BIGDATA TEAM

MegaFon Course: Big Data

Hive. SQL over Big Data

Dral Alexey, aadral@bigdatateam.org
CEO at BigData Team, http://bigdatateam.org/
https://www.facebook.com/bigdatateam/

08.07.2019, Moscow, Russia

BIGDATA TEAM

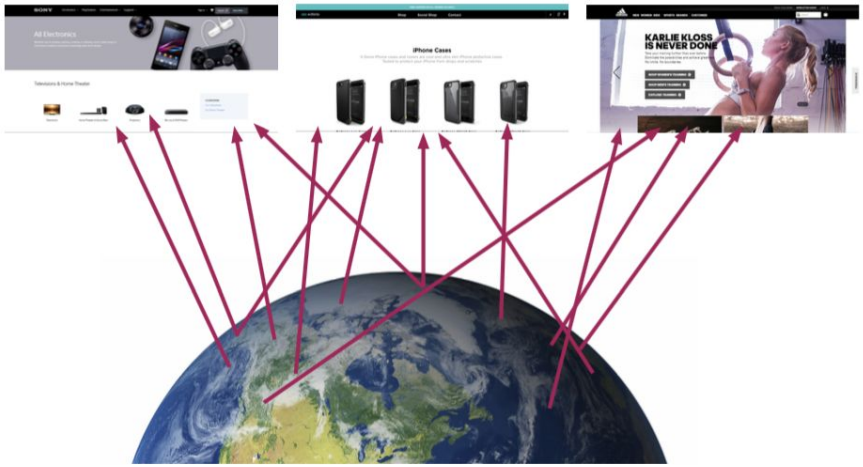
На этом занятии

- [18:30 - 19:20] Мотивация Big Data SQL, Map-Side Join
- -- перерыв, Q&A
- [19:30 - 20:10] Reduce-Side Join + оптимизации
- -- перерыв, Q&A
- [20:20 - 21:30] Hive Workshop
  - External / Managed таблицы в Hive + RegexpSerDe

Мотивация Big Data SQL

BIGDATA TEAM

1. Наиболее популярные регионы



Web-service
access logs

Geobase

IPv4: 109.188.67.224

↓

area: Moscow City Center

↓

city: Moscow

↓

country: Russia

↓

Earth

109.188.67.224, Moscow City Centre, Moscow, Russia, Earth
109.188.67.221, Moscow City Centre, Moscow, Russia, Earth
...

BIGDATA TEAM

1. Наиболее популярные регионы

Web-service access logs

format: ip, request, status_code, ...

Geobase

format: ip, $region_{city}$, $region_{country}$,...

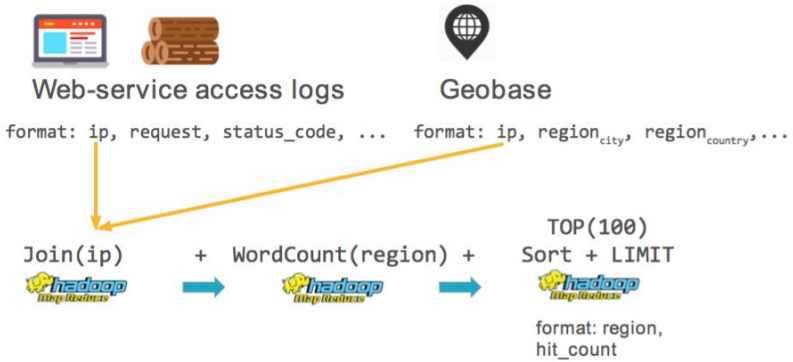
Join(ip) + WordCount(region) +

TOP(100)
Sort + LIMIT

format: region,
hit_count

BIGDATA TEAM

1. Наиболее популярные регионы



WordCount

Hive QL

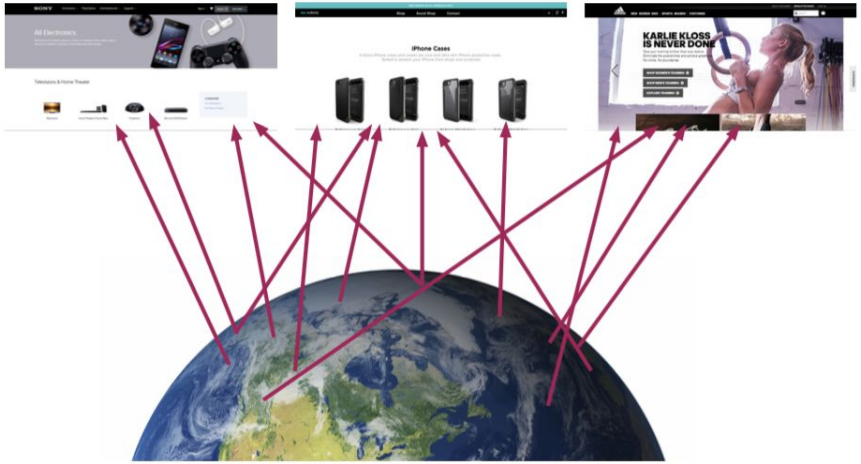
```
SELECT regioncity, COUNT(1) AS hit_count
FROM access_log JOIN geo_base
ON (access_log.host = geo_base.host)
GROUP BY regioncity ORDER BY hit_count LIMIT 100
```

join

TOP(100)

BIGDATA TEAM

2. Доля роботных запросов



Web-service
access logs

bot name + user_agent$_1$ ip$_1$
user_agent$_2$ + ip$_2$
... ...

[ro]bot database

BIGDATA TEAM

2. Доля роботных запросов

Web-service access logs

format: ip, request, user_agent, ...

[ro]bot database

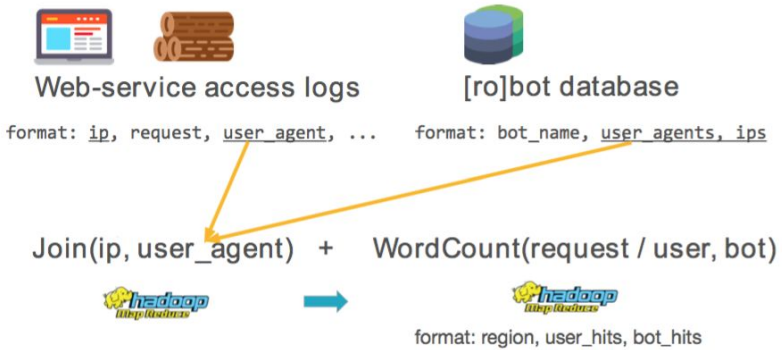
format: bot_name, user_agents, ips

Join(ip, user_agent) + WordCount(request / user, bot)

format: region, user_hits, bot_hits

BIGDATA TEAM

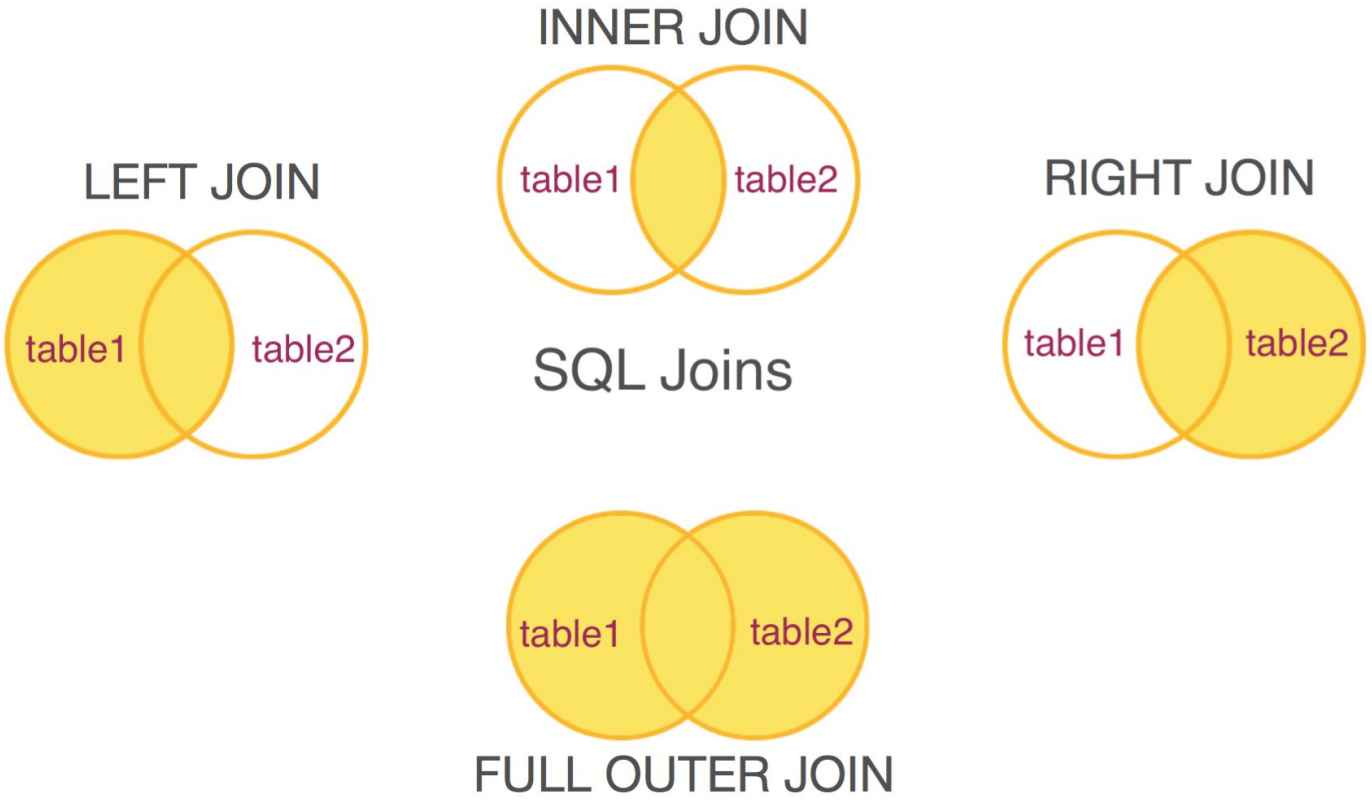
2. Доля роботных запросов



Hive QL

```
SELECT request,
    SUM(IF(robot.bot_name IS NULL, 1, 0)) as user_hit_count,
    SUM(IF(robot.bot_name IS NOT NULL, 1, 0)) as bot_hit_count
FROM access_log LEFT OUTER JOIN robot ON (
    access_log.host = robot.host
    AND access_log.user_agent = robot.user_agent
)
GROUP BY request
```

Join Operations



Join: **Q&A**

LEFT JOIN

a	b
1	null
2	null
3	3
4	4

INNER JOIN

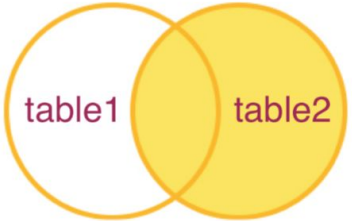
a	b
3	3
4	4

A	B
1	3
2	4
3	5
4	6

FULL OUTER JOIN

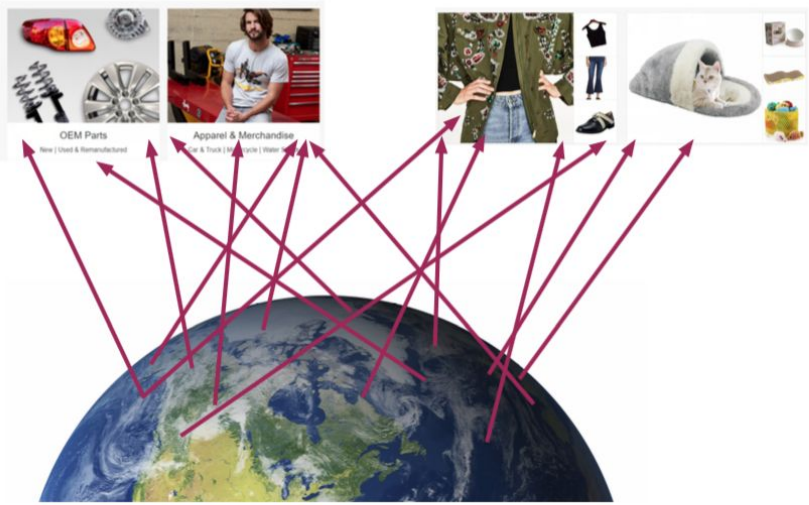
a	b
1	null
2	null
3	3
4	4
null	6
null	5

RIGHT JOIN



BIGDATA TEAM

3. Гендерное распределение



	age	gender	occupation	zipcode
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	05201
9	29	M	student	01002
10	53	M	lawyer	90703

Web-service access logs

Geobase

User personal data

BIGDATA TEAM

3. Гендерное распределение в регионах

Web-service access logs

Geobase

User personal data

format: ip, request, user_agent, ...

format: user_id, gender, age, ...

format: ip, region$_{city}$, region$_{country}$, ...

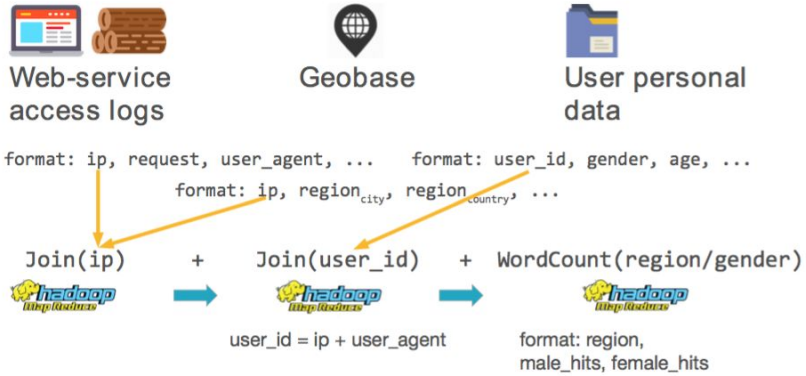
Join(ip) + Join(user_id) + WordCount(region/gender)

user_id = ip + user_agent

format: region, male_hits, female_hits

BIGDATA TEAM

3. Гендерное распределение в регионах



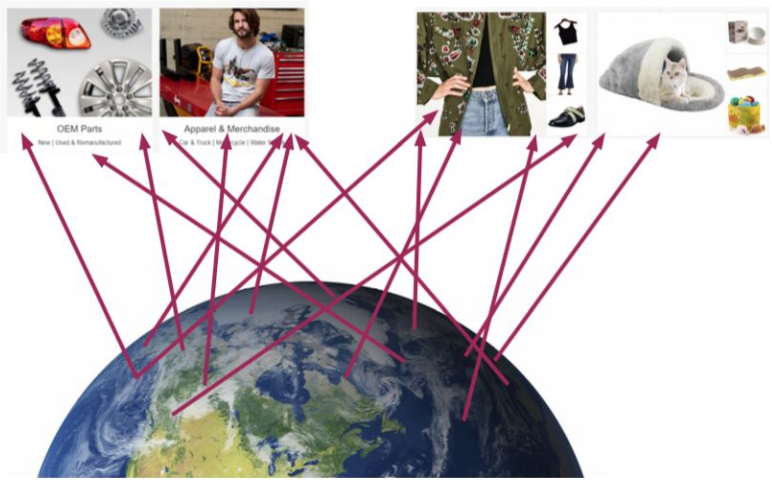
Hive QL

```
SELECT region_city,
    SUM(IF(user.gender = "M",1,0)) as male_hit_count,
    SUM(IF(user.gender = "F",1,0)) as female_hit_count
FROM access_log
    JOIN geo_base ON (access_log.host = geo_base.host)
    JOIN user ON (access_log.host = user.host
        AND access_log.user_agent = user.user_agent
    )
GROUP BY region_city
```

two joins

BIGDATA TEAM

4. Средний возраст клиента



	age	gender	occupation	zipcode
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	05201
9	29	M	student	01002
10	53	M	lawyer	90703

Web-service access logs

Geobase

User personal data

BIGDATA TEAM

4. Средний возраст клиента

Web-service access logs

Geobase

User personal data

format: ip, request, user_agent, ...

format: user_id, gender, age, ...

format: ip, $region_{city}$, $region_{country}$, ...

Join(ip) + Join(user_id) + Average(age)

user_id = ip + user_agent

format: region, average_age

BIGDATA TEAM

4. Средний возраст клиента

Web-service access logs

Geobase

User personal data

format: ip, request, user_agent, ...

format: user_id, gender, age, ...

format: ip, $region_{city}$, $region_{country}$, ...

Join(ip) + Join(user_id) + Average(age)

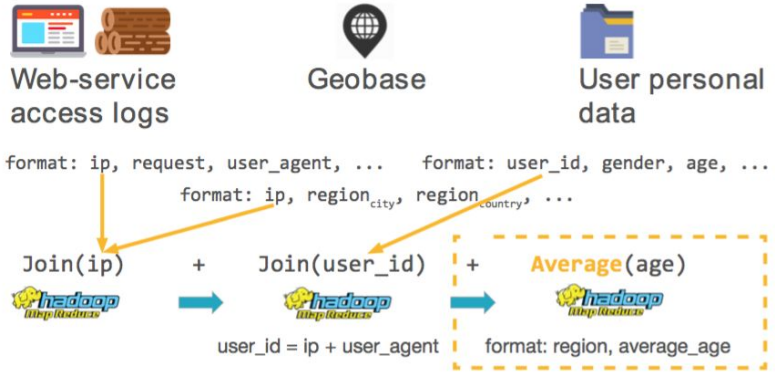
user_id = ip + user_agent

format: region, average_age

use Combiner for optimisation

BIGDATA TEAM

4. Средний возраст клиента



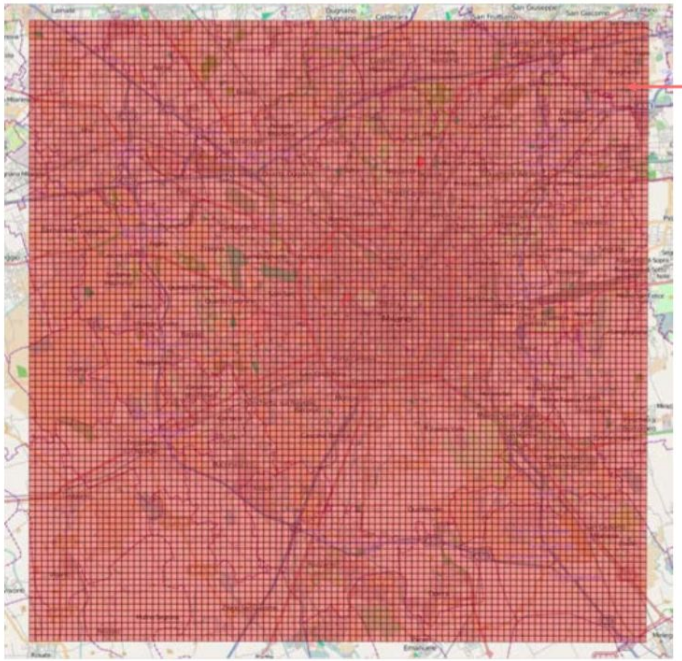
GenericUDAFAverage
Hive Java source code:
https://goo.gl/DCeVPk

Hive QL

```
SELECT region2, AVG(user.age)
FROM access_log
    JOIN geo_base ON (access_log.host = geo_base.host)
    JOIN user ON (access_log.host = user.host
        AND access_log.user_agent = user.user_agent
    )
GROUP BY region2
```

Map- Side Join

Telecommunications Dataset



Milano Grid

- Square ID
- Time Interval
- Country Code
- SMS-in Activity
- SMS-out Activity
- Call-in Activity
- Call-out Activity
- Internet Traffic Activity

Schema

https://dandelion.eu/datagems/SpazioDati/telecom-sms-call-internet-mi

BIGDATA TEAM

Telecommunications Dataset

BIG

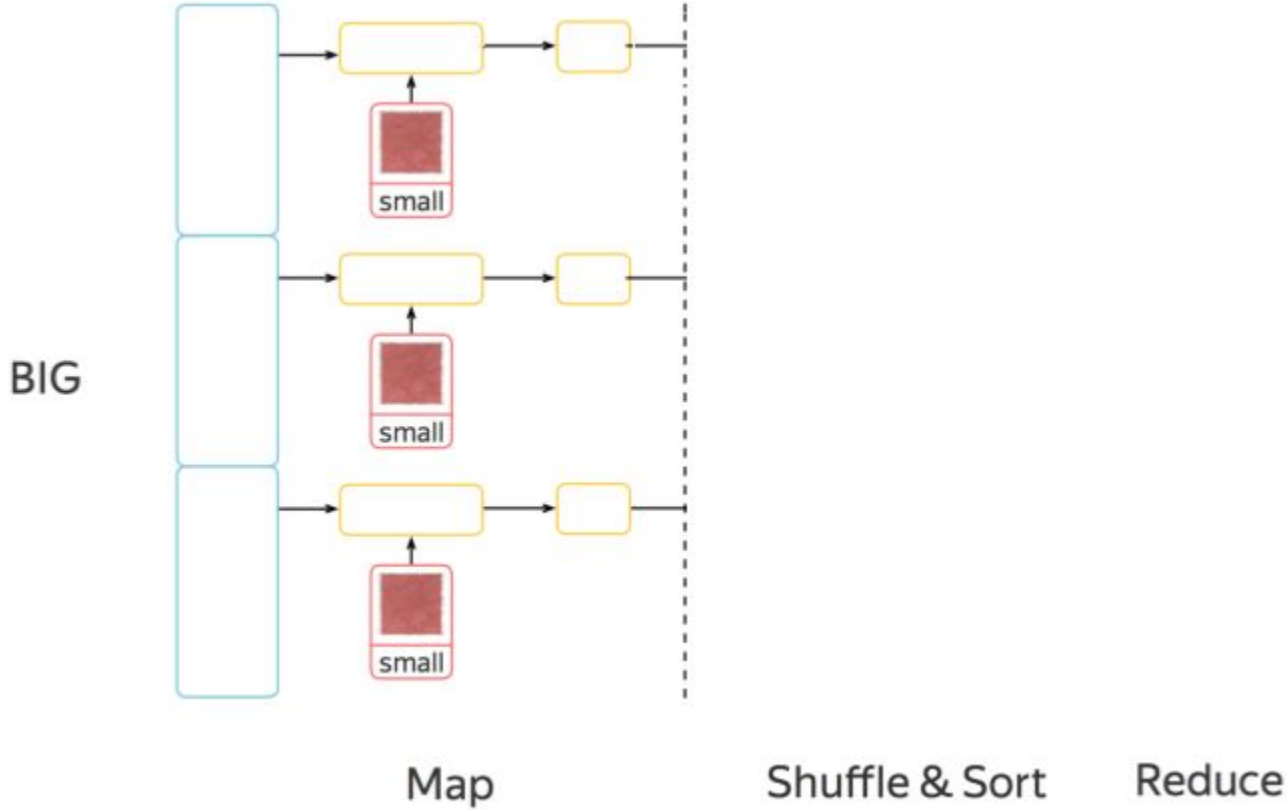
- Square ID
- Time Interval
- Country Code
- SMS-in Activity
- SMS-out Activity
- Call-in Activity
- Call-out Activity
- Internet Traffic Activity

```
1 1383260400000 0 0.08136262351125882
1 1383260400000 39 0.14186425470242922
0.1567870050390246 0.16093793691701822
0.052274848528573205 11.028366381681026
1 1383261000000 0 0.13658782275823106
0.02730046487718618
1 1383261000000 33
0.026137424264286602
...
```

small

```
{'type': 'Polygon', 'coordinates':
[[[9.0114910478323, 45.35880131440966],
[9.014491488013135, 45.35880097314403],
[9.0144909480813, 45.35668565341486],
[9.011490619692509,
45.356685994655464], [9.0114910478323,
45.35880131440966]]]}
...
```

Map- Side Join



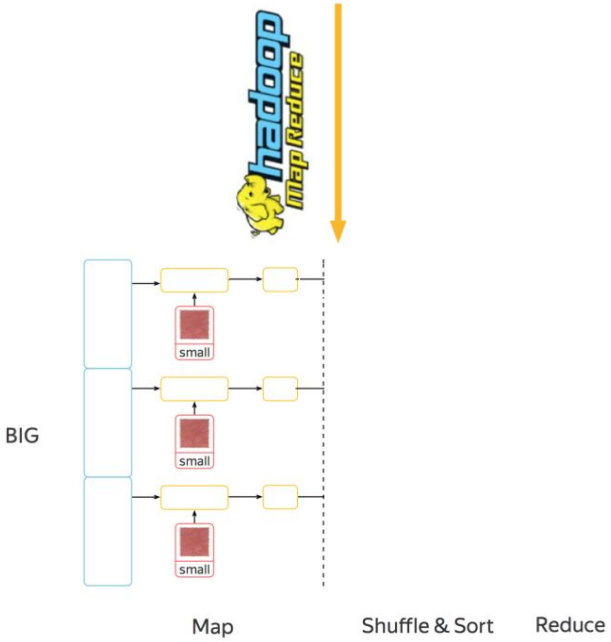
Map- Side Join


```
yarn jar $HADOOP_STREAMING_JAR \
        -files map_side_mapper.py,hdfs:///user/adral/milano-grid.geojson \
        -mapper 'python map_side_mapper.py' \
        -numReduceTasks 0 \
        -input /data/telecommunication \
        -output telecom-joins
```

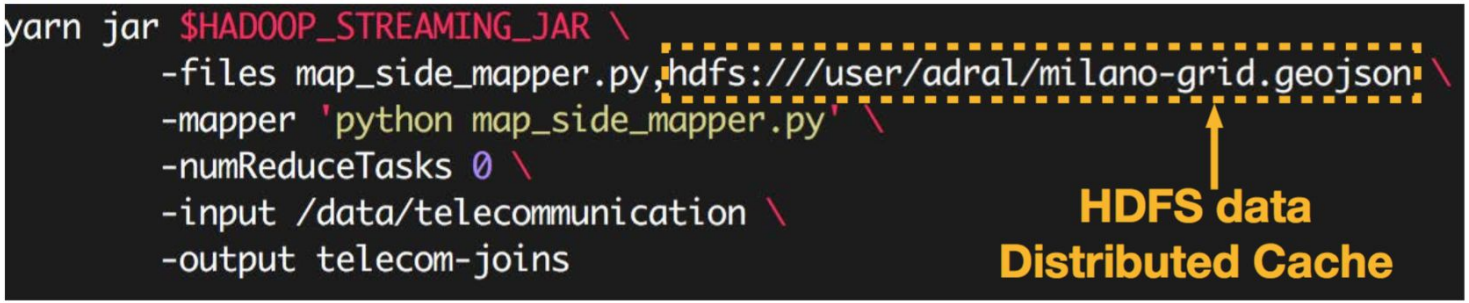
HDFS data
Distributed Cache

Hive Map- Side Join

```
SELECT region_city, COUNT(1) AS hit_count
FROM access_log JOIN geo_base
ON (access_log.host = geo_base.host)
GROUP BY region_city ORDER BY hit_count LIMIT 100
```



Hive Map- Side Join



1. Клиент: загружает датасет из HDFS
2. Строит hashtable
3. Загружает hashtable в Distributed Cache

Tea / Coffee Break

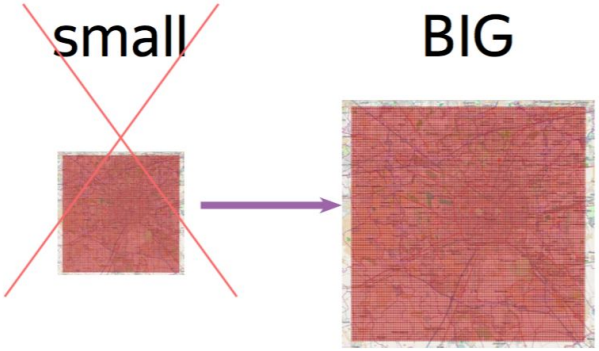
Reduce- Side Join

BIGDATA TEAM

Увеличение покрытия

BIG

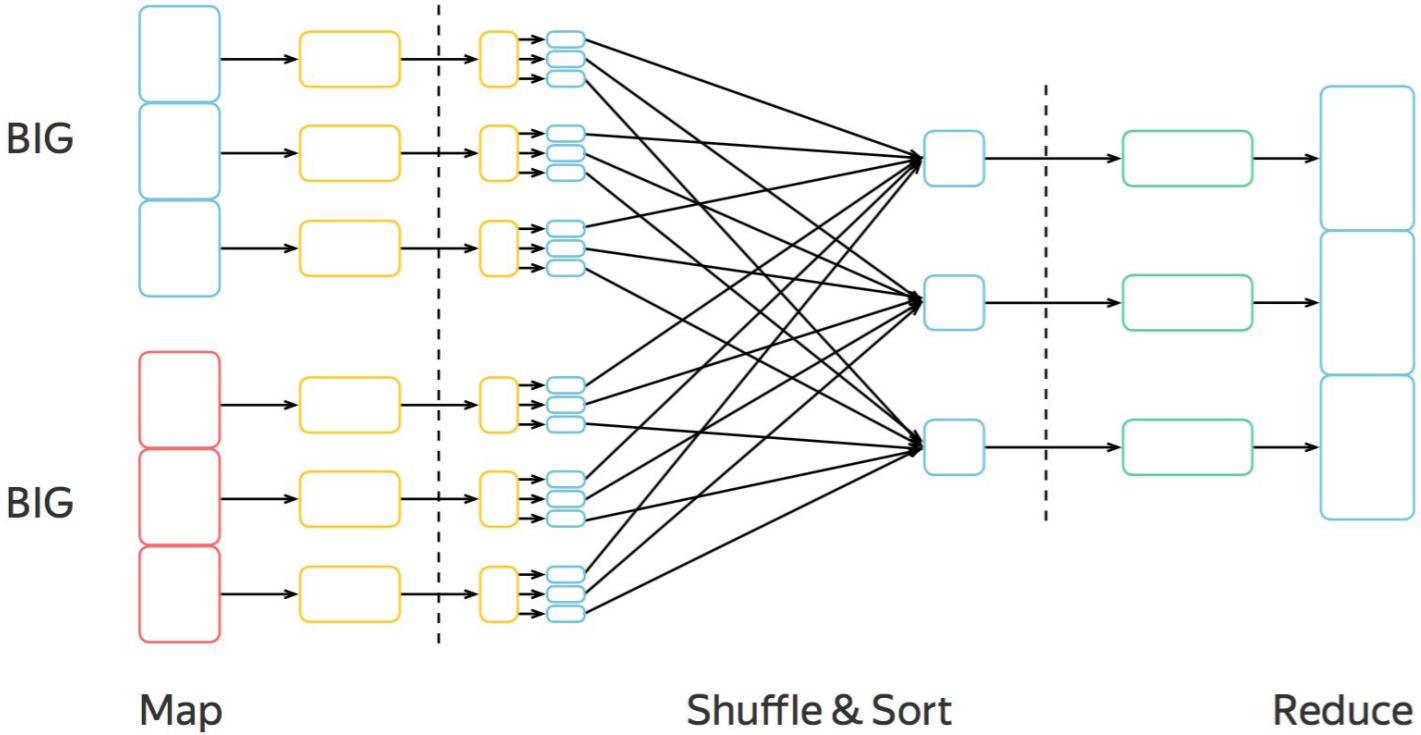
- Square ID
- Time Interval
- Country Code
- SMS-in Activity
- SMS-out Activity
- Call-in Activity
- Call-out Activity
- Internet Traffic Activity




```
1 1383260400000 0 0.08136262351125882
1 1383260400000 39 0.14186425470242922
0.1567870050390246 0.16093793691701822
0.052274848528573205 11.028366381681026
1 1383261000000 0 0.13658782275823106
0.02730046487718618
1 1383261000000 33
0.026137424264286602
...
```

```
{'type': 'Polygon', 'coordinates':
[[[9.0114910478323, 45.35880131440966],
[9.014491488013135, 45.35880097314403],
[9.0144909480813, 45.35668565341486],
[9.011490619692509,
45.356685994655464], [9.0114910478323,
45.35880131440966]]]}
...
```

Reduce- Side Join



Tagging

```
if "geojson" in os.environ["mapreduce_map_input_file"]:
    geojson = json.load(sys.stdin)
    grid = load_grid(geojson)
    for grid_id, ce11_type in grid.items():
        print(grid_id, "grid", ce11_type, sep="\t")
else
    for line in sys.stdin:
        square_id, aggregate = line.split("\t", 1)
        square_id = int(square_id)
        time_interval, country, sms_in, sms_out, call_in, call_out, internet = aggregate.split("\t"
        if sms_in:
            sms_in = float(sms_in)
            print(square_id, "logs", sms_in, sep="\t")
```

Пример запуска

```
yarn jar $HADOOP_STREAMING_JAR \
        -files reduce_side_mapper.py \
        -mapper 'python reduce_side_mapper.py' \
        -numReduceTasks 0 \
        -input /data/telecommunication,/user/adral/geojson \
        -output telecom-joins
```

```
$ hdfs dfs -text telecom-joins/part-00010 | head -3
1 grid South
2 grid South
3 grid South
```

string

```
$ hdfs dfs -text telecom-joins/part-00000 | head -3
1 logs 0.0813626235113
1 logs 0.141864254702
1 logs 0.136587822758
```

numeric

BIGDATA TEAM

Добавляем Identity Reducer

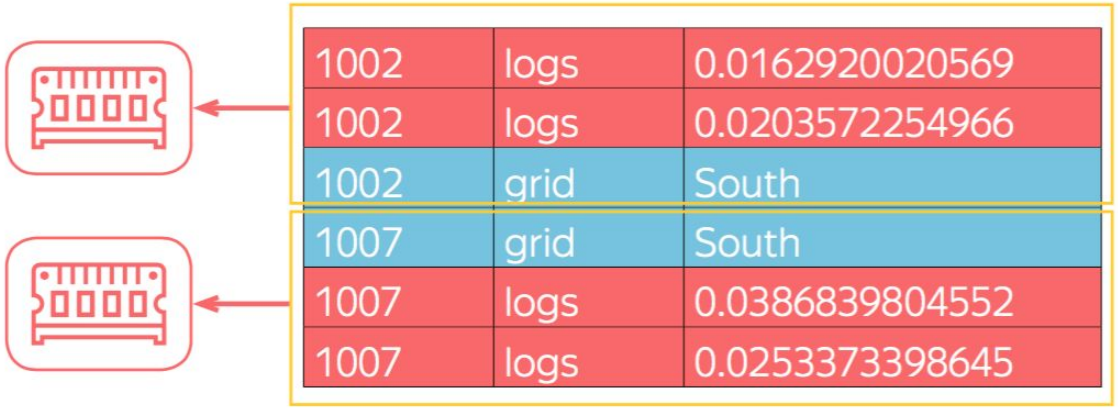
```
yarn jar $HADOOP_STREAMING_JAR \
    -D mapreduce.partition.keypartitioner.options="-k1,1"\
    -files reduce_side_mapper_slice.py \
    -mapper 'python reduce_side_mapper.py' \
    -numReduceTasks 5 \
    -input /data/telecommunication,/user/adral/geojson \
    -output telecom-joins \
    -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

1002	logs	0.0162920020569
1002	logs	0.0203572254966
1002	grid	South
1007	grid	South
1007	logs	0.0386839804552
1007	logs	0.0253373398645

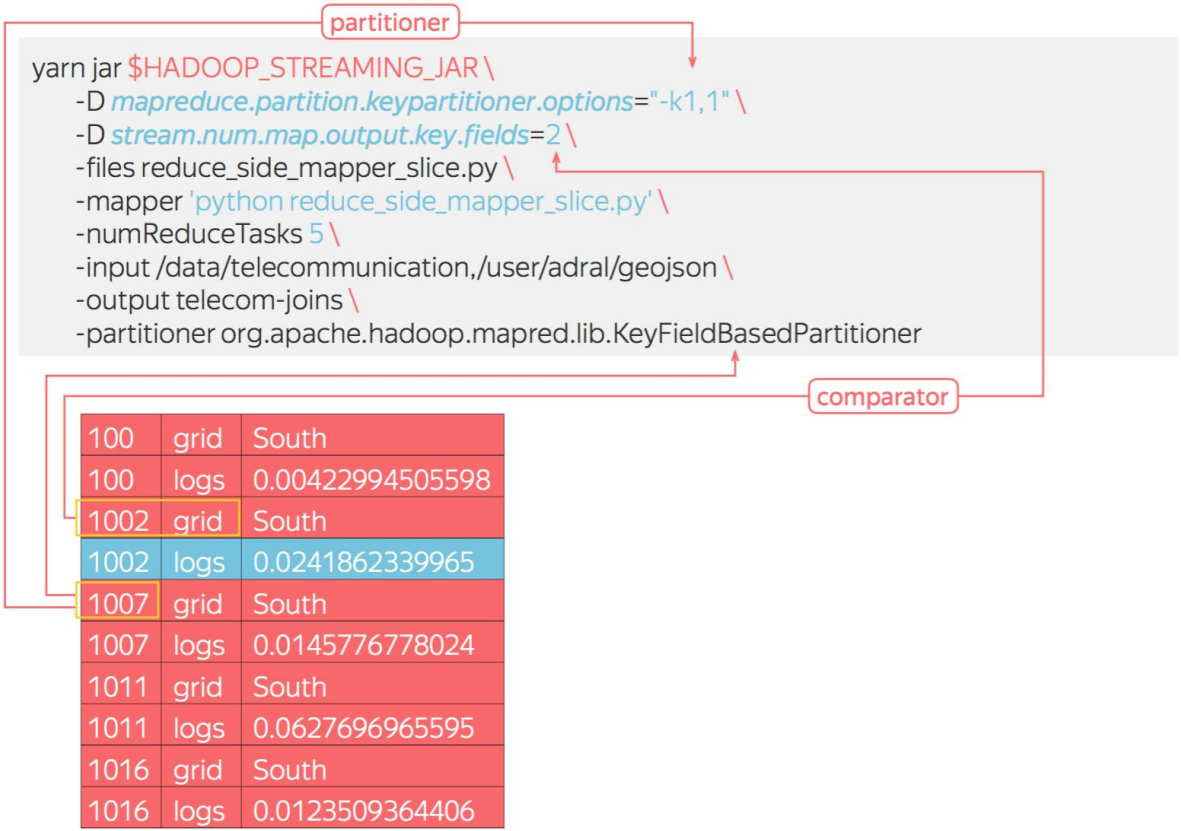
BIGDATA TEAM

Ограничения по памяти

```
yarn jar $HADOOP_STREAMING_JAR \
    -D mapreduce.partition.keypartitioner.options="-k1,1"\
    -files reduce_side_mapper_slice.py \
    -mapper 'python reduce_side_mapper.py' \
    -numReduceTasks 5 \
    -input /data/telecommunication,/user/adral/geojson \
    -output telecom-joins \
    -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```



Secondary Sort (via Comparator)



Secondary Sort: Comparator Flags

```
yarn jar $HADOOP_STREAMING_JAR \
        -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.KeyFieldBasedComparator \
        -D mapreduce.partition.keycomparator.options="-k1,2r" \
        -D mapreduce.partition.keypartitioner.options="-k1,1" \
        -D stream.num.map.output.key.fields=2 \
        -files reduce_side_mapper_slice.py \
        -mapper 'python reduce_side_mapper_slice.py' \
        -numReduceTasks 5 \
        -input /data/telecommunication,/user/adral/geojson \
        -output telecom-joins \
        -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

9996	logs	0.149333295147
9996	grid	North
9991	logs	0.330465627227
9991	grid	North
9987	logs	0.0296826530265
9987	grid	North
9982	logs	0.262932749854
9982	grid	North
998	logs	0.0881801546604
998	grid	South

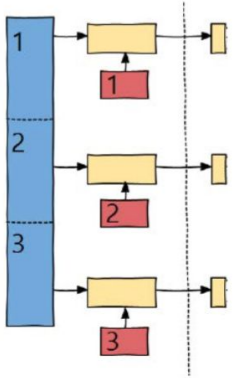
Join Options

Bucket Map- Side Join

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE geo_base (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join



Map Phase

BIGDATA TEAM

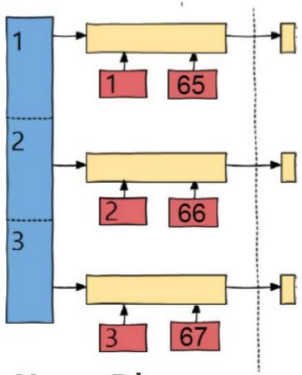
Bucket Map- Side Join

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    INTO 64 BUCKETS
…;
```

```
CREATE TABLE geo_base (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join

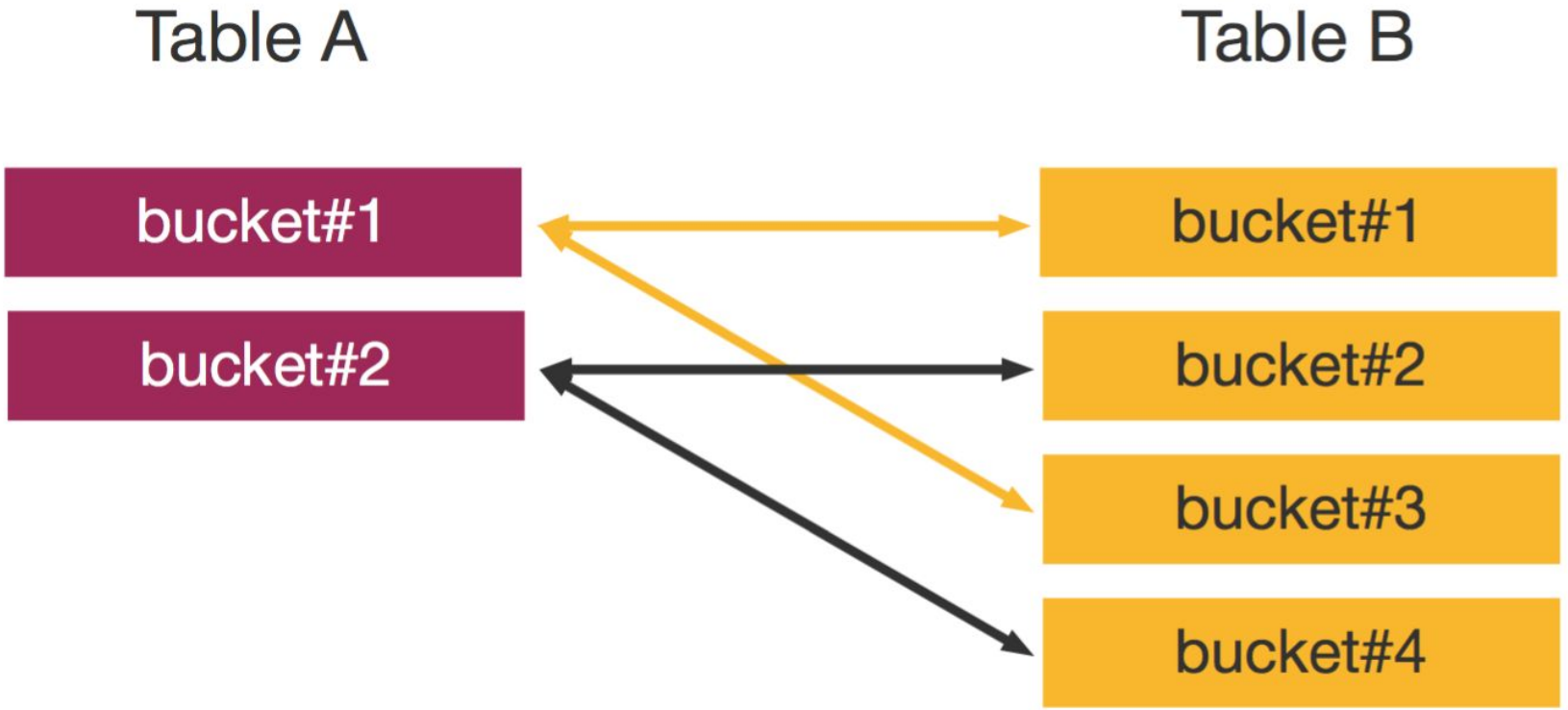
hash(ip) % 64 == 1



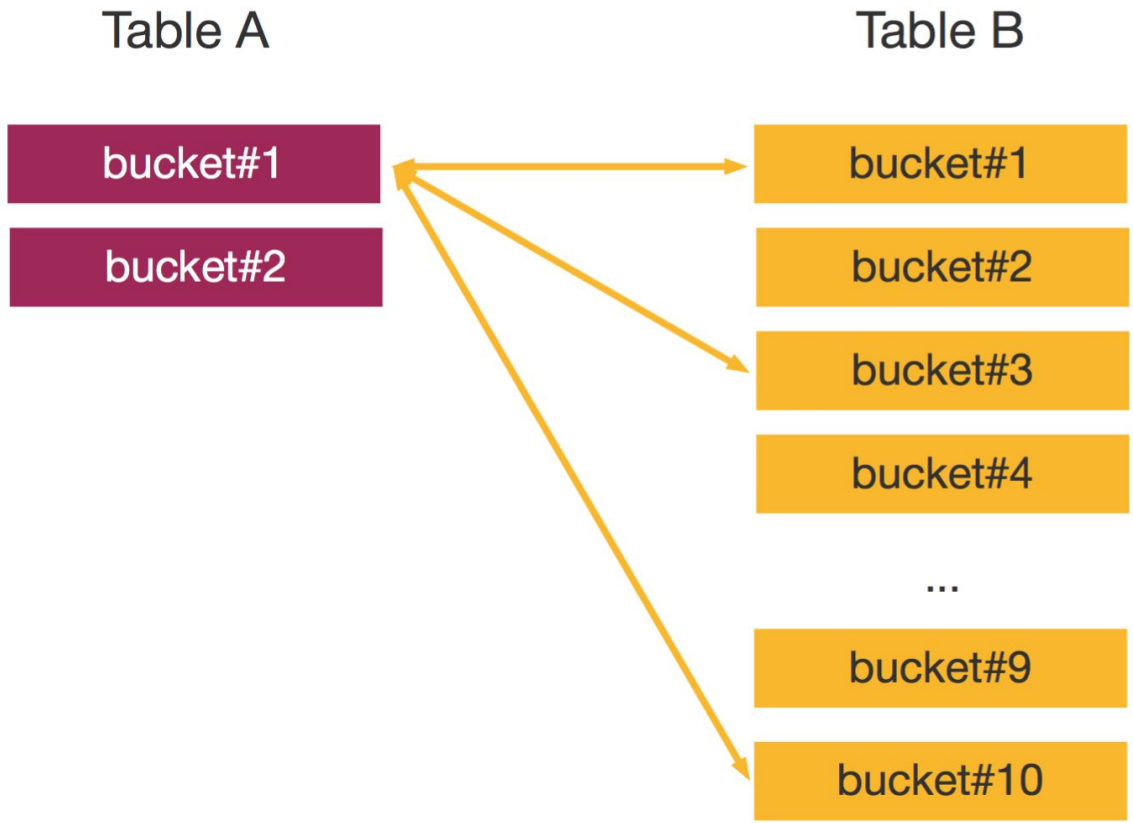
hash(ip) % 128 == 1,65

Map Phase

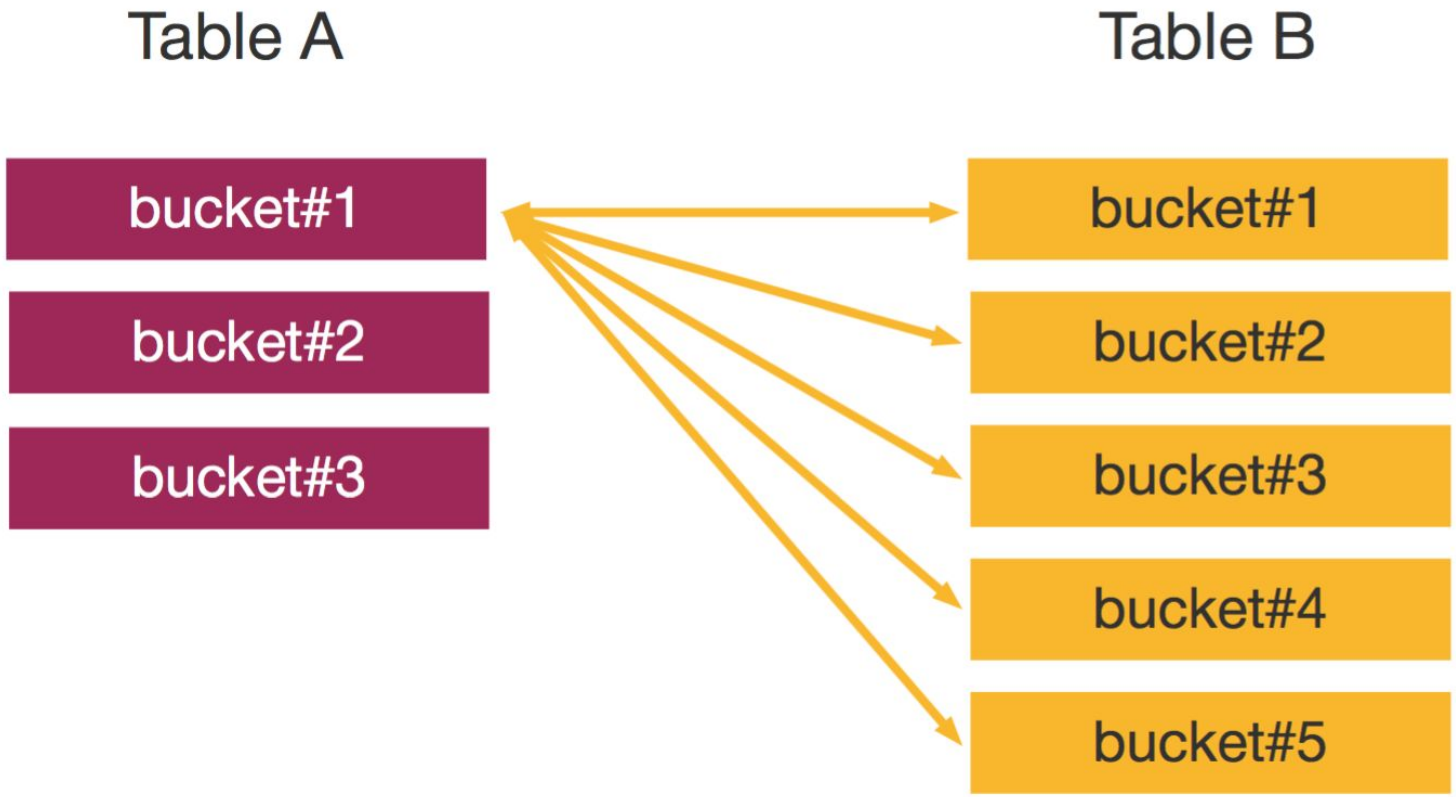
Bucket Map- Side Join



Bucket Map- Side Join

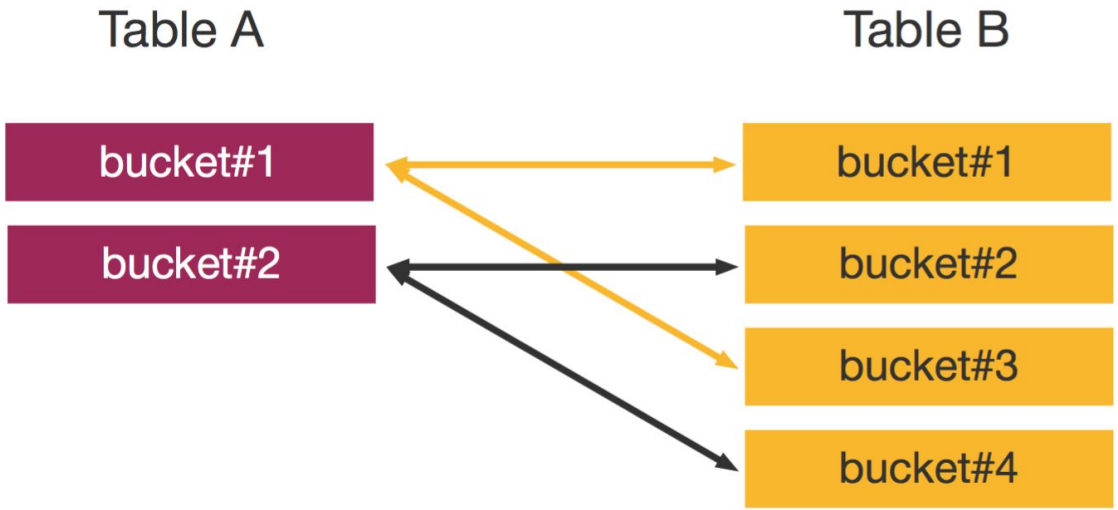


Bucket Map- Side Join



Bucket Map- Side Join

совет: использовать степени двойки



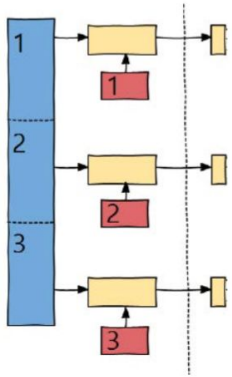
1. Если повезет, запустится Bucket Map-Side Join
2. Наиболее эффективно использует Namenode RAM

BIGDATA TEAM

Оптимизация Bucket Map- Side Join

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join limitations?



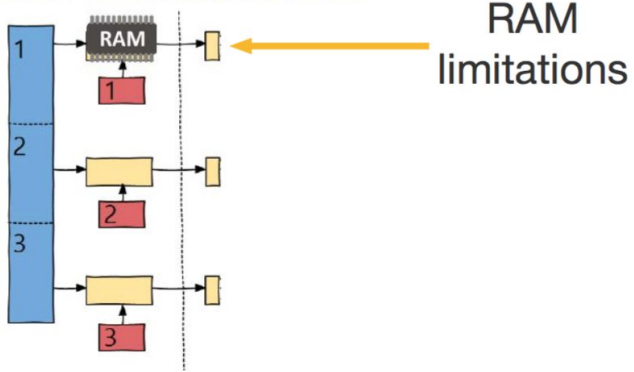
Map Phase

Bucket Map- Side Join Optimization

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

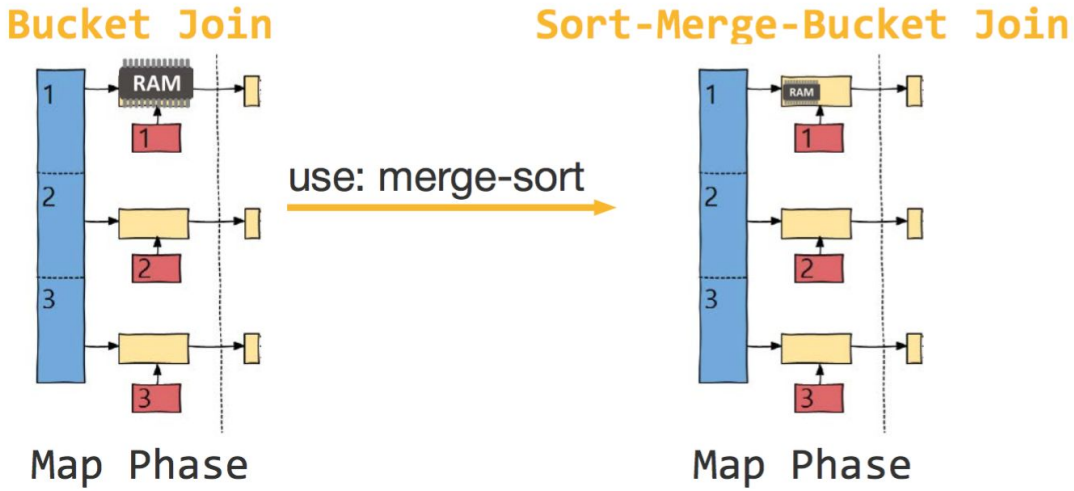
see: https://en.wikipedia.org/wiki/Merge_sort
see: https://en.wikipedia.org/wiki/External_sorting

Bucket Join limitations?

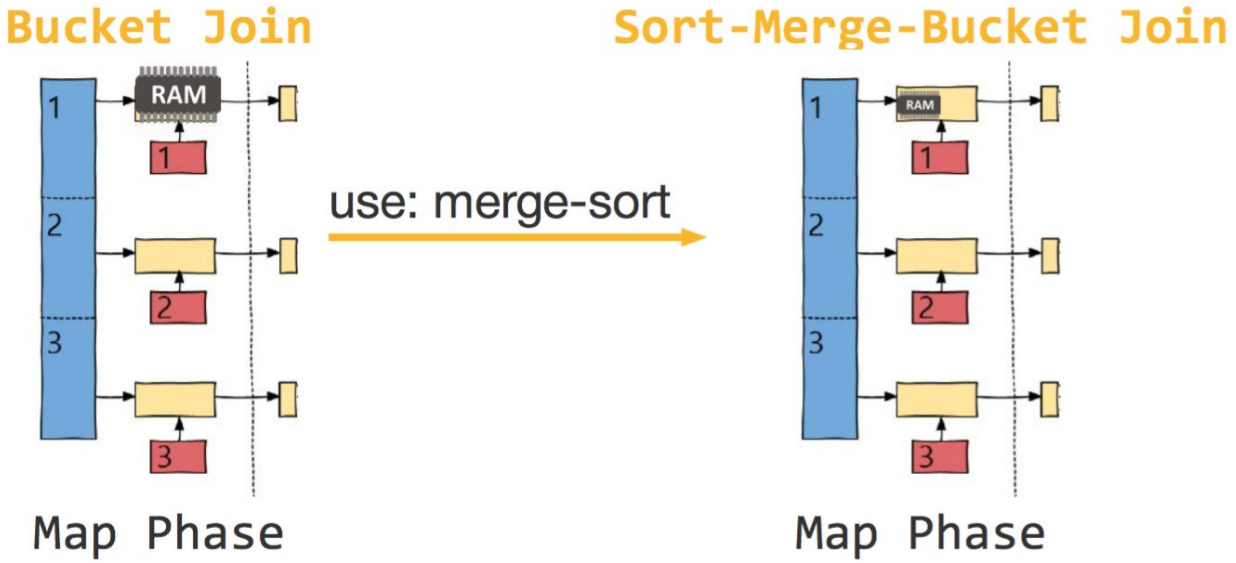


Sort-Merge-Bucket Join

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```



Sort-Merge-Bucket Join




```
hive> SET hive.auto.convert.sortmerge.join=true;
…
```

External / Managed таблицы в Hive + RegexpSerDe


KEEP
CALM
AND
TRY
CODING

Резюме

- Вы знаете, какие типы Join бывают в Hadoop / Hive (Map-, Reduce-, Bucket- и SMB).
- Вы умеете (и умели еще с прошлого занятия) решать задачу вторичной сортировки (Secondary Sort)
- Вы умеет пользоваться Managed и External таблицами без риска удаления production данных, а также RegexpSerDe для задания произвольной схемы.

Thank you! Questions?

Feedback: http://rebrand.ly/mf2019q2_feedback_04_hive

Dral Alexey, aadral@bigdatateam.org
CEO at BigData Team, http://bigdatateam.org/
https://www.linkedin.com/in/alexey-dral
https://www.facebook.com/bigdatateam/

Appendix

Материалы для погружения

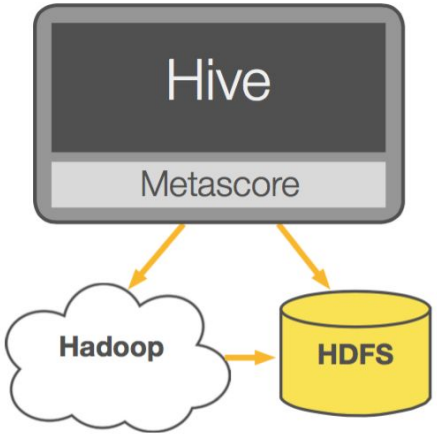
- Семплирование данных в Hive: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Sampling
- Бакетированные данные в Hive: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL+BucketedTables
- UDF, UDAF, UDTF: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF
- Работа с вложенными данными: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+LateralView
- Работа с View: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/AlterView

Материалы для погружения

- Настраиваем Hive таблицы поверх данных в HDFS: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL
- Пользуемся клаузой “EXPLAIN” для того, чтобы понять план выполнения Hive-запроса: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Explain

Hive DML

Hive DML: Implementation **Q&A**

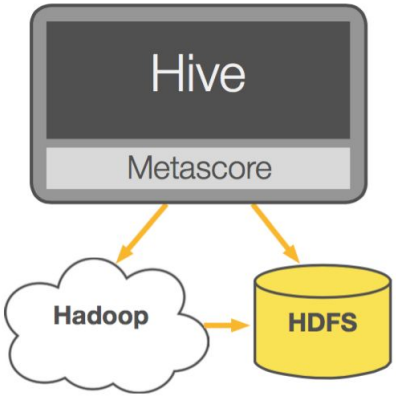


(1) DDL
(2) HiveQL (details)
(3) DML

MapReduce (?)

- SELECT .. FROM
- WHERE
- GROUP BY + HAVING
- JOIN
- ORDER BY / **SORT BY**

Hive DML: Implementation **Q&A**



(1) DDL
(2) HiveQL (details)
(3) DML

MapReduce (?)

- SELECT .. FROM **[<-- Map]**
- WHERE **[<-- Map]**
- GROUP BY **[<-- Shuffle & Sort]** + HAVING **[<-- Reduce]**
- JOIN **[<-- Map / Reduce "-side"]**
- ORDER BY / **SORT BY [<-- Reduce]**


Hive DML: Explain

```
EXPLAIN
FROM src
INSERT OVERWRITE TABLE dest_g1
SELECT src.key, sum(substr(src.value,4))
GROUP BY src.key;
```

(1) The Abstract Syntax Tree

```
ABSTRACT SYNTAX TREE:
  (TOK_QUERY (TOK_FROM (TOK_TABREF src))
 …
```

(2) The Dependency Graph

```
 STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-2 depends on stages: Stage-1
  Stage-0 depends on stages: Stage-2
```

(3) The plans of each Stage

```
STAGE PLANS:
  Stage: Stage-1
    Map Reduce
      Alias -> Map Operator Tree:
        src
            Reduce Output Operator
              key expressions:
                    expr: key
                    type: string
              sort order: +
```