Yandex

MapReduce

Distributed Cache

<article id> <tab> <article content>

key value


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

WIKIPEDIA
The Free Encyclopedia

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

↓

hadoop

↓


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

WIKIPEDIA
The Free Encyclopedia

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


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

```python
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")
```

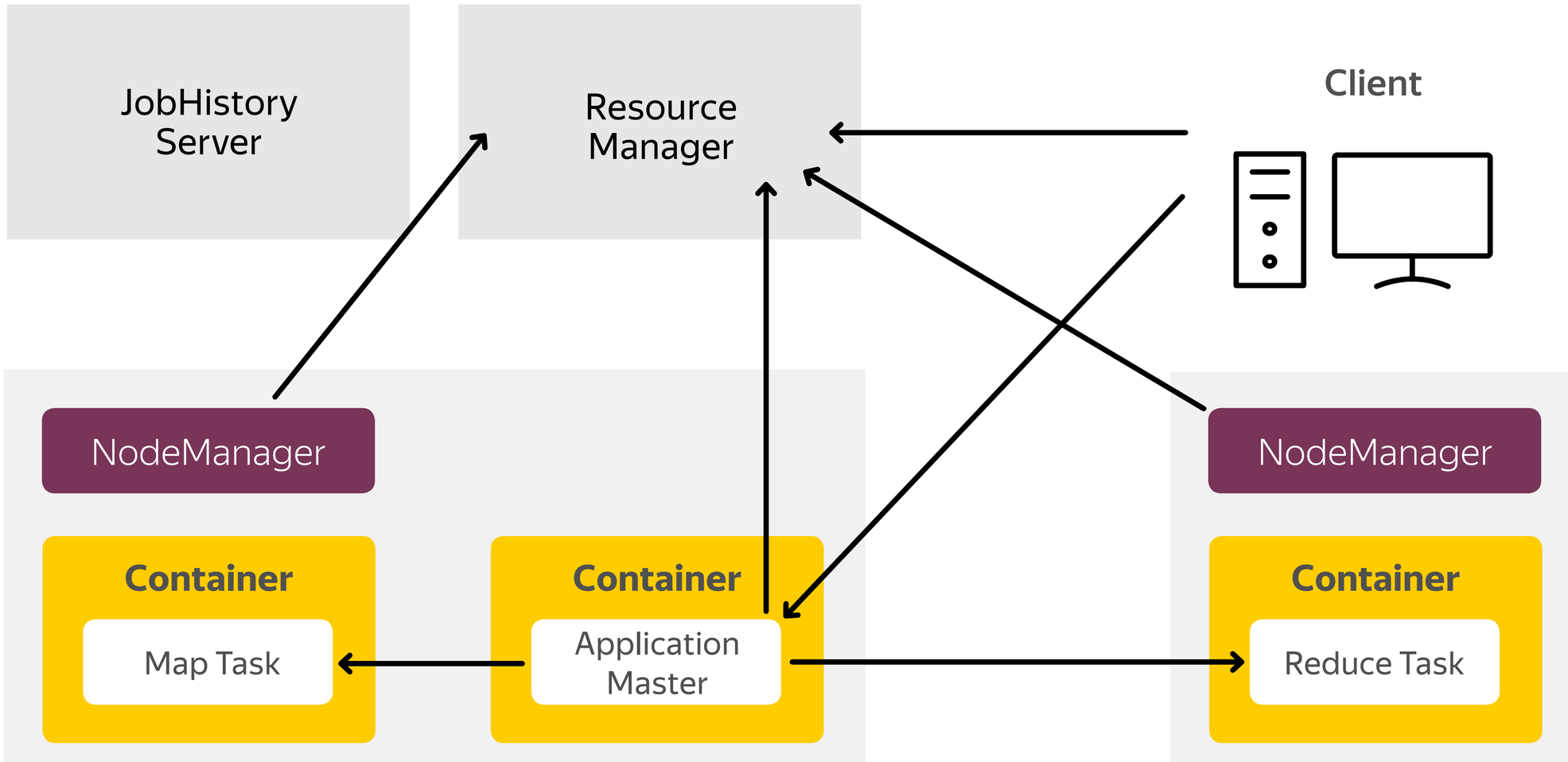
```
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py,reducer.py,vocabulary.txt \
          -mapper 'python mapper.py' \
          -reducer 'python reducer.py' \
          -input /data/wiki/en_articles \
          -output word_count
```

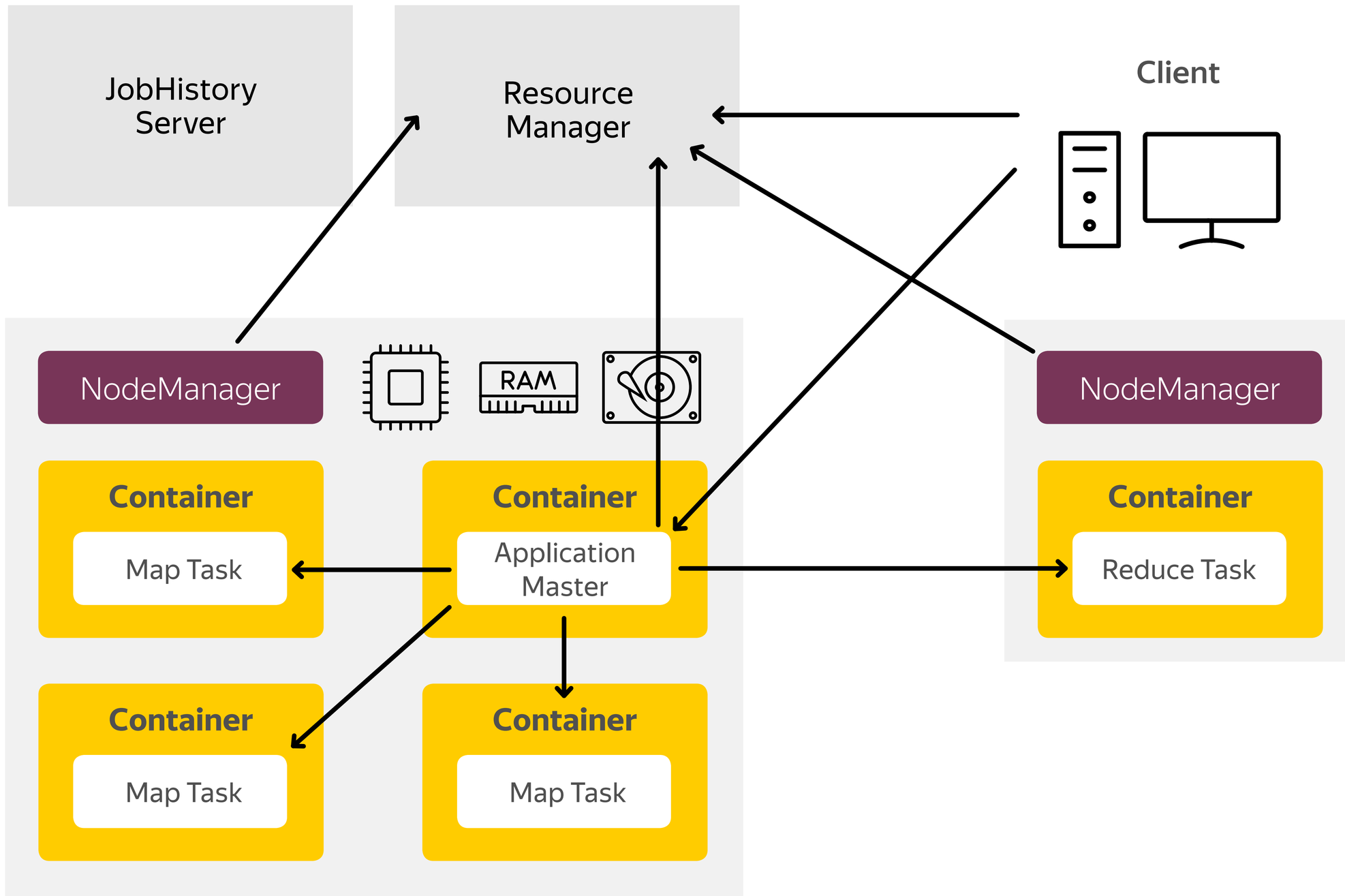
```
yarn jar $HADOOP_STREAMING_JAR \
          -files mapper.py,reducer.py,vocabulary.txt \
          -mapper 'python mapper.py' \
          -reducer 'python reducer.py' \
          -input /data/wiki/en_articles \
          -output word_count
```

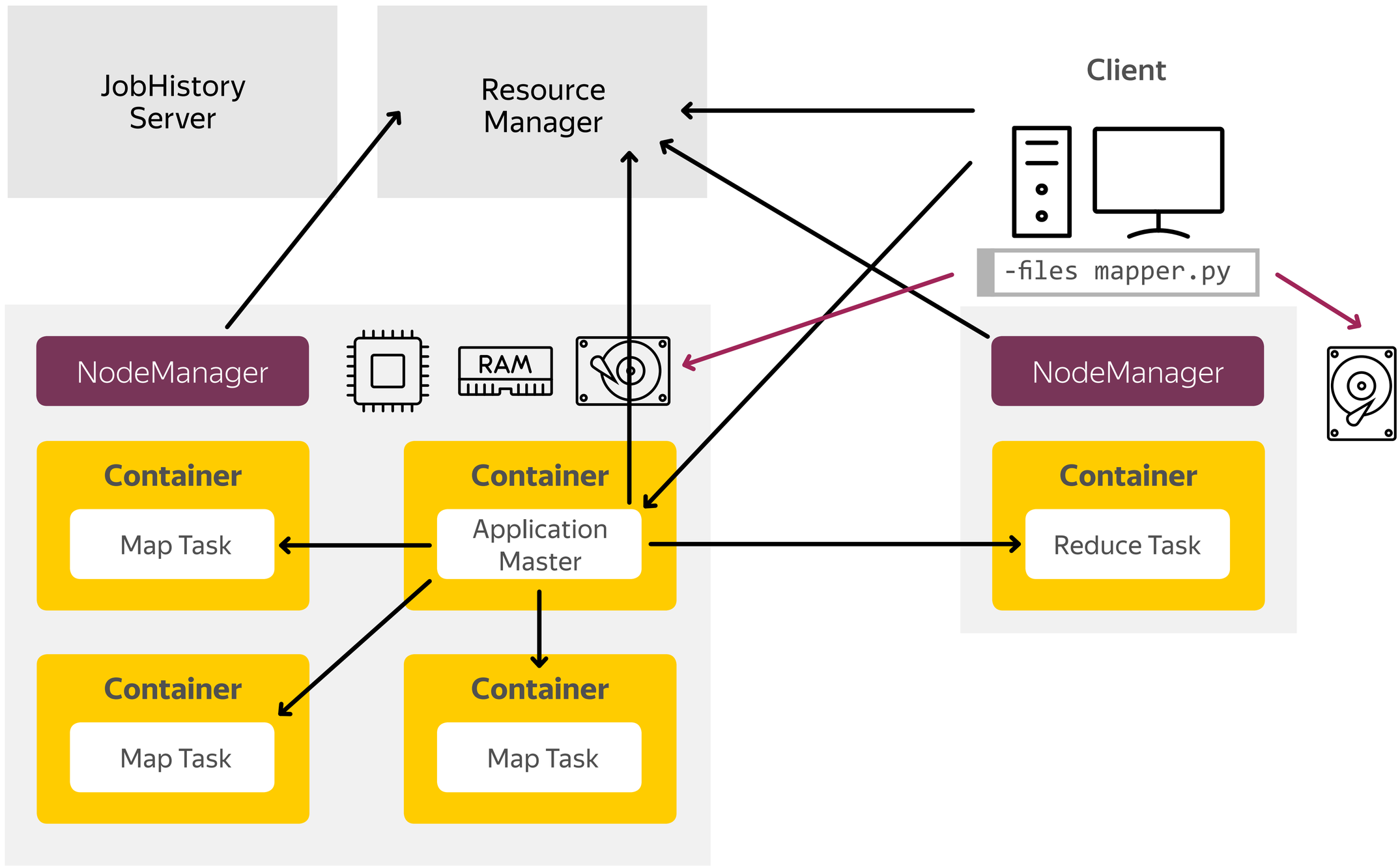
Mapper (Python): mapper.py

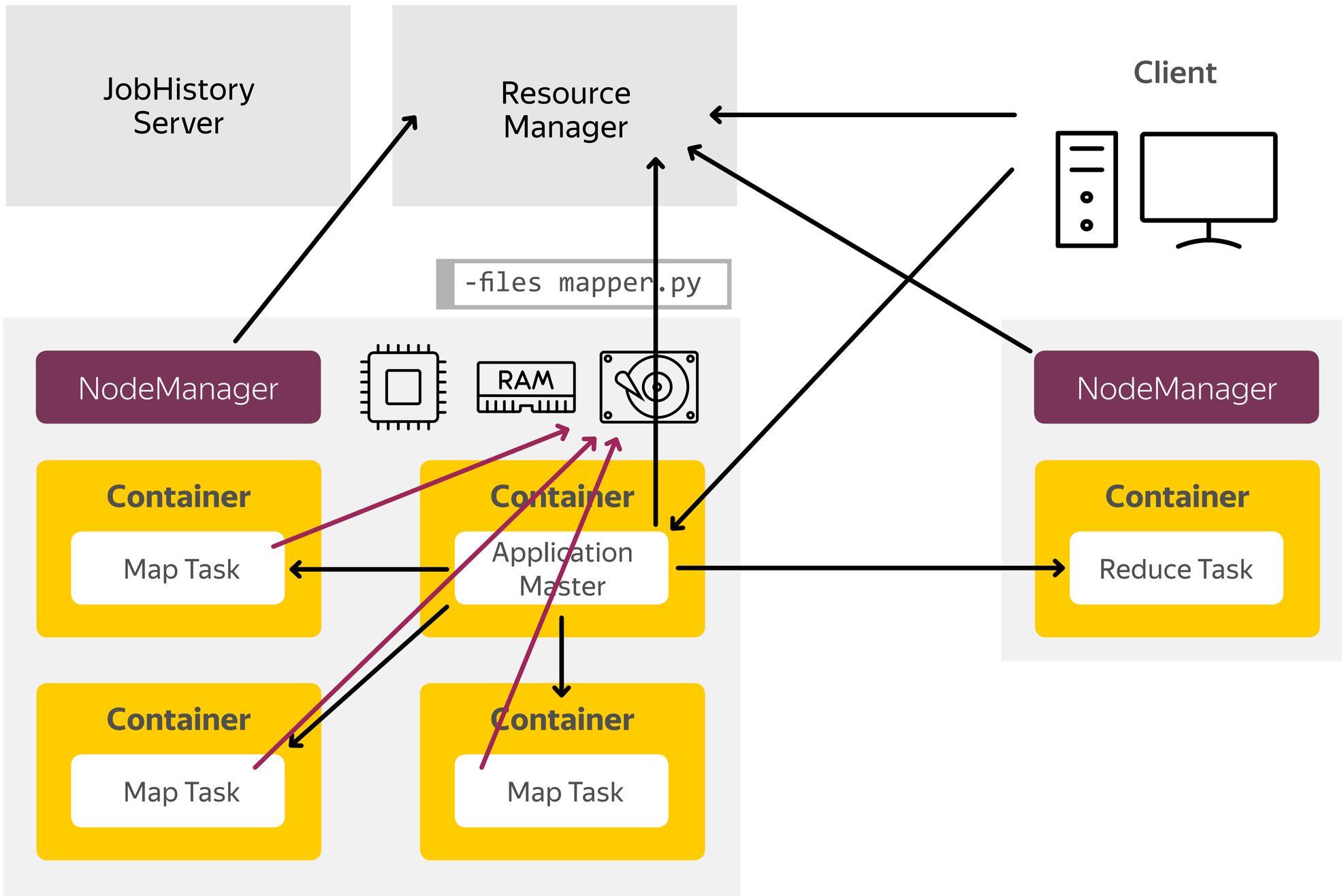
```
def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

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


JobHistory Server
Resource Manager
Client
NodeManager
NodeManager
Container
Map Task
Container
Application Master
Container
Reduce Task


JobHistory Server
Resource Manager
Client
NodeManager
RAM
NodeManager
Container
Map Task
Container
Application Master
Container
Reduce Task
Container
Map Task
Container
Map Task


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


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

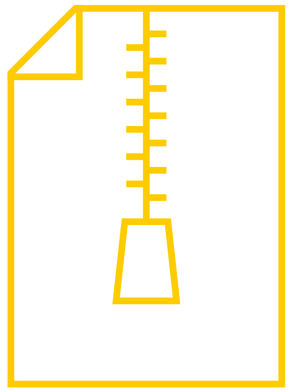
Distributed Cache

Distributed Cache

-files

Distributed Cache

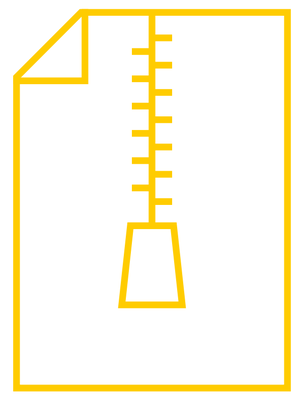
-files



-archives

Distributed Cache

-files



-archives

-libjars

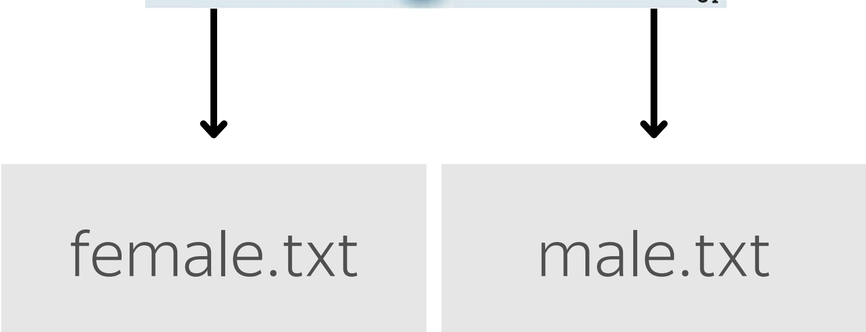
TOP 10 NAMES

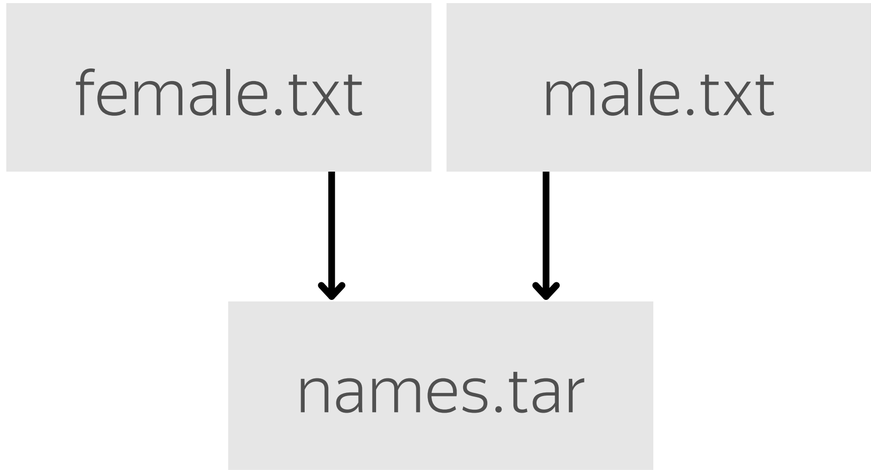
Figures for 2015

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

Source: BabyCentre


TOP 10 NAMES
GIRLS
Figures for 2015
BOYS
1 Olivia
2 Sophia
3 Lily
4 Emily
5 Amelia
6 Chloe
7 Isabelle
8 Sophie
9 Ella
10 Isabella
1 Muhammad
2 Oliver
3 Jack
4 Noah
5 Jacob
6 Harry
7 Charlie
8 Ethan
9 James
10 Thomas
Source: BabyCentre


female.txt
male.txt




```
$ 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,vocabulary.txt \
          -archives names.tar \
          -mapper 'python mapper.py' \
          -reducer 'python reducer.py' \
          -input /data/wiki/en_articles \
          -output word_count
```

```python
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")
```

```python
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")
```

```python
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")
```

TOP 10 NAMES

Figures for 2015

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

Source: BabyCentre


```
$ 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
```

Summary

Summary

- you know **how to distribute** files and archives to MapReduce workers

Summary

- you know **how to distribute** files and archives to MapReduce workers
- you know **how to use distributed** files and archives in MapReduce applications

Summary

- you know **how to distribute** files and archives to MapReduce workers
- you know **how to use distributed** files and archives in MapReduce applications
- you can identify the number of a distributed file available on each node (based on   )

BigDATAteam