source) (open-source software) (open-source software) (softwar framework) (framework used) (used for) (for distributed) (distributed storage) (storage and)
block 2
(and processing) (processing of) (of dataset) (dataset of) (of big) (big data) (data using) (using the) (the MapReduce) (MapReduce programming) (programming model) (is consists)
block M
(consists of) (of computer) (computer clusters) (clusters built) (built from) (from commodity) (commodity hardwre) (All the) (the modules) (modules in) (inHadoop) (Hadoop are) (are designed) (designed
hash(word) % R
hash(word) % R
hash(word) % R
unic -c
unic -c
...
unic -c
wikipedia.dump
map()
word
shuffle & sort
reduce()

Collocations

natural English	unnatural English
the fast train	the ~~quick~~ train
fast food	~~quick~~ food
a quick shower	a ~~fast~~ shower
a quick meal	a ~~fast~~ meal

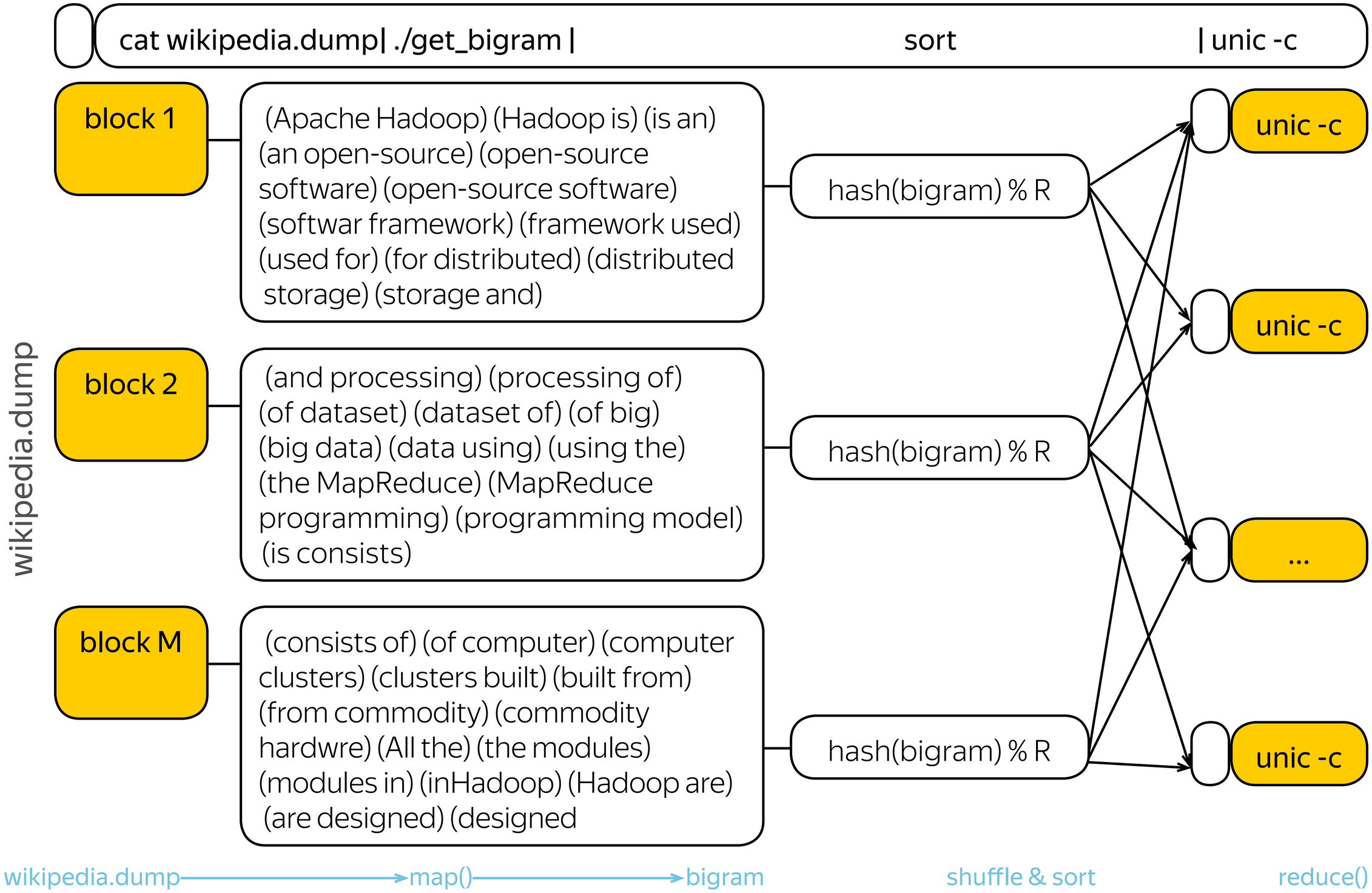
Collocations

natural English	unnatural English
the fast train	the ~~quick~~ train
fast food	~~quick~~ food
a quick shower	a ~~fast~~ shower
a quick meal	a ~~fast~~ meal

New and York

United and States

WordCount


cat wikipedia.dump| ./get_bigram |
sort
| unic -c
wikipedia.dump
block 1
(Apache Hadoop) (Hadoop is) (is an) (an open-source) (open-source software) (open-source software) (softwar framework) (framework used) (used for) (for distributed) (distributed storage) (storage and)
block 2
(and processing) (processing of) (of dataset) (dataset of) (of big) (big data) (data using) (using the) (the MapReduce) (MapReduce programming) (programming model) (is consists)
block M
(consists of) (of computer) (computer clusters) (clusters built) (built from) (from commodity) (commodity hardwre) (All the) (the modules) (modules in) (inHadoop) (Hadoop are) (are designed) (designed
hash(bigram) % R
hash(bigram) % R
hash(bigram) % R
unic -c
unic -c
...
unic -c
wikipedia.dump
map()
bigram
shuffle & sort
reduce()

input

```
$ head -c 100 wikipedia_sample.txt
12 Anarchism Anarchism is often defined as a political
philosophy which holds the state to ...
```

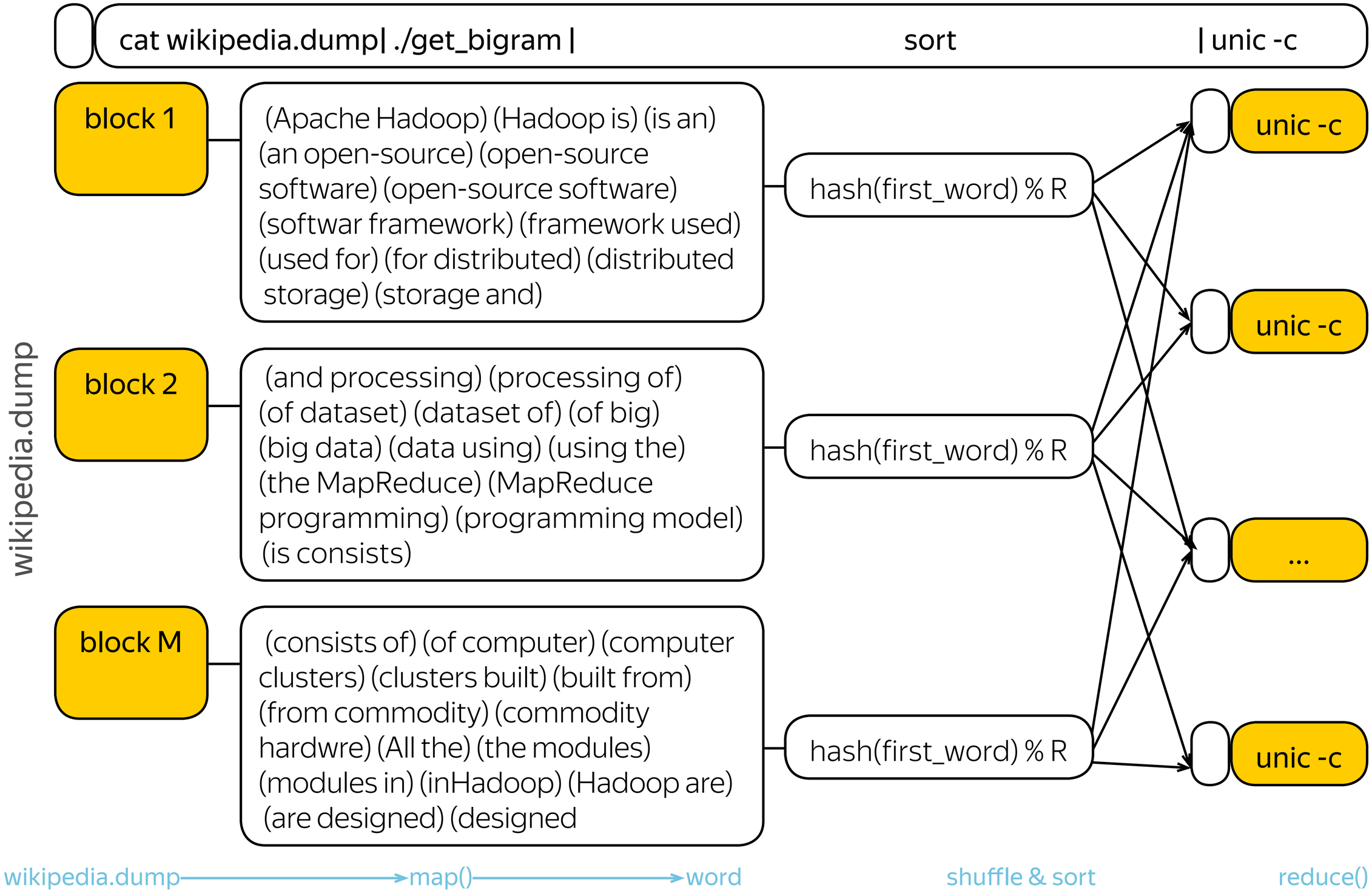
Mapper (Python): bigram_mapper.py

```
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 index in range(len(words) - 1):
    print(words[index], words[index + 1], 1, sep="\t")
```

output

```
$ head word_count/part-00000
Anarchism Anarchism 1
Anarchism is 1
is often 1
```

WordCount


cat wikipedia.dump| ./get_bigram |
sort
| unic -c
wikipedia.dump
block 1
(Apache Hadoop) (Hadoop is) (is an) (an open-source) (open-source software) (open-source software) (softwar framework) (framework used) (used for) (for distributed) (distributed storage) (storage and)
block 2
(and processing) (processing of) (of dataset) (dataset of) (of big) (big data) (data using) (using the) (the MapReduce) (MapReduce programming) (programming model) (is consists)
block M
(consists of) (of computer) (computer clusters) (clusters built) (built from) (from commodity) (commodity hardwre) (All the) (the modules) (modules in) (inHadoop) (Hadoop are) (are designed) (designed
hash(first_word) % R
hash(first_word) % R
hash(first_word) % R
unic -c
unic -c
...
unic -c
wikipedia.dump
map()
word
shuffle & sort
reduce()

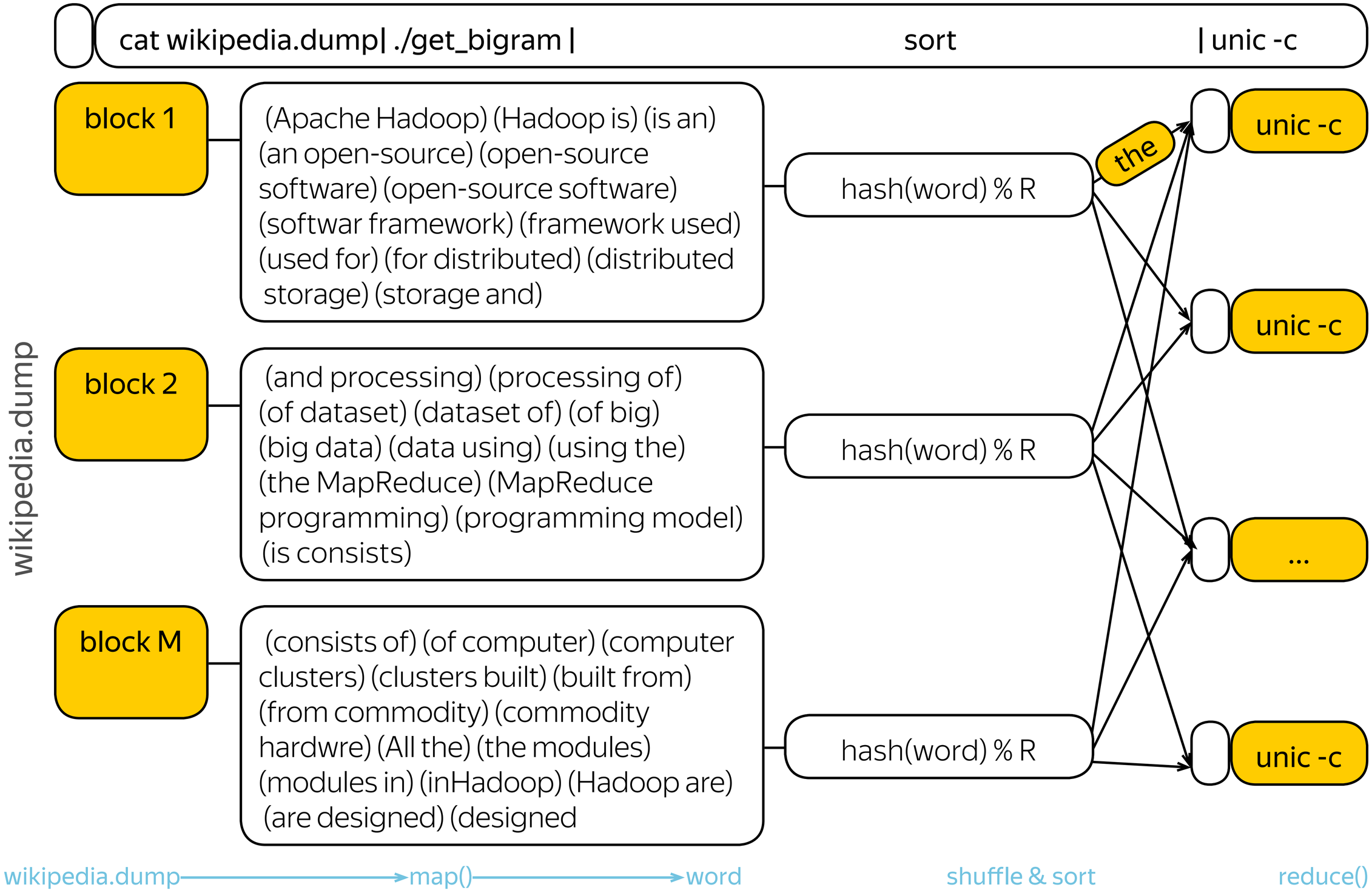
Mapper (Python): inmemroy_bigram_reducer.py

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

current_word = None
bigram_count = Counter()

for line in sys.stdin:
  first_word, second_word, counts = line.split("\t", 2)
  counts = Counter({second_word: int(counts)})
  if first_word == current_word:
    bigram_count += counts
  else:
    if current_word:
      for second_word, bigram_count in bigram_count.items():
        print(current_word, second_word,
```

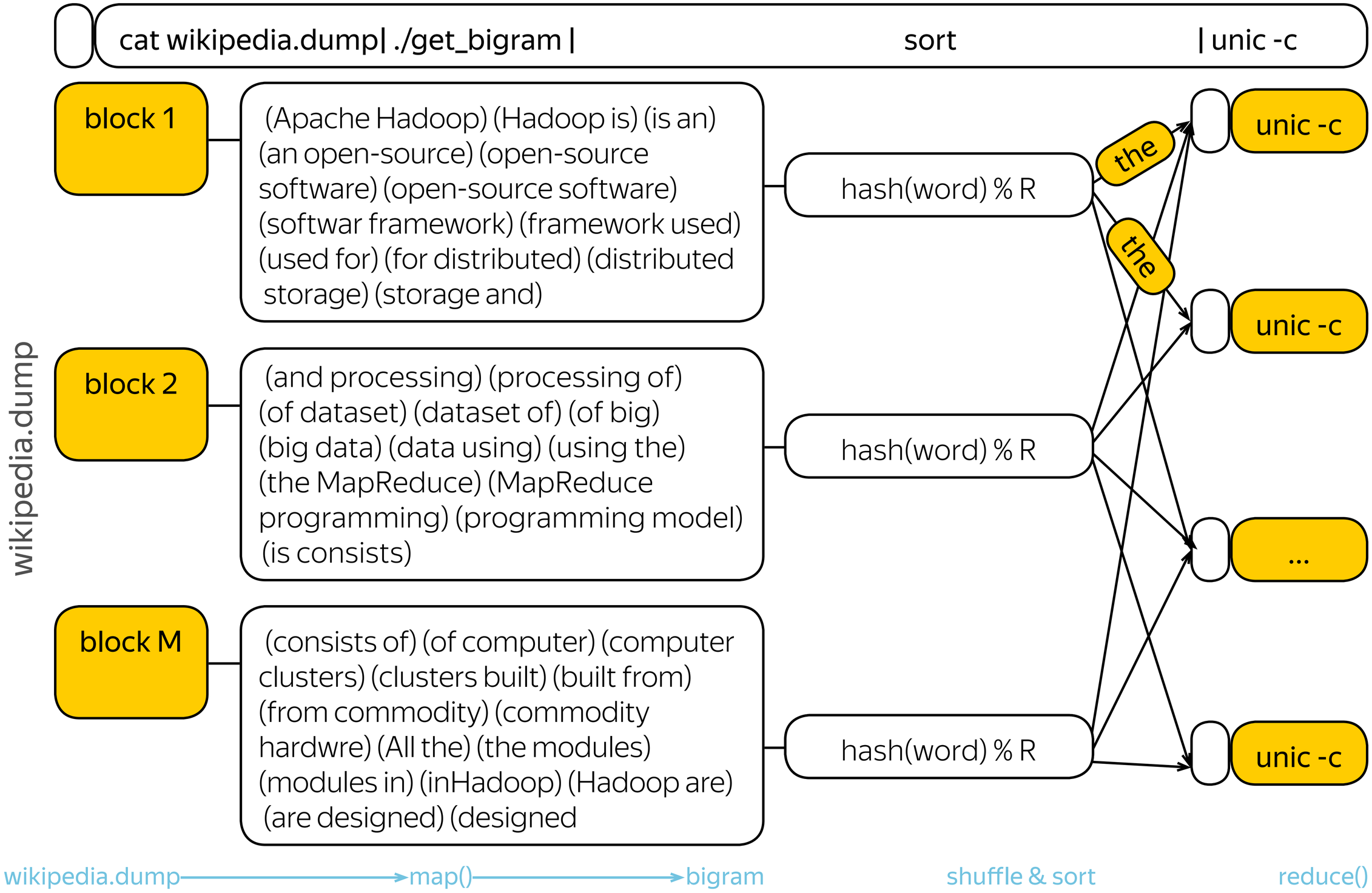
```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -files bigram_mapper.py,inmemory_bigram_reducer.py \
        -mapper 'python bigram_mapper.py' \
        -reducer 'python inmemory_bigram_reducer.py' \
        -numReduceTasks 5 \
        -input wikipedia_sample.txt \
        -output word_count \
```

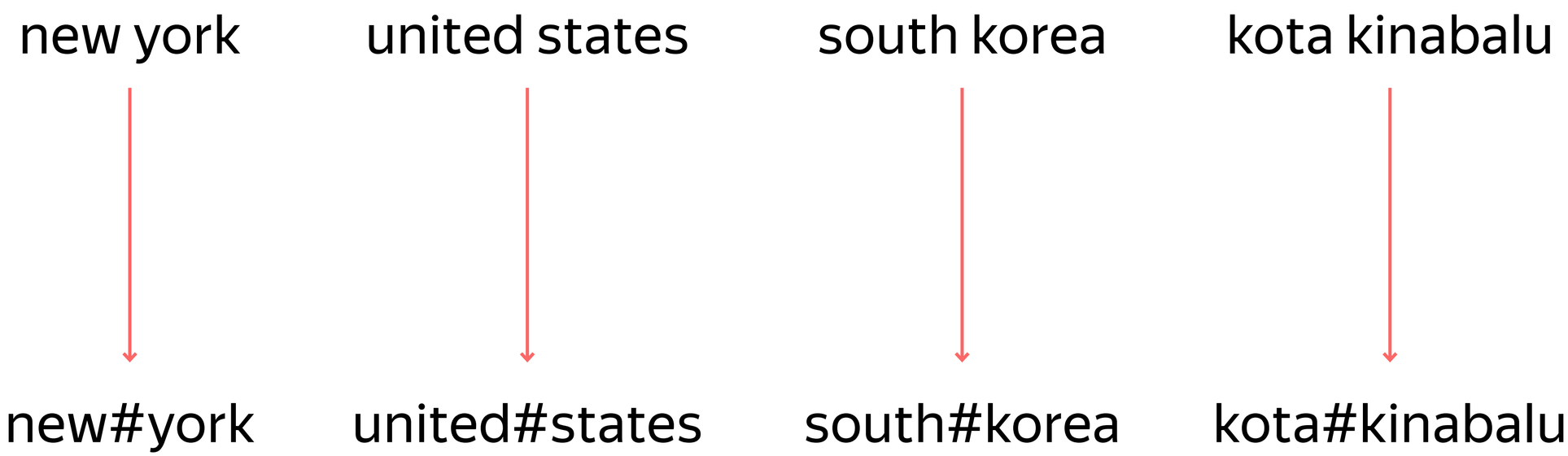
```
$ grep $'^new\t' word_count/* | sort -nrk3,3 | head -4
word_count/part-00001:new     york    112
word_count/part-00001:new     world   15
word_count/part-00001:new     jewrsey   12
word_count/part-00001:new     constitution    12
$ grep $'\tyork\t' word_count/* | sort -nrk3,3 | head
word_count/part-00001:new     york    112
word_count/part-00004:sergeant york    1
```

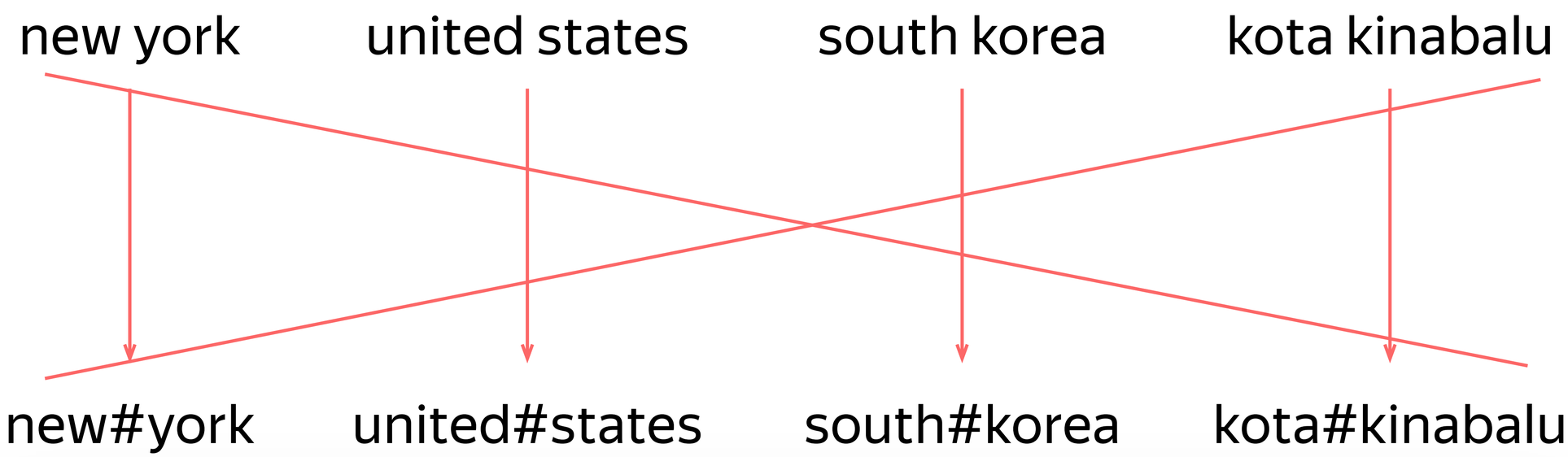
WordCount


cat wikipedia.dump| ./get_bigram |
sort
| unic -c
wikipedia.dump
block 1
(Apache Hadoop) (Hadoop is) (is an) (an open-source) (open-source software) (open-source software) (softwar framework) (framework used) (used for) (for distributed) (distributed storage) (storage and)
block 2
(and processing) (processing of) (of dataset) (dataset of) (of big) (big data) (data using) (using the) (the MapReduce) (MapReduce programming) (programming model) (is consists)
block M
(consists of) (of computer) (computer clusters) (clusters built) (built from) (from commodity) (commodity hardwre) (All the) (the modules) (modules in) (inHadoop) (Hadoop are) (are designed) (designed
hash(word) % R
hash(word) % R
hash(word) % R
the
unic -c
unic -c
...
unic -c
wikipedia.dump
map()
word
shuffle & sort
reduce()

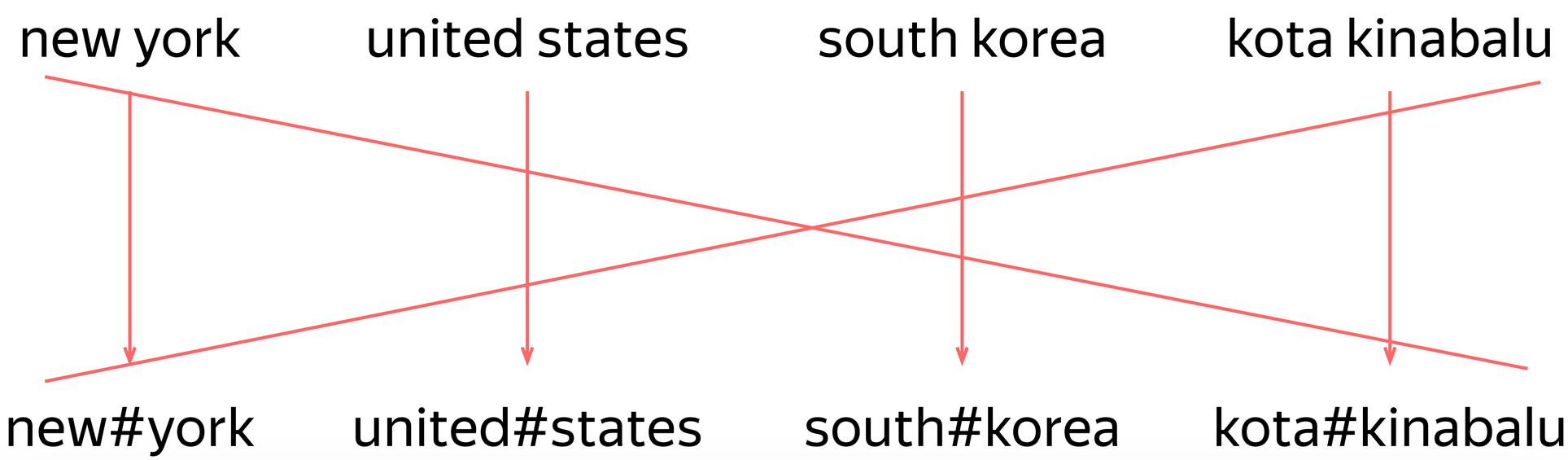
WordCount


cat wikipedia.dump| ./get_bigram |
sort
| unic -c
wikipedia.dump
block 1
(Apache Hadoop) (Hadoop is) (is an) (an open-source) (open-source software) (open-source software) (softwar framework) (framework used) (used for) (for distributed) (distributed storage) (storage and)
block 2
(and processing) (processing of) (of dataset) (dataset of) (of big) (big data) (data using) (using the) (the MapReduce) (MapReduce programming) (programming model) (is consists)
block M
(consists of) (of computer) (computer clusters) (clusters built) (built from) (from commodity) (commodity hardwre) (All the) (the modules) (modules in) (inHadoop) (Hadoop are) (are designed) (designed
hash(word) % R
hash(word) % R
hash(word) % R
the
the
unic -c
unic -c
...
unic -c
wikipedia.dump
map()
bigram
shuffle & sort
reduce()


new york
united states
south korea
kota kinabalu
new#york
united#states
south#korea
kota#kinabalu

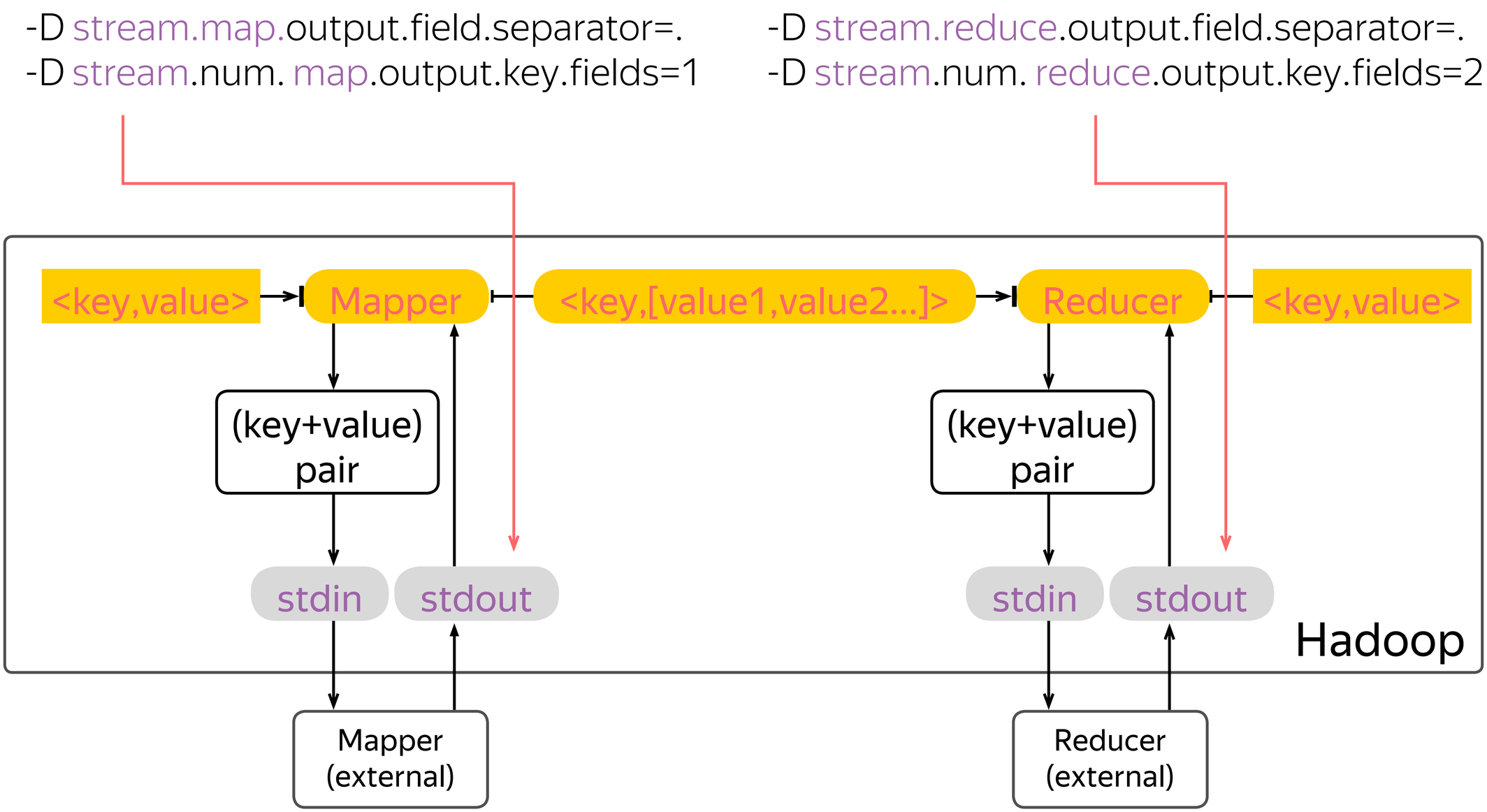


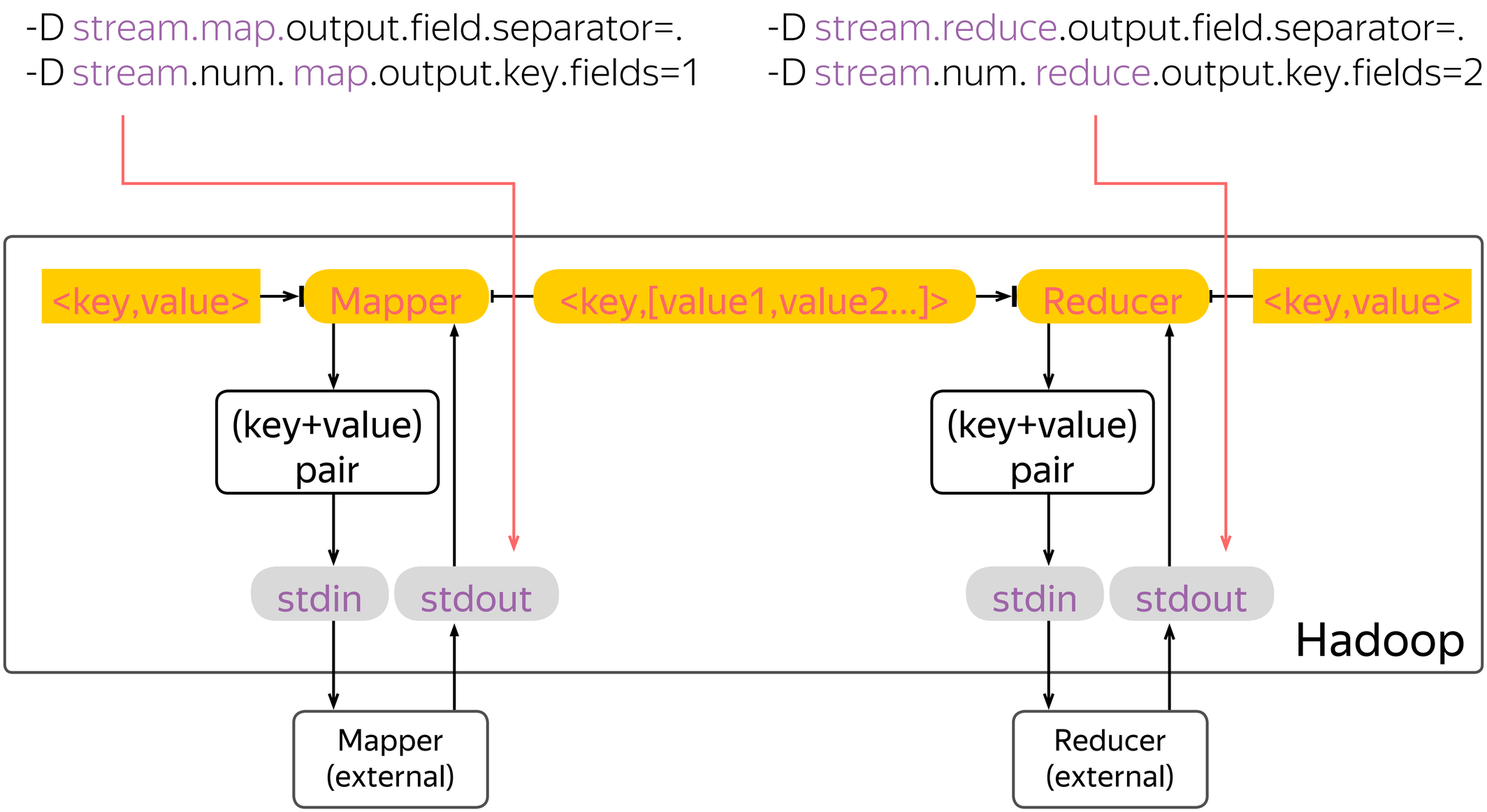

```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D stream.num.map.output.key.fields=2 \
        -files bigram_mapper.py,bigram_reducer.py \
        -mapper 'python bigram_mapper.py' \
        -reducer 'python bigram_reducer.py' \
        -numReduceTasks 5 \
        -input wikipedia_sample.txt \
        -output word_count \
```




```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
    -D stream.num.map.output.key.fields=2 \
    -files bigram_mapper.py,bigram_reducer.py \
    -mapper 'python bigram_mapper.py' \
    -reducer 'python bigram_reducer.py' \
    -numReduceTasks 5 \
    -input wikipedia_sample.txt \
    -output word_count \
```

```
$ grep $'^new\t' word_count/* | sort -nrk3,3 | head -4
word_count/part-00001:new york 112
word_count/part-00001:new world 15
word_count/part-00002:new constitution 12
word_count/part-00000:new jersey 12
```


-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
<key,value>
Mapper
<key,[value1,value2...]>
Reducer
<key,value>
(key+value) pair
(key+value) pair
stdin
stdout
stdin
stdout
Hadoop
Mapper (external)
Reducer (external)



IPv4 address (example): 69.89.31.226

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
```

input

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

```
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
```

input

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

```
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
```

input

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

```
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
```

input

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

man sort | grep KEYDEF

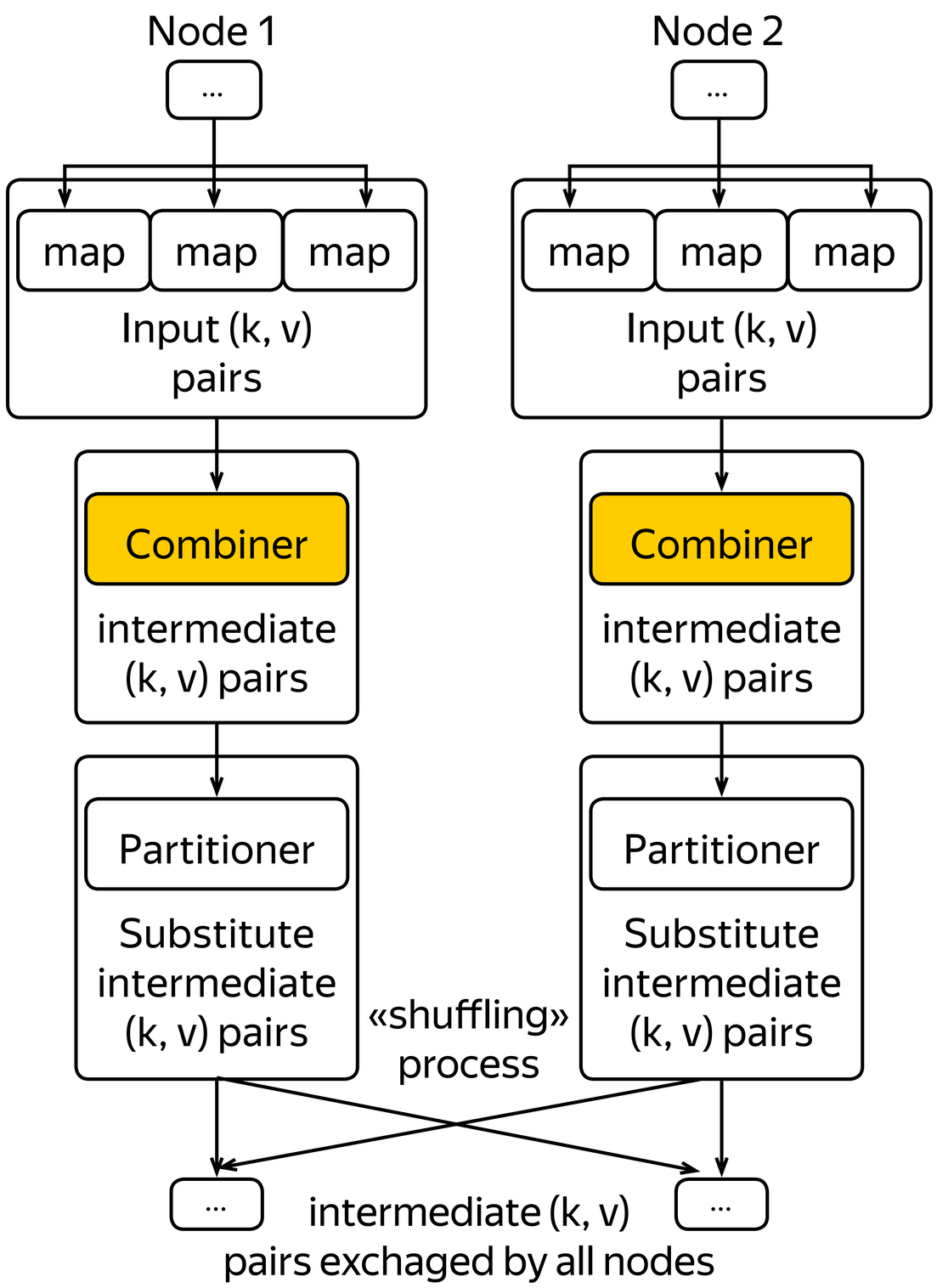
```
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR \
        -D map.output.key.field.separator=. \
        -D mapreduce.partition.keypartitioner.options=-k1.2,1.2 \
        -files identity_mr.py \
        -mapper 'python identity_mr.py' \
        -reducer 'python identity_mr.py' \
    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

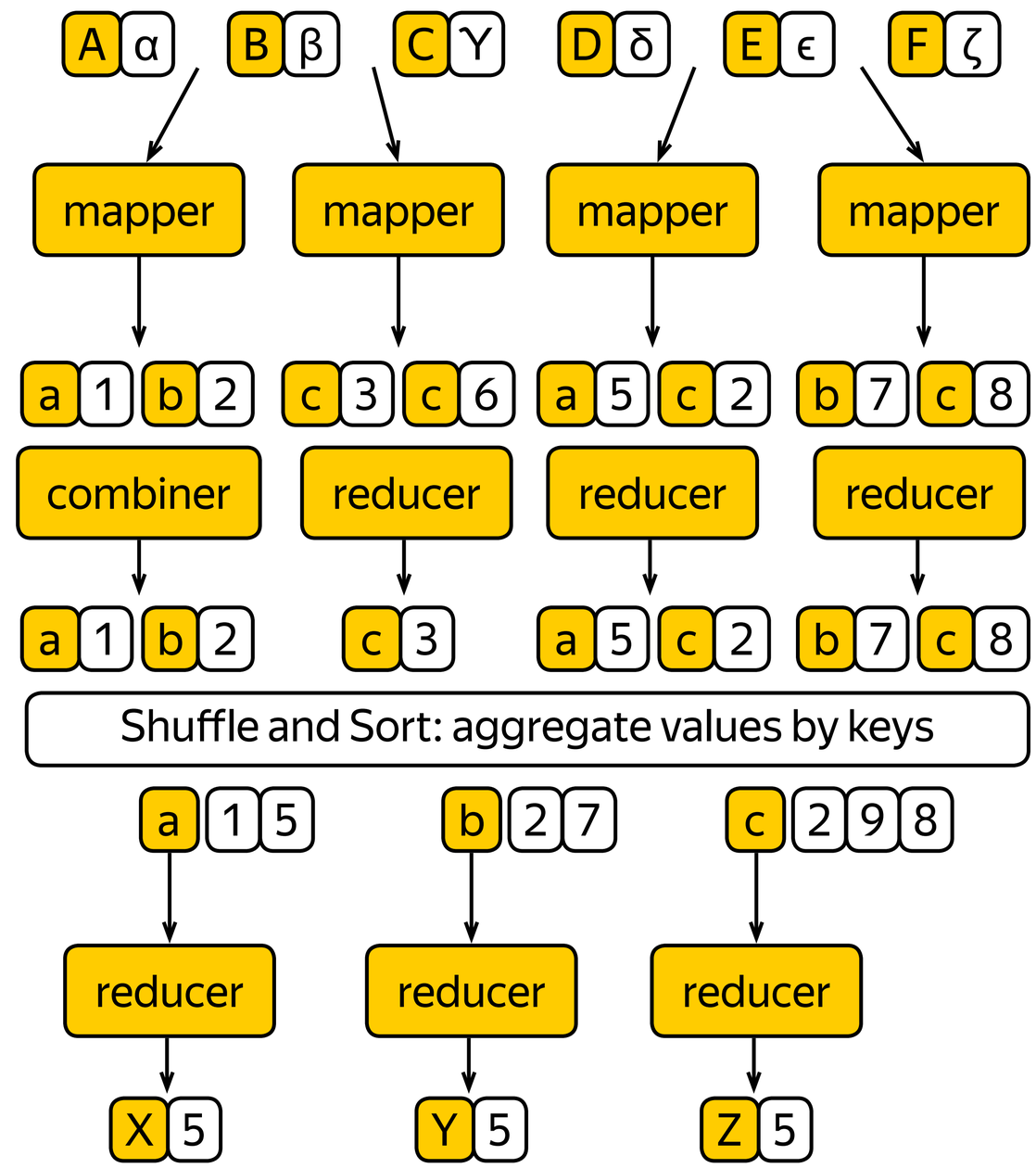
output

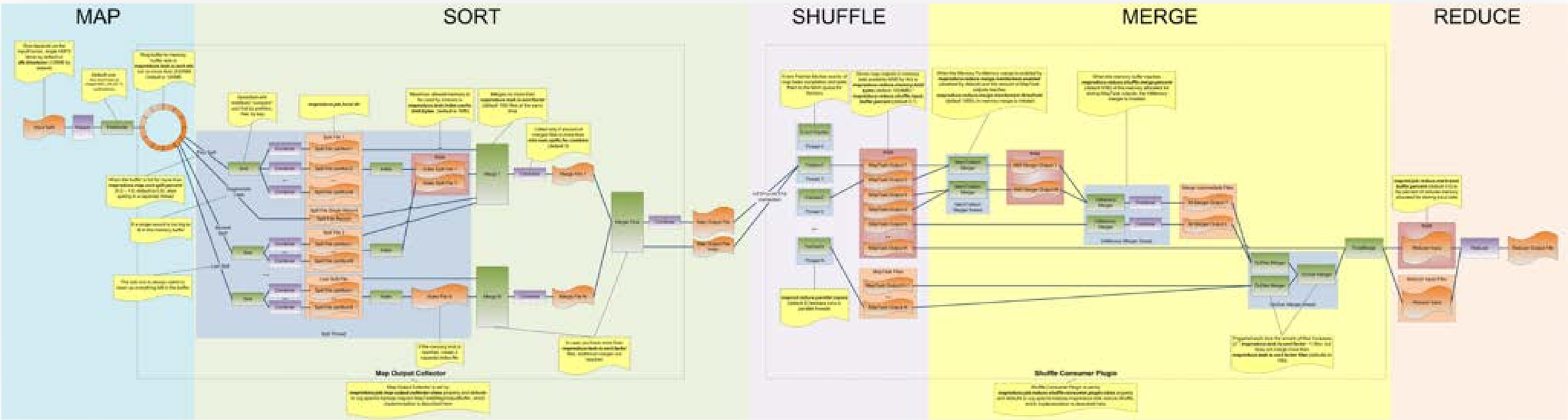
```
cat subnet_out/*
3b.4.5.6
4b.5.6.7
1a.2.3.4
2a.3.4.5
```

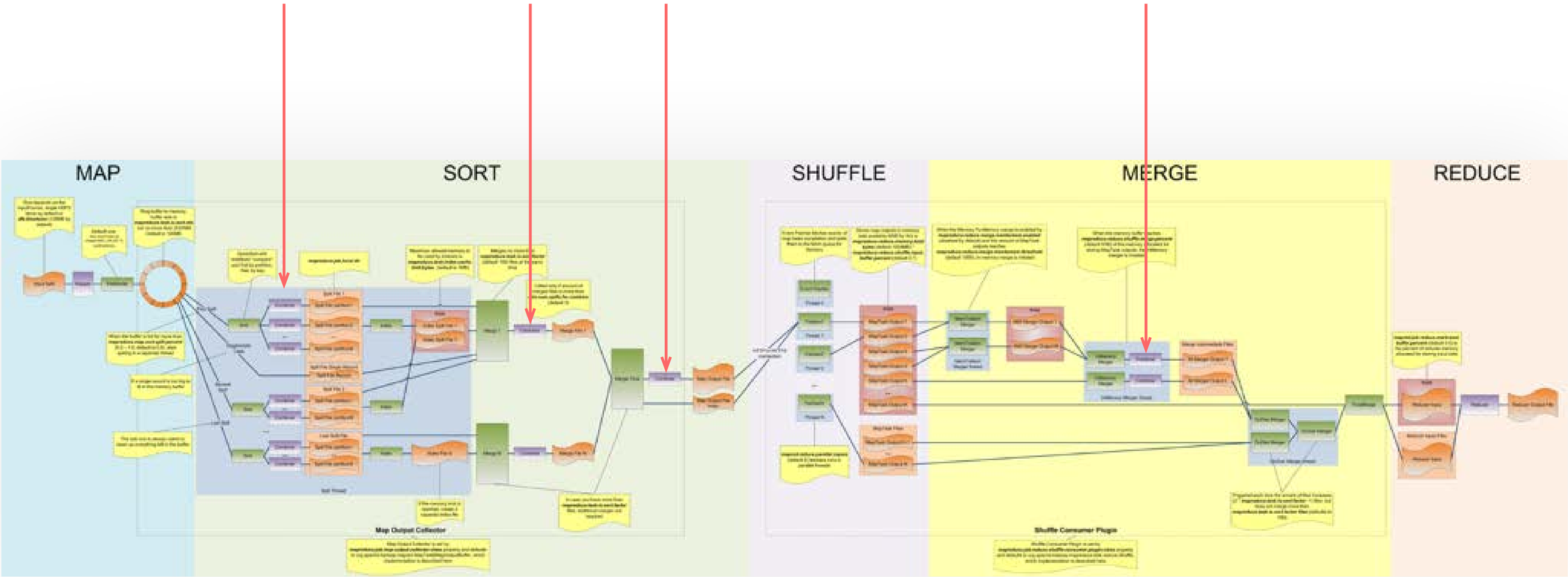
```
ls -lth subnet_out
total 8.0K
0 Apr 1 14:59 _SUCCESS
20 Apr 1 14:59 part-00001
20 Apr 1 14:59 part-00000
```

```
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
```


Node 1
...
map
map
map
Input (k, v) pairs
Combiner
intermediate (k, v) pairs
Partitioner
Substitute intermediate (k, v) pairs
Node 2
...
map
map
map
Input (k, v) pairs
Combiner
intermediate (k, v) pairs
Partitioner
Substitute intermediate (k, v) pairs
«shuffling» process
...
intermediate (k, v) pairs exchaged by all nodes
...


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 3
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 5
Z 5


MAP
SORT
SHUFFLE
MERGE
REDUCE
Map Output Collector
Shuffle Consumer Plugin


MAP
SORT
SHUFFLE
MERGE
REDUCE
Map Output Collector
Shuffle Consumer Plugin

Summary

Summary

- You know how to use Partitioner in MapReduce application

Summary

- You know how to use Partitioner in MapReduce application
- You know how to count bigrams with Hadoop MapReduce and distribute load over reducers

BigDATAteam