

Хранение и Обработка Больших Объёмов Данных

Антон Горохов

старший разработчик, Яндекс

anton.gorokhov@gmail.com

План лекции

I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

II. Join

- 1) Reduce side
- 2) Map side
- 3) Bucket

III. Streaming

Старый и новый API

Старый

org.apache.hadoop.mapred.*

part-0000

Mapper,Reducer – интерфейсы

Новый

org.apache.hadoop.mapreduce.*

part-m-0000, part-r-0000

Mapper,Reducer – классы

подробнее:

<http://ruhadoop.blogspot.ru/2012/07/hadoop-api.html>

Функциональность примерно одинакова

В литературе в основном – новый API

Рекомендуется использовать новый

Параметры запуска задачи

\$ `hadoop [--config CONFDIR] jar <jar> [mainClass] args...`

args: `-Dparam=value`

`-files=file1,file2,...`

Для разбора **args** надо использовать Tool и ToolRunner:

```
public class UserCount extends Configured implements Tool {  
    public static int main(String[] args) throws Exception {  
        return ToolRunner.run(new UserCount(), args);  
    }  
}
```

@Override

```
public int run(String[] args) throws Exception {  
    Configuration conf = this.getConf();  
    conf.get("param");  
}
```

Параметры запуска задачи

\$ `hadoop [--config CONFDIR] jar <jar> [mainClass] args...`

В Configuration можно заносить свои параметры:

```
@Override
public int run(String[] args) {
    Configuration conf = this.getConf();
    conf.set("param", args[3]);
}
```

И получать их в mapper, reducer, etc через Context:

```
@Override
public void map(LongWritable offset, Text line,
    Context context) {
    String value = context.getConfiguration().get("param");
    ...
}
```

Счетчики

- Способ подсчета статистики в MR задаче
- Бывают встроенные и пользовательские
 - Встроенные – определены в Hadoop
 - Пользовательские

```
@Override
public void map(LongWritable lineNo, Text line,
                Context context) {
    ...
    context.getCounter("Mapper stat", "total lines")
        .increment(1);
}
```

- Прочитать: веб-интерфейс, консоль,
программно

```
job.getCounters().findCounter("Mapper stat", "total lines")
    .getValue();
```

План лекции

I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

II. Join

- 1) Reduce side
- 2) Map side
- 3) Bucket

III. Streaming

Интерфейс класса Mapper

- `Mapper<K1,V1,K2,V2>`
 - `void setup(Mapper.Context context)`
 - `void run(Mapper.Context context)`
 - `void map(K1 key, V1 value, Mapper.Context context)`
 - `void cleanup(Mapper.Context context)`

- Реализация:

```
public void run(Context context) {  
    setup(context);  
    while (context.nextKeyValue()) {  
        map(context.getCurrentKey(),  
            context.getCurrentValue(), context);  
    }  
    cleanup(context);  
}
```


Интерфейс класса Reducer

- `Reducer<K1,V1,K2,V2>`
 - `void setup(Reducer.Context context)`
 - `run(Reducer.Context context)`
 - `reduce(K2 key, Iterable<V2> values, Reducer.Context context)`
 - `cleanup(Reducer.Context context)`

- Реализация:

```
public void run(Context context) {  
    setup(context);  
    while (context.nextKey()) {  
        reduce(context.getCurrentKey(), context.getValues(),  
               context);  
    }  
    cleanup(context);  
}
```

Использование setup() и cleanup()

- setup() – инициализация
- cleanup() – вывод последнего ключа

Агрегируем
последовательные
ключи



write() в map или reduce()
при изменении ключа

последний ключ ->



write() в cleanup()

- Другие примеры – дальше

In-mapper combiner

- Пример: обратный индекс

Input: docid, content

Output: term, [(docid, tf), ...]

term – слово из content, tf – term frequency, сколько раз term в content

map: docid, content -> (term, docid), 1, ...

Агрегация в map(): docid, content -> (term, docid), N,...

- для одного вызова map()
 - т.к. term могут повторяться для одного docid
- для всех map(), вывод по завершении mapper'а
 - т.е. все записи выводятся в cleanup()

План лекции

I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

II. Join

- 1) Reduce side
- 2) Map side
- 3) Bucket

III. Streaming

Пример: обратный индекс

Input: docid, content

Output: term, [(docid, tf), ...]

Что если term – в каждом документе?

term1, [(docid, tf)]

term2, [(docid, tf)]

...

term1, docid, tf

term1, docid, tf

...

term2, docid, tf

...

- Как хранить?
- Как передавать на reduce?

Stripes vs. Pairs

Stripes

term1, [(docid, tf)]

term2, [(docid, tf)]

...

Pairs

term1, docid, tf

term1, docid, tf

...

term2, docid, tf

...

Stripes vs. Pairs

- Стратегия Pairs
 - больше записей, больше объем
 - нетребовательна к памяти при обработке
- Стратегия Stripes
 - более компактная запись, но сложное значение
 - удобна для локальной агрегации
 - требовательна к памяти при обработке
 - при «разумной» длине списка быстрее

План лекции

I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

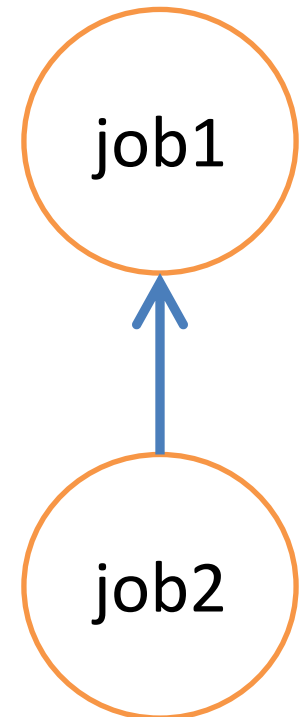
II. Join

- 1) Reduce side
- 2) Map side
- 3) Bucket

III. Streaming

Несколько Hadoop задач

- Пример: wordcount + сортировка по количеству
 - Задача1: посчитать статистику
 - Задача2: отсортировать
- “job2 зависит от job1”

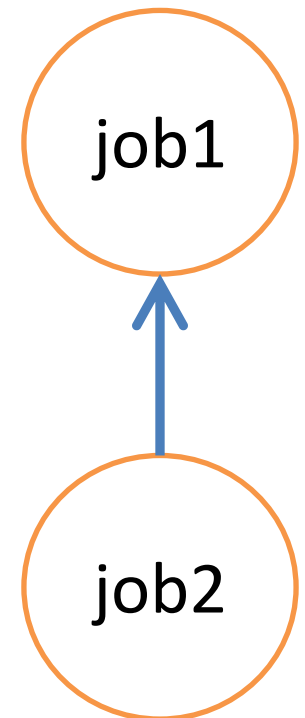


Несколько Hadoop задач

Реализация:

inPath -> **job1** -> tmpPath -> **job2** -> outPath

```
public int run(String[] args) throws Exception {  
    Job job1 = new Job(getConf());  
    ... // настройка job1  
    String tmpPath = outPath + ".tmp";  
    FileOutputFormat.setOutputPath(job1,  
        new Path(tmpPath));  
  
    Job job2 = new Job(getConf());  
    ... // настройка job2  
    FileOutputFormat.setInputPath(job2,  
        new Path(tmpPath));  
  
    job1.waitForCompletion(true);  
    job2.waitForCompletion(true);  
    fs.delete(new Path(tmpPath), true) // !!!  
}
```



Несколько Hadoop задач (2)

Реализация:

inPath -> **job1** -> tmpPath -> **job2** -> outPath

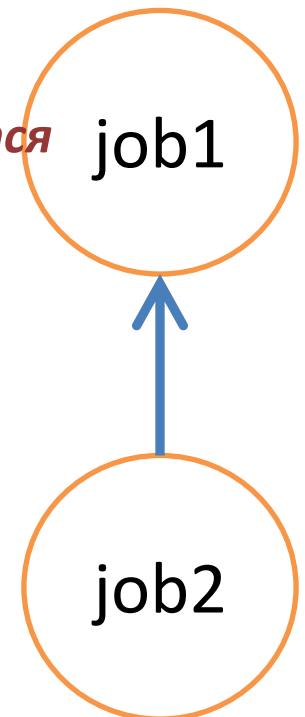
```
public int run(String[] args) throws Exception {  
    ...  
    ... // настройка job1  
    ... // настройка job2  
    job1.submit();  
    ... // сделать что-то еще  
  
    while (!job1.getStatus().isJobComplete()) {  
        System.out.println("Still running...");  
        Thread.sleep(3000);  
    }  
  
    status = job2.waitForCompletion(true) ? 0 : 1;  
    ...  
}
```

запуск задачи1

пока задача выполняется

*ждем завершения
задачи1*

*ждем завершения
задачи2*



Зависимые задачи: JobControl

```
JobControl control = new JobControl("jobs control");
```

```
control.addJob(cJob1);
```

cJob1 и cJob2 – объекты ControlledJob

```
control.addJob(cJob2);
```

```
cJob2.addDependingJob(cJob1);
```

cJob2 зависит от cJob1

```
Thread workflow = new Thread(control, "workflow");
```

*запускаем поток
с задачами*

```
workflow.start();
```

```
while (!control.allFinished()) {
```

```
    System.out.println("Still running...");
```

```
    Thread.sleep(3000);
```

ждем завершения

```
}
```

```
System.out.println("Done");
```

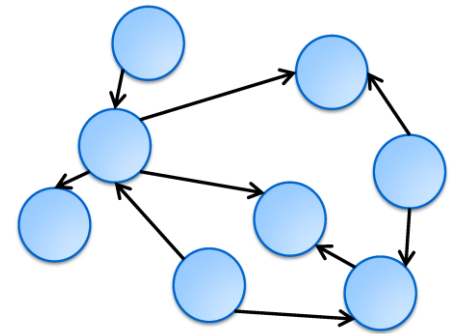
```
control.stop();
```

```
int status = control.getFailedJobList().size() > 0 ? 1 : 0;
```

проверяем результат

Зависимые задачи: общий случай

- Направленный ациклический граф (DAG)
- Топологическая сортировка
=> порядок вычислений
- Главное:
 - Соблюсти зависимости
 - Выяснить, что можно делать параллельно
- Реализации:
 - JobControl
 - Oozie – отдельный проект
 - Cascading, etc.



Композиция преобразований

- ChainMapper
 - extends Mapper<K1,V1,K2,V2>
 - addMapper(..., Mapper.class, ...) – добавить mapper в chain
- ChainReducer
 - extends Reducer<K1,V1,K2,V2>
 - setReducer(... Reducer.class, ...) – установить reducer
 - addMapper(..., Mapper.class, ...) – добавить mapper в chain после reducer'а

Композиция преобразований

- Пример: IntSumReducer + InverseMapper

```
ChainReducer.setReducer(job, IntSumReducer.class, Text.class,  
IntWritable.class, Text.class, IntWritable.class, conf);  
ChainReducer.addMapper(job, InverseMapper.class, Text.class,  
IntWritable.class, IntWritable.class, Text.class, conf);
```

- При вызове приходится добавлять явно jar-файлы и
HADOOP_CLASSPATH:

```
$ HADOOP_CLASSPATH=/usr/lib/hadoop/hadoop-  
common.jar:/usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-  
core.jar \  
hadoop jar jar/wordCount.jar ru.mipt.WordCount  
-libjars /usr/lib/hadoop/hadoop-  
common.jar,/usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-  
core.jar  
INPATH OUTPATH
```

План лекции

I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

II. Join

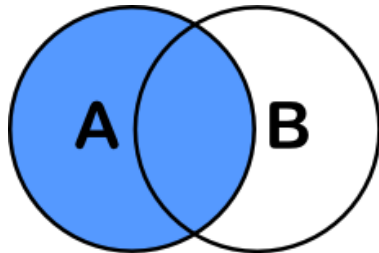
- 1) Reduce side
- 2) Map side
- 3) Bucket

III. Streaming

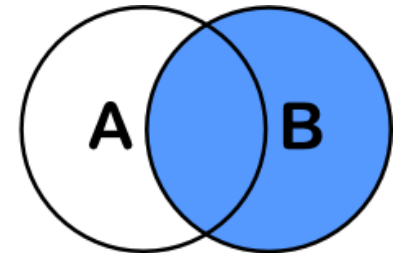
Join

- Данные обычно денормализованы. Но не всегда.
- Примеры:
 - Лог-файл посещений и анкетные данные людей, связь по `user_id`
 - 2 лог-файла, связаны по `user_id` и `timestamp`
- 2 датасета: A и B, надо объединить по некоторому признаку

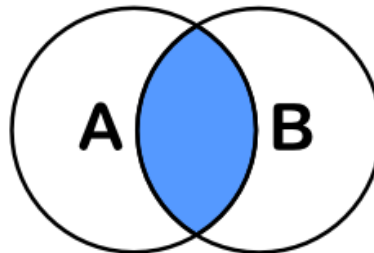
Что значит «объединить»?



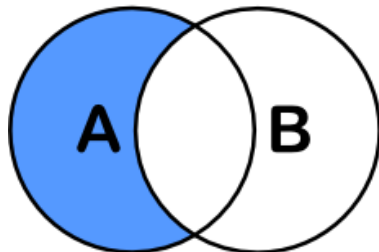
```
SELECT <auswahl>  
FROM tabelleA A  
LEFT JOIN tabelleB B  
ON A.key = B.key
```



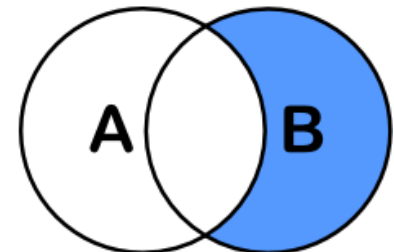
```
SELECT <auswahl>  
FROM tabelleA A  
RIGHT JOIN tabelleB B  
ON A.key = B.key
```



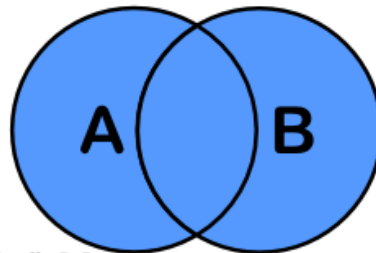
```
SELECT <auswahl>  
FROM tabelleA A  
INNER JOIN tabelleB B  
ON A.key = B.key
```



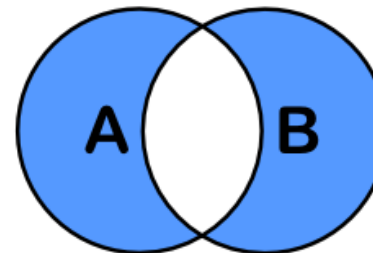
```
SELECT <auswahl>  
FROM tabelleA A  
LEFT JOIN tabelleB B  
ON A.key = B.key  
WHERE B.key IS NULL
```



```
SELECT <auswahl>  
FROM tabelleA A  
RIGHT JOIN tabelleB B  
ON A.key = B.key  
WHERE A.key IS NULL
```

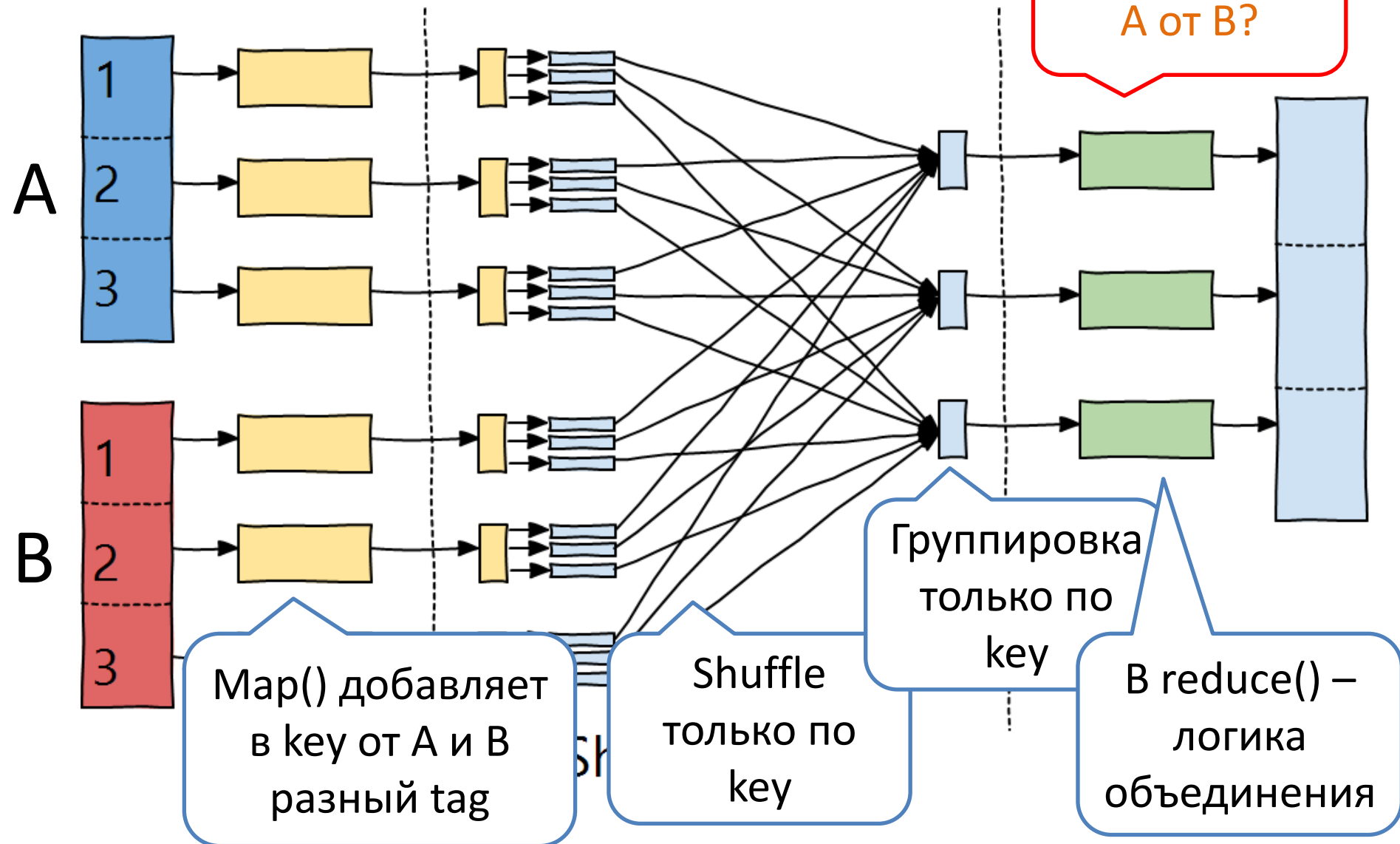


```
SELECT <auswahl>  
FROM tabelleA A  
FULL OUTER JOIN tabelleB B  
ON A.key = B.key
```



```
SELECT <auswahl>  
FROM tabelleA A  
FULL OUTER JOIN tabelleB B  
ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL
```

Reduce side join



Reduce side join

1. Как отличить A и B в Mapper

```
private static class JoinMapper extends Mapper... {  
    public void setup(Context context) {  
        pathA = conf.get("pathA");  
        FileSplit split = (FileSplit)context.getInputSplit();  
        path = split.getPath();  
        String tag = path.toString().startsWith(pathA) ? "A" : "B";  
    }  
}
```

В map() добавляем tag к ключу:

key, value -> (key, tag), value

Reduce side join

2. Shuffle (key, tag) только по key

```
public class TagJoinPartitioner extends Partitioner<TaggedKey,Text> {  
    @Override  
    public int getPartition(TaggedKey taggedKey, Text text, int R) {  
        return taggedKey.getKey().hashCode() % R;  
    }  
}
```

Reduce side join

3. Группировка только по key


```
public class TagGroupComparator extends WritableComparator {  
  
    @Override  
    public int compare(WritableComparable a, WritableComparable b) {  
        return a.getKey().compareTo(b.getKey());  
    }  
}
```

// В настройках задачи:

```
job.setGroupingComparatorClass(TagGroupComparator.class);
```

4. Но сортировка по обоим полям (key, tag)

- Так реализовали тип составного ключа

На входе reduce(): 

кстати, Secondary Sort

- Упорядочить значения на входе reduce()
- Решения:
 - Сортировка в памяти внутри reduce()
 - GroupingComparator, value-to-key conversion:
key, value -> (key, subkey), value

Reduce side join

- + Всегда работает

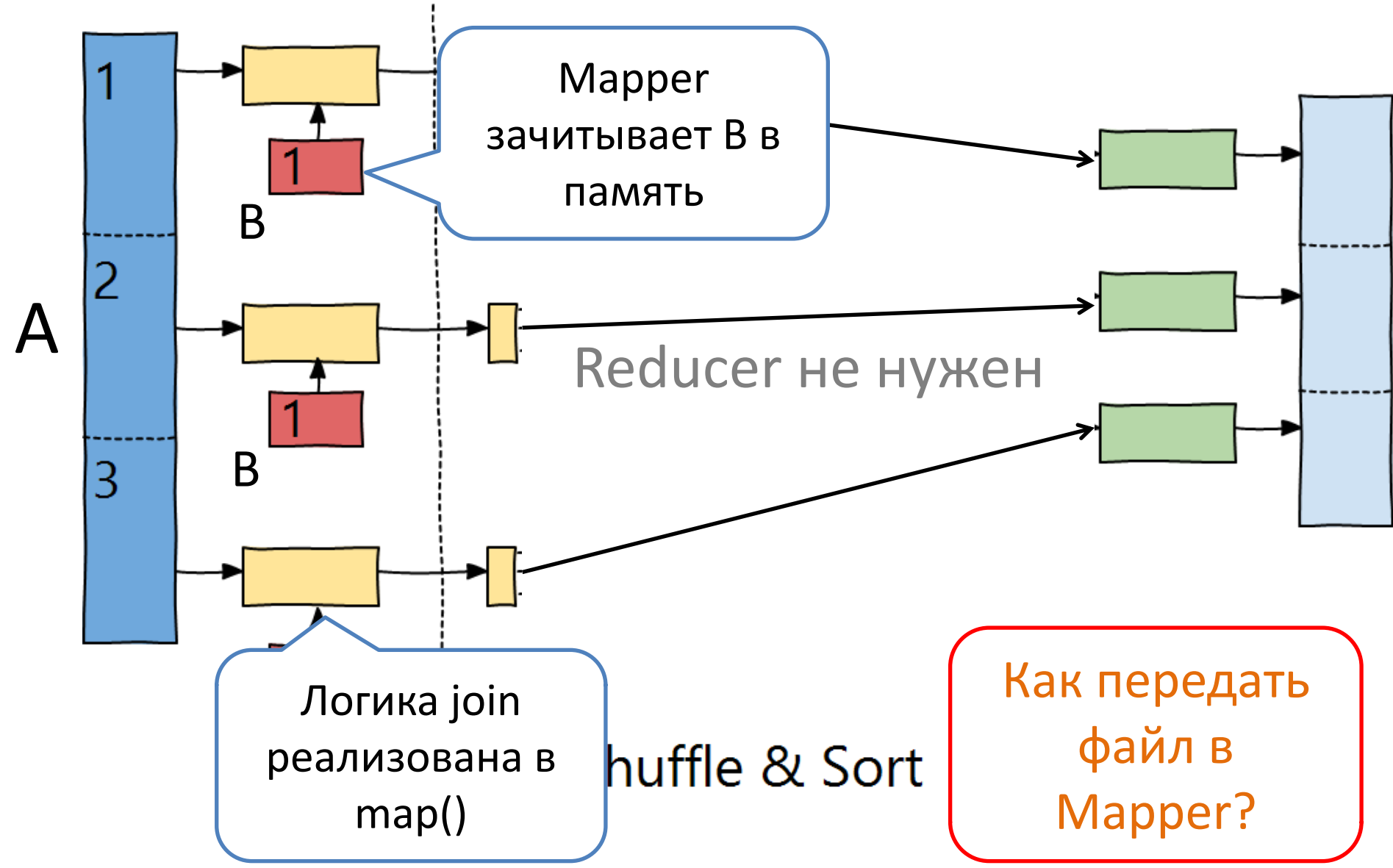
- Накладно

Если много значений на один ключ

- см. Pairs vs. Stripes

Можно ли ускорить Join?

Map-side join



Map-side join

Чтение файлов в Mapper

- class FileSystem

```
Path path = new Path (filename);  
FileSystem fs = FileSystem.get(conf);  
BufferedReader reader = new BufferedReader(  
    new InputStreamReader(fs.open(path)));  
String line = reader.readLine ();
```

- class DistributedCache

Distributed cache

- Содержит файлы, необходимые для работы конкретного task
 - По одной копии на ноду
 - т.е. менять их нельзя
 - Файлы указываются в параметрах:
 - files
 - archives – будут разархивированы
 - libjars – будут добавлены в CLASSPATH)
- ```
$ hadoop jar <jar_file> <class> -files ...
```

# Distributed cache

- Доступ – через классы `java.io.*`
  - например, в методах `Mapper.setup()`, `Reducer.setup()`

**NB:** файл читаем из текущей директории задачи

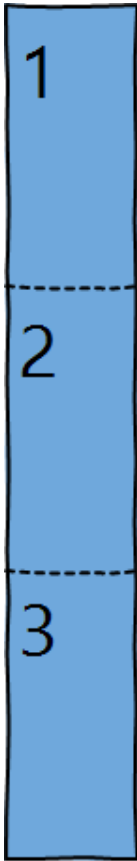
Пример:

```
$ hadoop jar join.jar ru.mipt.Join
 -files /tmp/setB.txt setA_dir out_dir
```

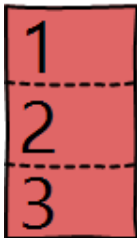
В методе `Mapper.setup()`:

```
BufferedReader reader = new BufferedReader(
 new FileReader("setB.txt"));
String line;
while ((line = reader.readLine()) != null) {
 ...
}
```

# Bucket side join

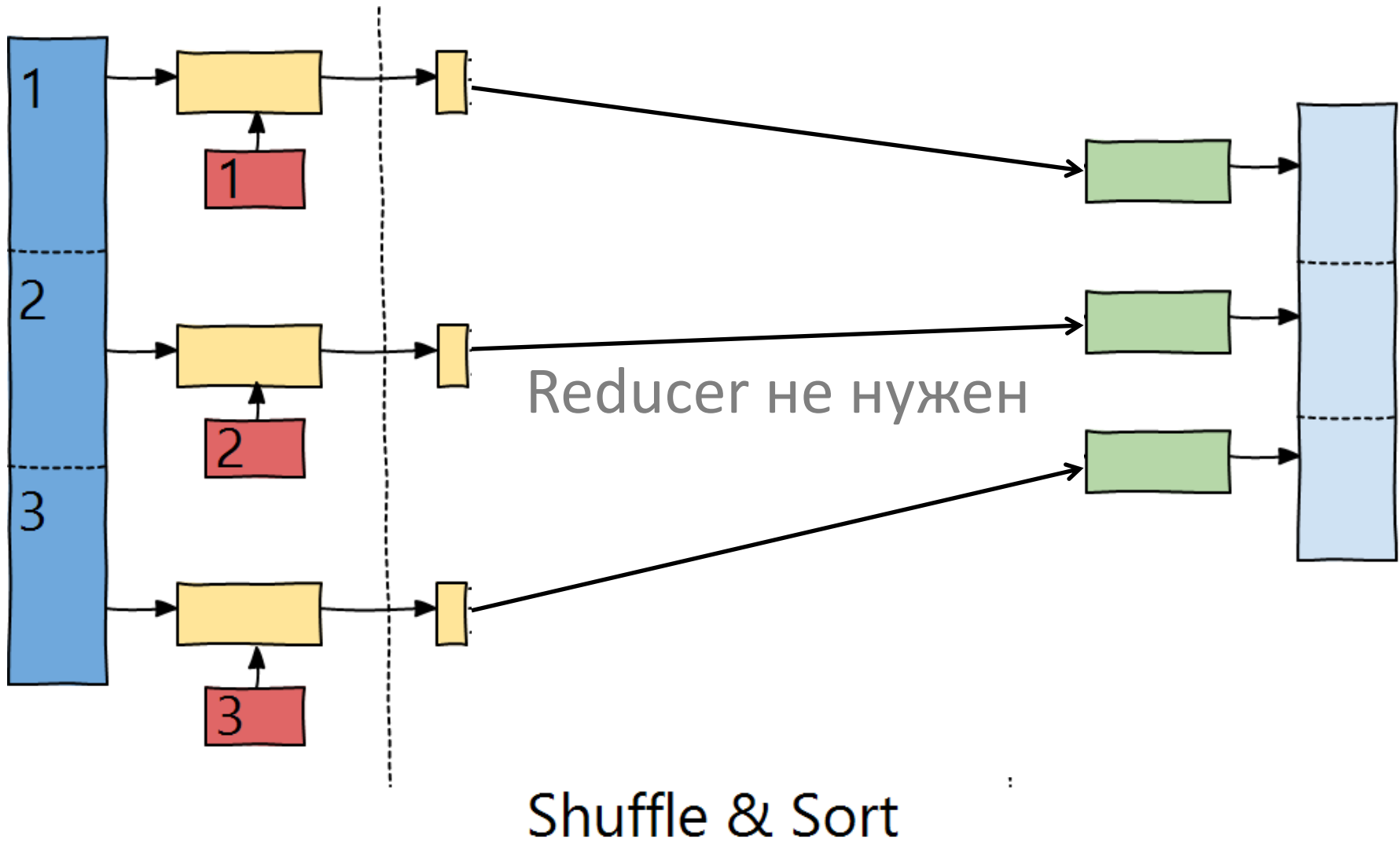


- Меньший датасет (**B**) не влезает в память
- Большой датасет (**A**) разбит на диапазоны ключей
- Mapper читает в память только те ключи из **B**, которые он будет обрабатывать из **A**
  - **B** заранее разбит на такие же диапазоны
  - или читаем все, но оставляем только нужные



Если **A** и **B** – результаты MR задачи, то можно подготовить разбиение на диапазоны в ней

# Bucket side join



# План лекции

## I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

## II. Join

- 1) Reduce side
- 2) Map side
- 3) Bucket

## III. Streaming

# Streaming

- Вспомним user (или word) count

```
$ cat access.log | get_uid.sh | sort | uniq -c
```

read                      map                      sort                      reduce

- То же на streaming:

```
$ hadoop jar hadoop-streaming.jar
 -input in_dir
 -output out_dir
 -mapper get_uid.sh
 -reducer uniq -c
 -file get_uid.sh
```



# Streaming

- Mapper и reducer – программы, команды или Java-класс
- Читают из stdin, пишут в stdout
- Sort & shuffle обеспечивает hadoop
- Reducer вызывается на каждую запись  
– т.е. смену ключа надо отслеживать самому

# Streaming: WordCount

- Mapper (python)

```
for line in sys.stdin:
 words = line.strip().split()
 foreach word in words:
 print "\t".join([word, "1"])
```

- Reducer

```
current_key = None
sum = 0
for line in sys.stdin:
 key, value = line.strip().split("\t", 1)
 if key != current_key:
 if current_key:
 print "\t".join(current_key, str(sum))
 current_key = key
 sum = 0
 sum += int(value)
if current_key:
 print "\t".join(current_key, str(sum))
```

# Streaming

- + Любой язык
- + Скорость разработки
- Возможностей меньше, чем в Java
- Программа – набор hadoop-команд
- Все нестандартные модули надо установить на весь кластер
  - или заносить в Distributed Cache

# Streaming

<http://hadoop.apache.org> -> Hadoop Streaming

```
hadoop jar hadoop-streaming-2.6.0.jar \
-D mapreduce.map.output.key.field.separator=. \
-D mapreduce.partition.keypartitioner.options=-k1,2 \
-D mapreduce.fieldsel.data.field.separator=. \
-D mapreduce.fieldsel.map.output.key.value.fields.spec=6,5,1-3:0- \
-D mapreduce.fieldsel.reduce.output.key.value.fields.spec=0-2:5- \
-D mapreduce.map.output.key.class=org.apache.hadoop.io.Text \
-D mapreduce.job.reduces=12 \
-input myInputDirs \
-output myOutputDir \
-mapper org.apache.hadoop.mapred.lib.FieldSelectionMapReduce \
-reducer org.apache.hadoop.mapred.lib.FieldSelectionMapReduce \
-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

# Streaming: Hadoop + Python

- Dumbo

```
def mapper(key, value):
 for word in value.split():
 yield word, 1

def reducer(key, values):
 yield key, sum(values)

if __name__ == "__main__":
 import dumbo
 dumbo.run(mapper, reducer, combiner=reducer)
```

- А также mrjob, hadoop, pydoop

<http://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/>

# Java + Python

- Jython

- `cm. src/examples/python/WordCount.py`

```
from org.apache.hadoop.fs import Path
from org.apache.hadoop.io import *
from org.apache.hadoop.mapred import *

import sys
import getopt

class WordCountMap(Mapper, MapReduceBase):
 one = IntWritable(1)
 def map(self, key, value, output, reporter):
 for w in value.toString().split():
 output.collect(Text(w), self.one)
```

- не совместим со scipy

# Вопросы?

## I. «Tips of the day»

- 1) Интерфейсы Mapper и Reducer
- 2) Стратегии Stripes и Pairs
- 3) Последовательности задач

## II. Join

- 1) Reduce side
- 2) Map side
- 3) Bucket

## III. Streaming