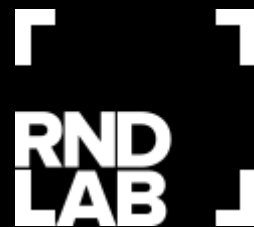


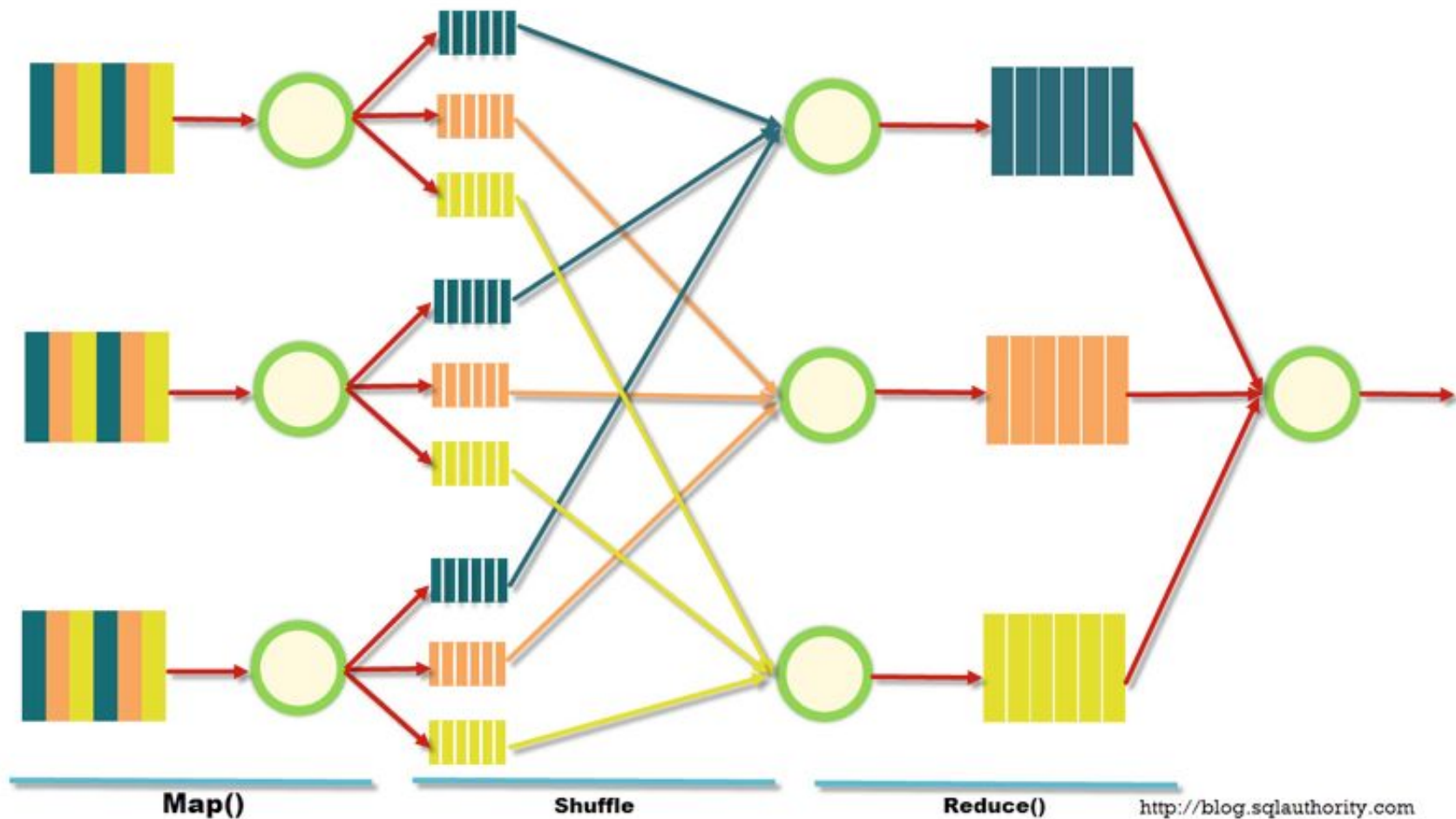


# Приёмы и стратегии работы с MapReduce

Kirill Danilyuk, DS @ RnD Lab



## How MapReduce Works?



# Уровни абстракции MapReduce

**Уровень 0:** примитивы map и reduce

**Уровень 1:** концептуальная модель в рамках [пейпы от Google 2004 года](#)

**Уровень 2:** (сегодняшнее занятие)  
утилизация парадигмы MapReduce в виде кода, реализующего функции map и reduce

**Уровень 3:** реализация MapReduce в виде фреймворка, абстрагирующего инженерные аспекты работы

```
map(String key, String value):  
    // key: document name  
    // value: document contents  
    for each word w in value:  
        EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):  
    // key: a word  
    // values: a list of counts  
    int result = 0;  
    for each v in values:  
        result += ParseInt(v);  
    Emit(AsString(result));
```

# Map, Mapper, map

**Map:** шаг в терминах фреймворка

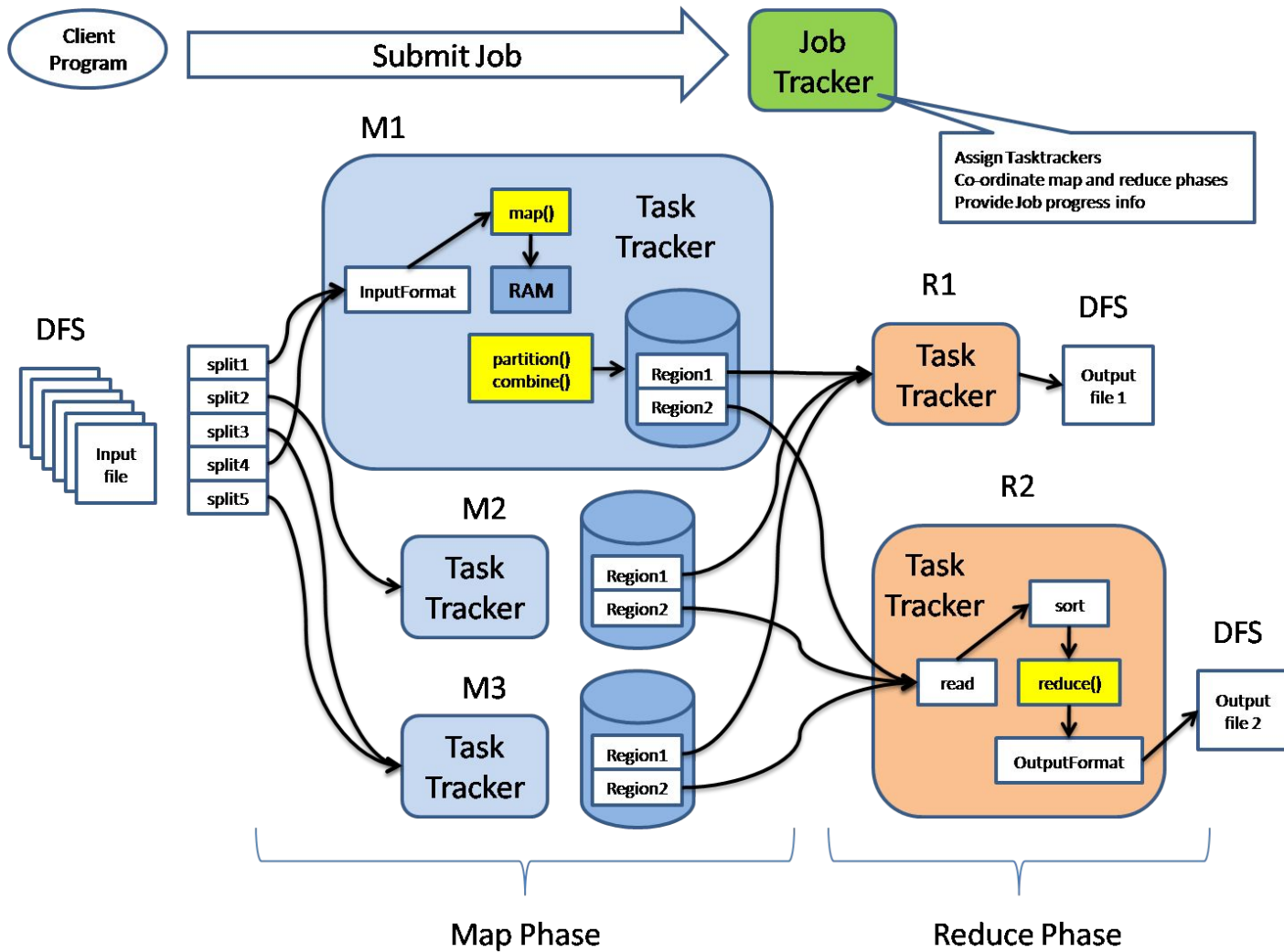
**Mapper:** интерфейс в терминах Java API. Отображает исходные key/value пары в набор intermediate key/value пар

**map:** процедура, которую необходимо реализовать в рамках интерфейса. Трансформирует одну пару key с value.

```
public class TokenCounterMapper
    extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}
```



# **0. Word Count (пример)**

# Работа на кластере

Работать будем на master-машине, зайдите на неё сейчас по ssh.

Скопируйте себе в домашнюю директорию master-машины данные:

```
cp -R /tmp/mapreduce_seminar/data ~/.
```

После занятия ответы будут доступны в директории

```
/tmp/mapreduce_seminar/solutions
```

 master-ноды.

## Локальная отладка

```
$ cat data/doc.txt | python3 mapper.py | sort  
-k1,1 | python3 reducer.py > data/result.txt
```



# Вспомним Word Count

```
# mapper.py
```

```
import sys
```

```
for line in sys.stdin:
    for token in line.strip().split():
        print(token + '\t1')
```

```
# reducer.py
```

```
import sys
```

```
prev_key = None
```

```
sum = 0
```

```
for line in sys.stdin:
    key, value = line.split("\t")
    if key != prev_key and prev_key is not None:
        print(prev_key, sum)
        sum = 0
    sum += 1
    prev_key = key
if key != prev_key and prev_key is not None:
    print(prev_key, sum)
```

```
import sys
```

```
# Accumulators  
prev_key = None  
values = []
```

```
# reduce() method
```

```
def do_reduce(key, values):  
    yield key, sum(values)
```

```
# Reducer
```

```
for line in sys.stdin:  
    key, value = line.split("\t")  
    if key != prev_key and prev_key is not None:  
        for k, v in do_reduce(prev_key, values):  
            sys.stdout.write('{},{}\n'.format(k, str(v)))  
        values = []  
    values.append(int(value))  
    prev_key = key
```

```
# Off by one correction
```

```
if key != prev_key and prev_key is not None:  
    for k, v in do_reduce(prev_key, values):  
        sys.stdout.write('{},{}\n'.format(k, str(v)))
```

# Word Count (Reducer)

# Оформление задачи

# mapper:

map(x) →

```
for token in x.split():  
    emit(token, 1)
```

# reducer:

reduce(key, values) →

```
    emit(key, sum(values))
```

# **1. Поиск отличников**

# Поиск отличников

Дан файл вида студент<tab>оценка

- Фейн 3
- Куликов 4
- Петровна 3
- Мордовина 4
- Коробейников 5
- Кремнев 5
- Мордовина 3

Студент считается отличником, если его средний балл  $\geq 4.5$ . Найти всех отличников в файле.

# Поиск отличников

```
# mapper
```

```
map(x, score) →  
    emit (x, score)
```

```
# reducer
```

```
reduce(x, scores) →  
    avg = average(scores)  
    if avg >= 4.5:  
        emit(x, avg)
```

# Поиск отличников (решение)

```
# mapper

import sys
for line in sys.stdin:
    print(line.strip())

# reducer
import sys

prev_student = None
marks = []

def emit_top_student_result(student, mk):
    avg = sum(mk) / len(mk)
    if avg >= 4.5: print('{:}: {:.2f}'.format(student, avg))

for line in sys.stdin:
    key, value = line.strip().split("\t")
    if key != prev_student and prev_student is not None:
        emit_top_student_result(prev_student, marks)
        marks = []
    marks.append(float(value))
    prev_student = key

if prev_student is not None:
    emit_top_student_result(prev_student, marks)
```

## **2. Гистограмма оценок**



# Гистограмма

Ожидаемый результат:

```
3.1      *
3.2      ****
3.3      *****
3.4      *****
3.5      *****
3.6      *****
3.7      *****
3.8      *****
3.9      *****
4.0      *****
4.1      *****
4.2      *****
4.3      *****
4.4      *****
4.5      *****
4.6      *****
4.7      *****
4.8      *****
```

# Гистограмма (решение)

## JOB 1:

- mapper – пустой
- reducer(key, value) →  
    emit(avg(values), 1)

## JOB 2:

- mapper – пустой
- reducer(key, values) ->  
    emit (key, '\*'\*sum(values))

# Гистограмма (код)

```
# Reduce 1
```

```
import sys
```

```
prev_student = None  
marks = []
```

```
def emit_student_result(student, mk):  
    res = sum(mk) / len(mk)  
    print('{:.1f}\t{}'.format(res, 1))
```

```
for line in sys.stdin:  
    key, value = line.strip().split("\t")  
    if key != prev_student and prev_student is not None:  
        emit_student_result(prev_student, marks)  
        marks = []  
    marks.append(float(value))  
    prev_student = key
```

```
if prev_student is not None:  
    emit_student_result(prev_student, marks)
```

```
# Reduce 2
```

```
import sys
```

```
prev_mark = None  
marks = []
```

```
def emit_histogram_value(mark, ones):  
    res = '*' * sum(ones)  
    print('{:10}\t{}'.format(mark, res))
```

```
for line in sys.stdin:  
    key, value = line.strip().split("\t")  
    if key != prev_mark and prev_mark is not  
None:  
        emit_histogram_value(prev_mark, marks)  
        marks = []  
    prev_mark = key  
    marks.append(1)
```

```
if prev_mark is not None:  
    emit_histogram_value(prev_mark, marks)
```

### **3. Map-Only Jobs**

# Map-Only Jobs

- Исходный файл - как в примере №1
- Оставить записи только по студентам с фамилией начинающейся на «П»

```
mapper(key, value) ->  
    if key.startswith('П'):  
        emit (key, value)
```

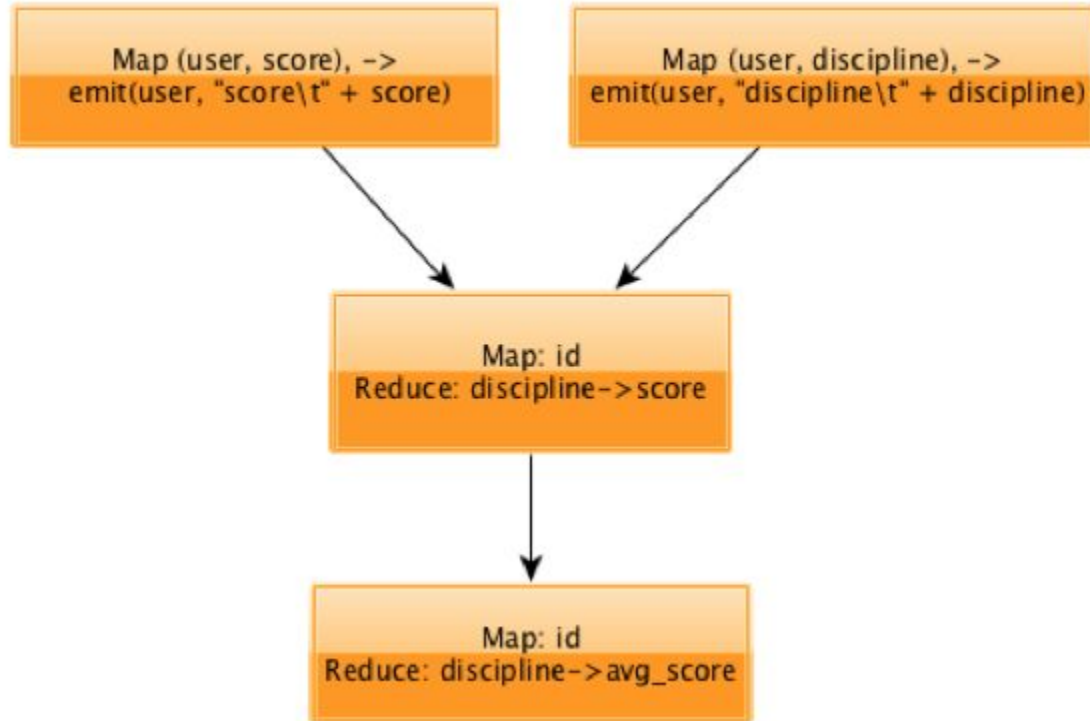
## **4. Reduce Joins**

# Средняя оценка vs любимый предмет

- **Файл 1** — как в первом задании
- **Файл 2** — `<user>\t<любимый предмет>`

Посчитать среднюю оценку среди любителей данного предмета

# Reduce Joins





# Код мапперов

```
# Mapper 1
import sys
for line in sys.stdin:
    user, score =
line.strip().split()
    print (user + "\t" +
"score" + "\t" + score)
```

```
# Mapper 2
import sys
for line in sys.stdin:
    user, score =
line.strip().split()
    print (user + "\t" +
"discipline" + "\t" + score)
```

# Reduce Join

```
import sys
prev_key = None
values = []

def print_res(key, values):
    scores = [float(score[1]) for score in filter(lambda value: value[0] == 'score', values)]
    avg = sum(scores)/len(scores)
    discipline = list(filter(lambda value: value[0] == 'discipline', values))[0][1]
    print("%s\t%.2f" % (discipline, avg))

for line in sys.stdin:
    key, record_type, value = line.strip().split("\t")
    if key != prev_key and prev_key is not None:
        print_res(prev_key, values)
        values = []
    values.append((record_type, value))
    prev_key = key

if prev_key is not None:
    print_res(prev_key, values)
```

# Reducer (2)

```
import sys
prev_key = None
marks = []

def print_res(key, values):
    avg = sum(values)/len(values)
    print("%s\t%.2f" % (key, avg))

values = []
for line in sys.stdin:
    key, value = line.strip().split("\t")
    if key != prev_key and prev_key is not None:
        print_res(prev_key, values)
        values = []
    values.append(float(value))
    prev_key = key

if prev_key is not None:
    print_res(prev_key, values)
```

## Reduce Join: локальная отладка

```
cat data/student_scores.txt | python mapper_3_1.py > data/tmp
cat data/student_disciplines.txt | python mapper_3_2.py >>
data/tmp
sort -k1,1 data/tmp | python reducer3_1.py > data/reducer3_1_output
sort -k1,1 data/reducer3_1_output | python reducer3_2.py
```

# Distributed Cache

Механизм в hadoop позволяющий добавить ресурсы в окружение map-reduce задачи

## Синтаксис:

```
yarn jar ... streaming.jar ... -file 'data.txt'
```

Можно использовать data.txt как локальный файл

Кроме локальных файлов можно добавлять данные с HDFS

## **5. Map Joins**

# Map Join

Подсчет средней оценки по предмету без усреднения по пользователям

# Map Join

```
import sys
student_disciplines = {}
for line in open('student_disciplines.txt'):
    student, discipline = line.strip().split("\t")
    student_disciplines[student] = discipline

for line in sys.stdin:
    student, score = line.strip().split('\t')
    print(student_disciplines[student] + "\t" + score)
```

```
import sys
prev_key = None
values = []

def print_res(key, values):
    avg = sum(values)/len(values)
    print("%s\t%.2f" % (key, avg))

for line in sys.stdin:
    key, value = line.strip().split("\t")
    if key != prev_key and prev_key is not None:
        print_res(prev_key, values)
        values = []
    values.append(float(value))
    prev_key = key

if prev_key is not None:
    print_res(prev_key, values)
```



**Спасибо!**