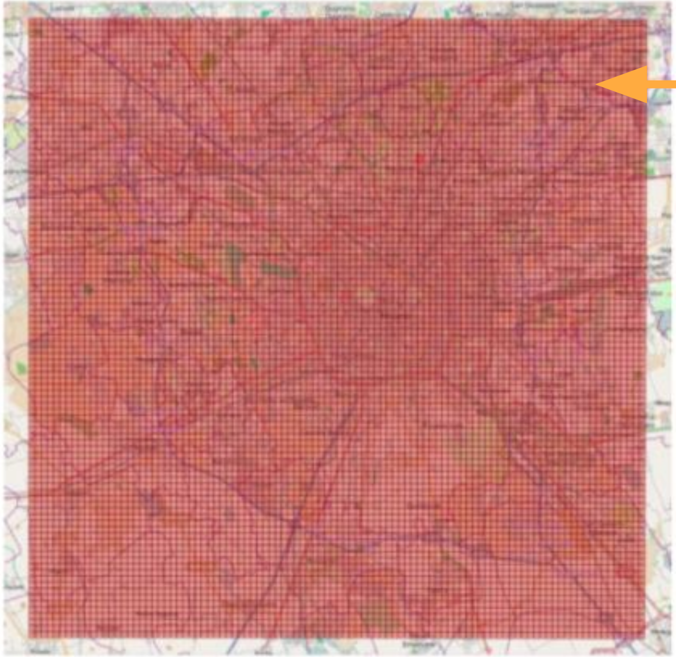
# Как солить косые данные (Data Skew, Salting)

**Драль Алексей**, study@bigdatateam.org
CEO at BigData Team, https://bigdatateam.org
https://www.facebook.com/bigdatateam

# Telecommunications Dataset



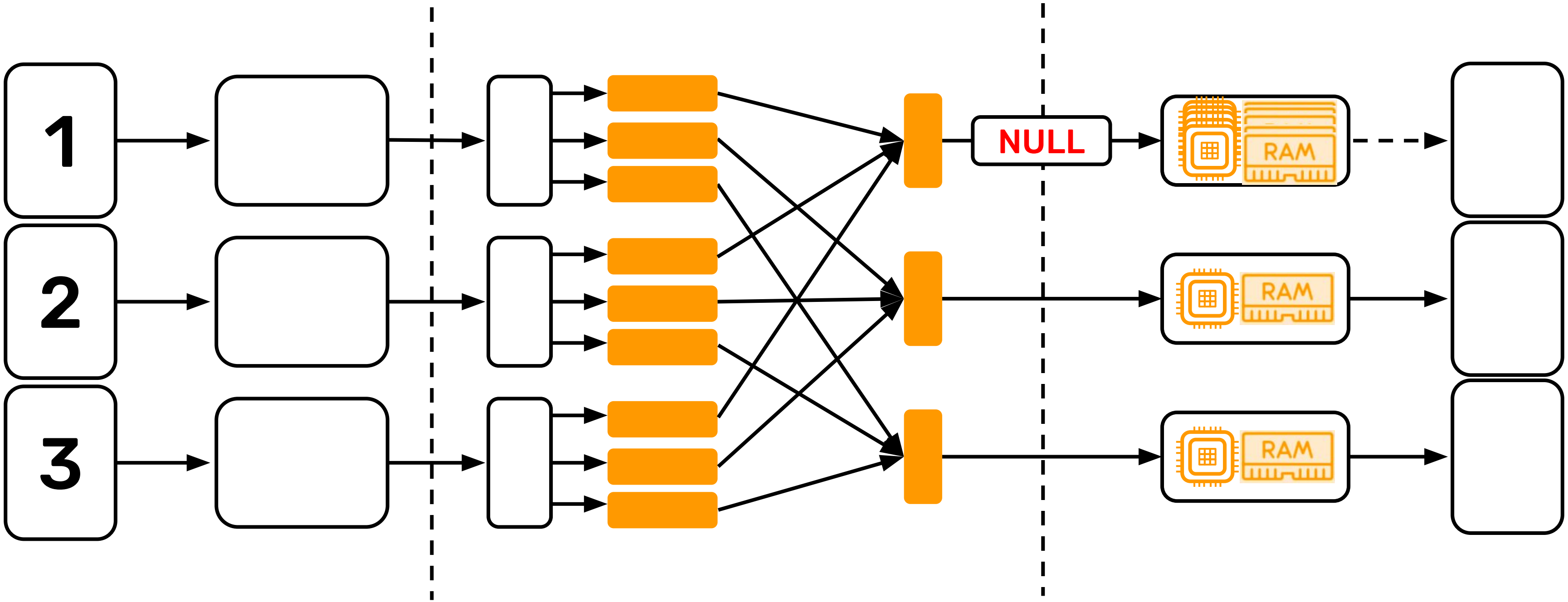
Milano Grid

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

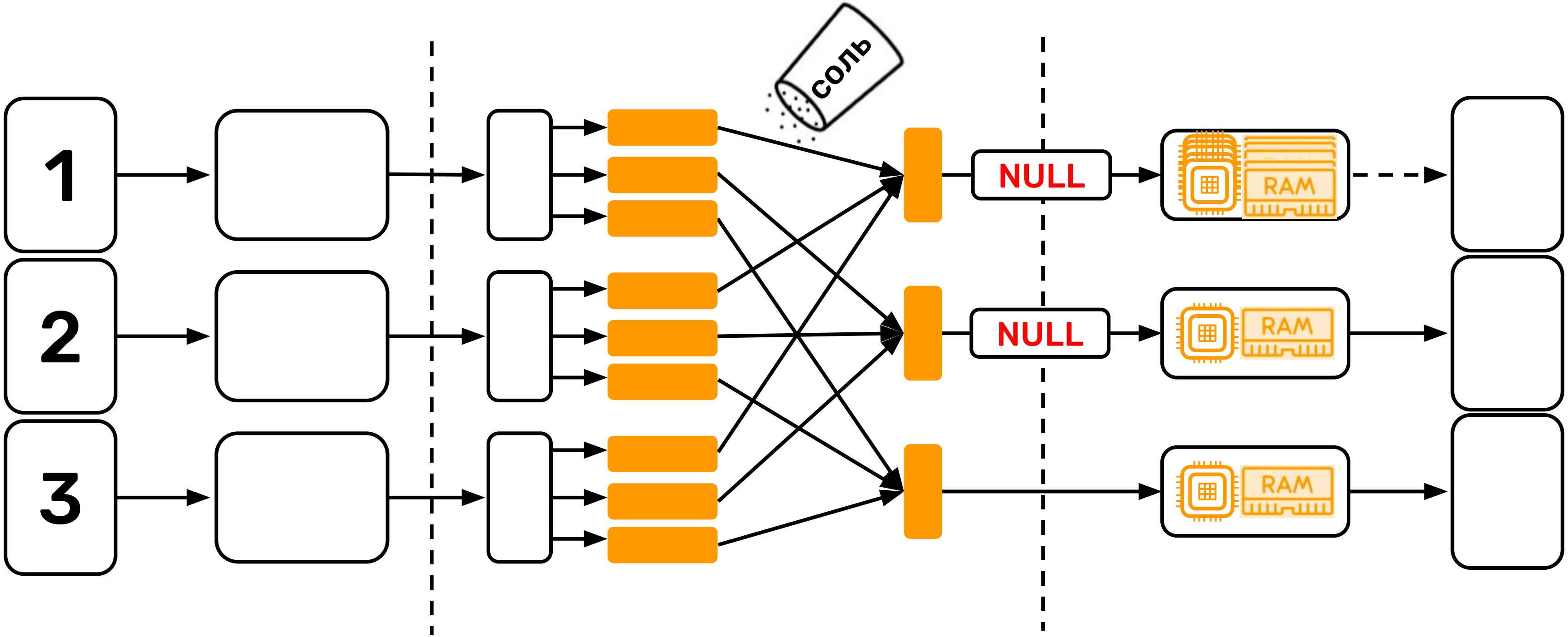
Schema

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

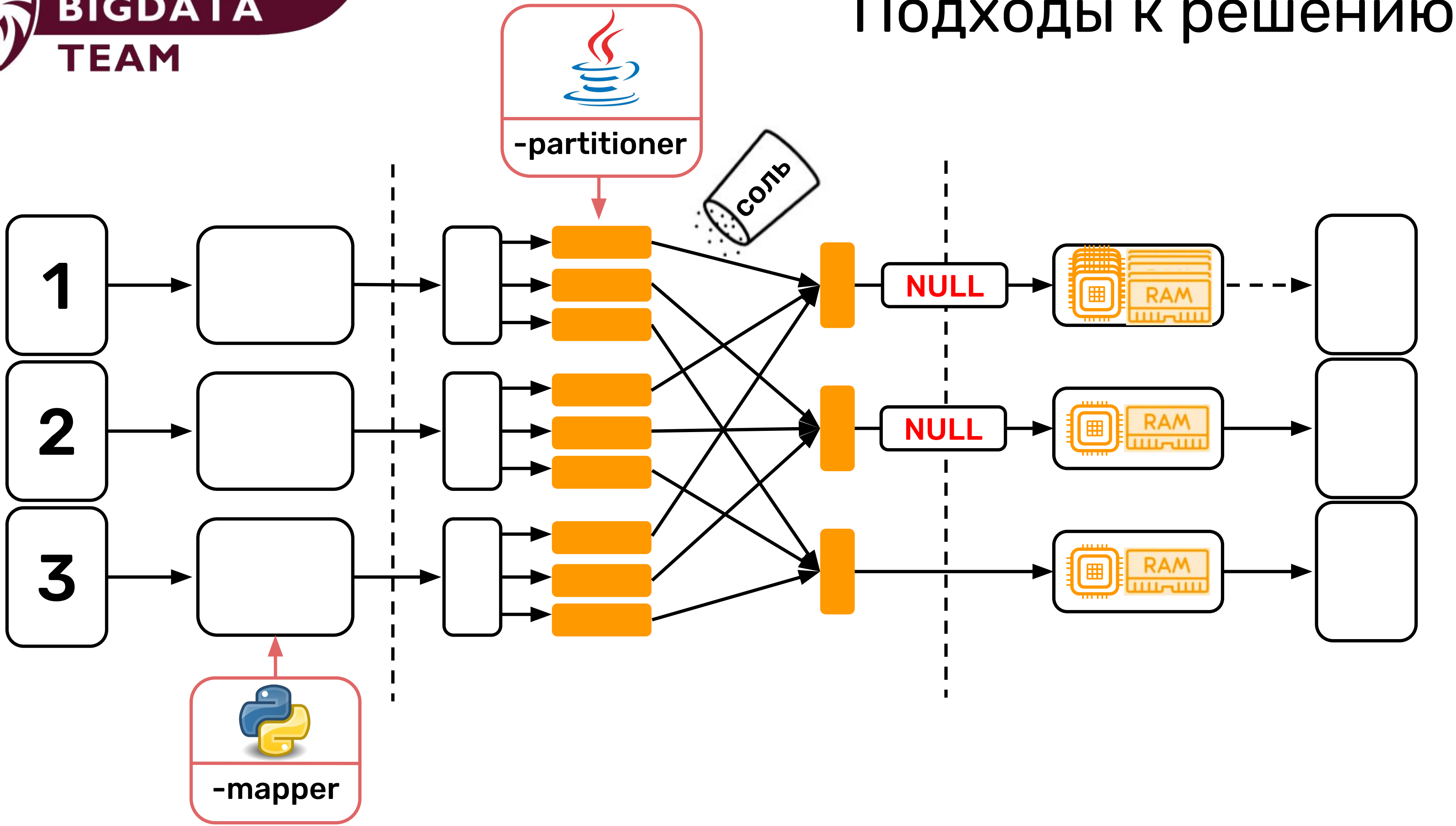
# Dataset Skew (Nulls)



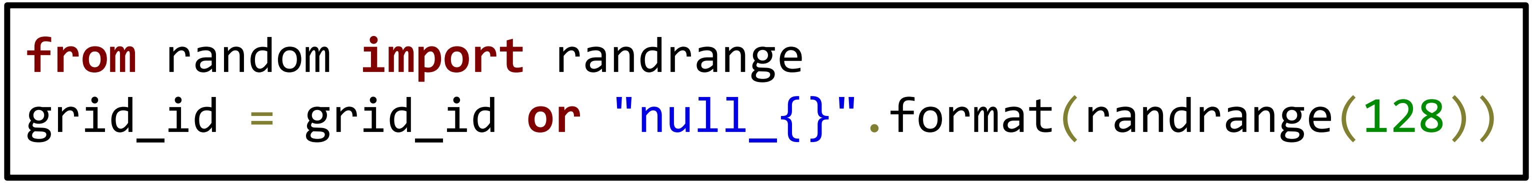
# Соль ~~и перец~~



# Подходы к решению



# Засол (Salting)



```python
from random import randrange
grid_id = grid_id or "null_{}".format(randrange(128))
```

# Пример вывода

```
...
null_58  40989.56529872355
null_67  40775.58025775422
null_76  42430.98650098723
null_85  41811.88806991089
null_94  41086.03092382825
...
```

# Вторая стадия соления

```
...
null_58    40989.56529872355
null_67    40775.58025775422
null_76    42430.98650098723
null_85    41811.88806991089
null_94    41086.03092382825
...
```

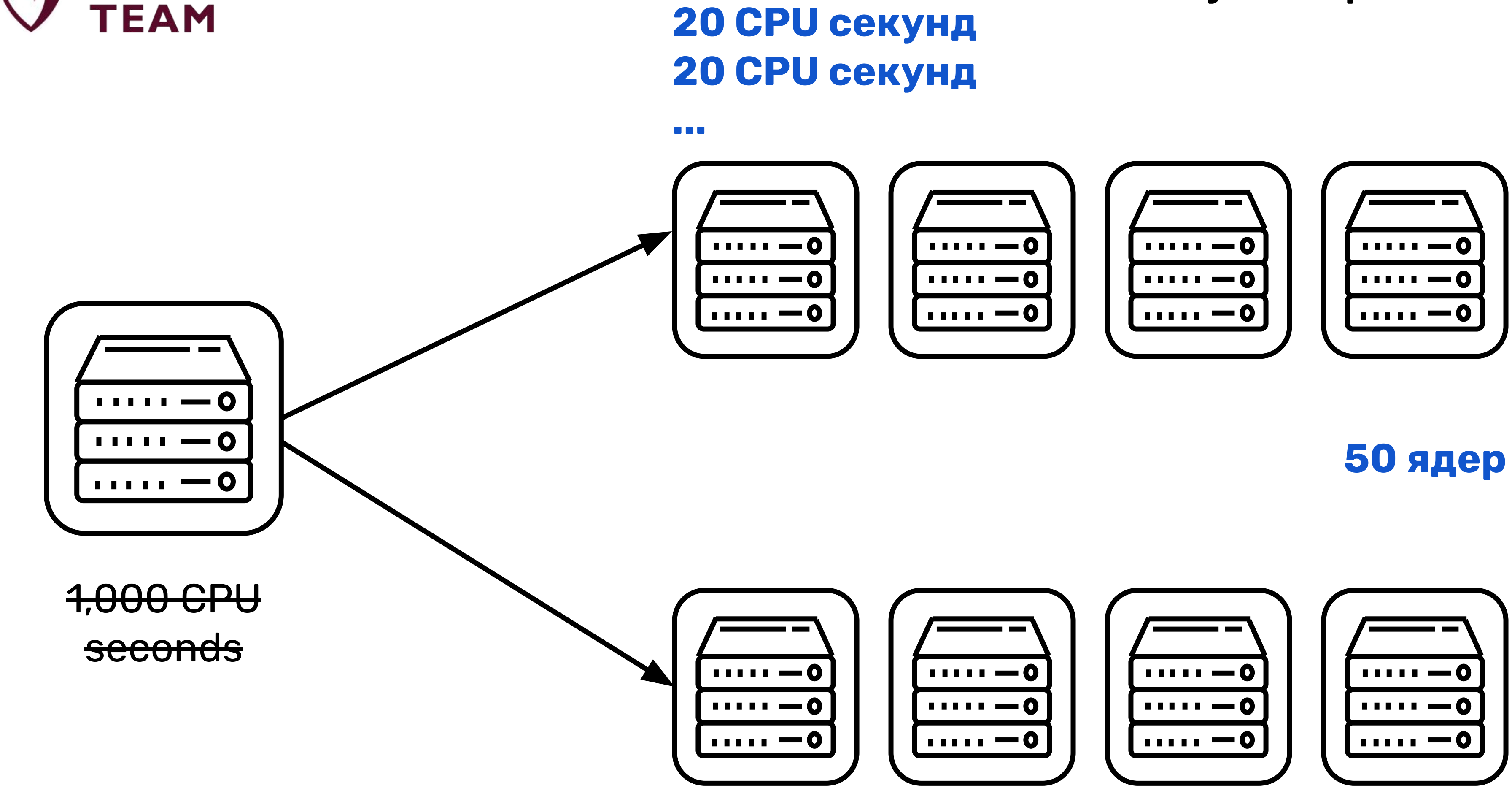
```
for line in sys.stdin:
    key, value = line.rstrip("\n").split("\t", 1)
    key = "null" if key.startswith("null_") else key
    print(key, value, sep="\t")
```

```
South       164302.581973124
null       4145425.004916422
North       296659.744074992
```

# Оценка ускорения

~~1,000 CPU seconds~~

# Оценка ускорения

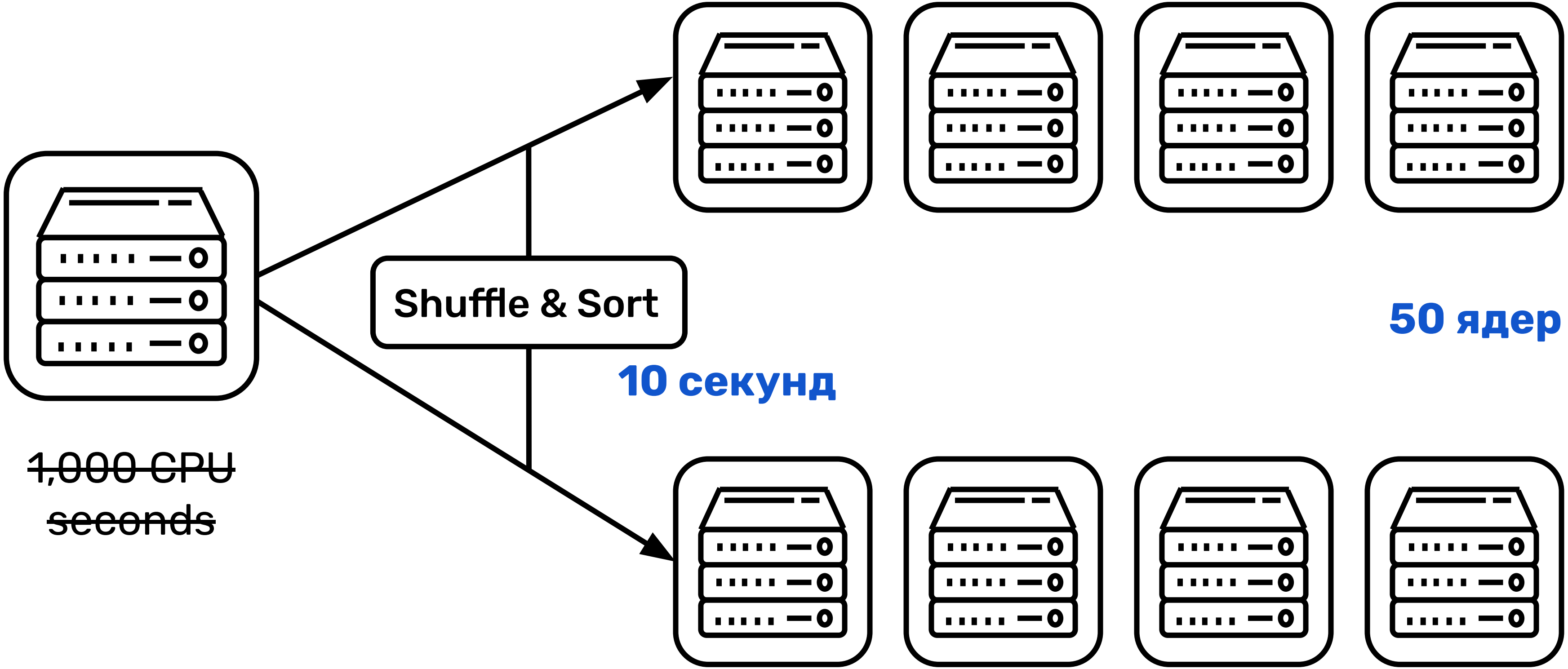


# Оценка ускорения

**20 CPU секунд**

**20 CPU секунд**

**...**



**10 секунд**

**50 ядер**

~~1,000 CPU seconds~~

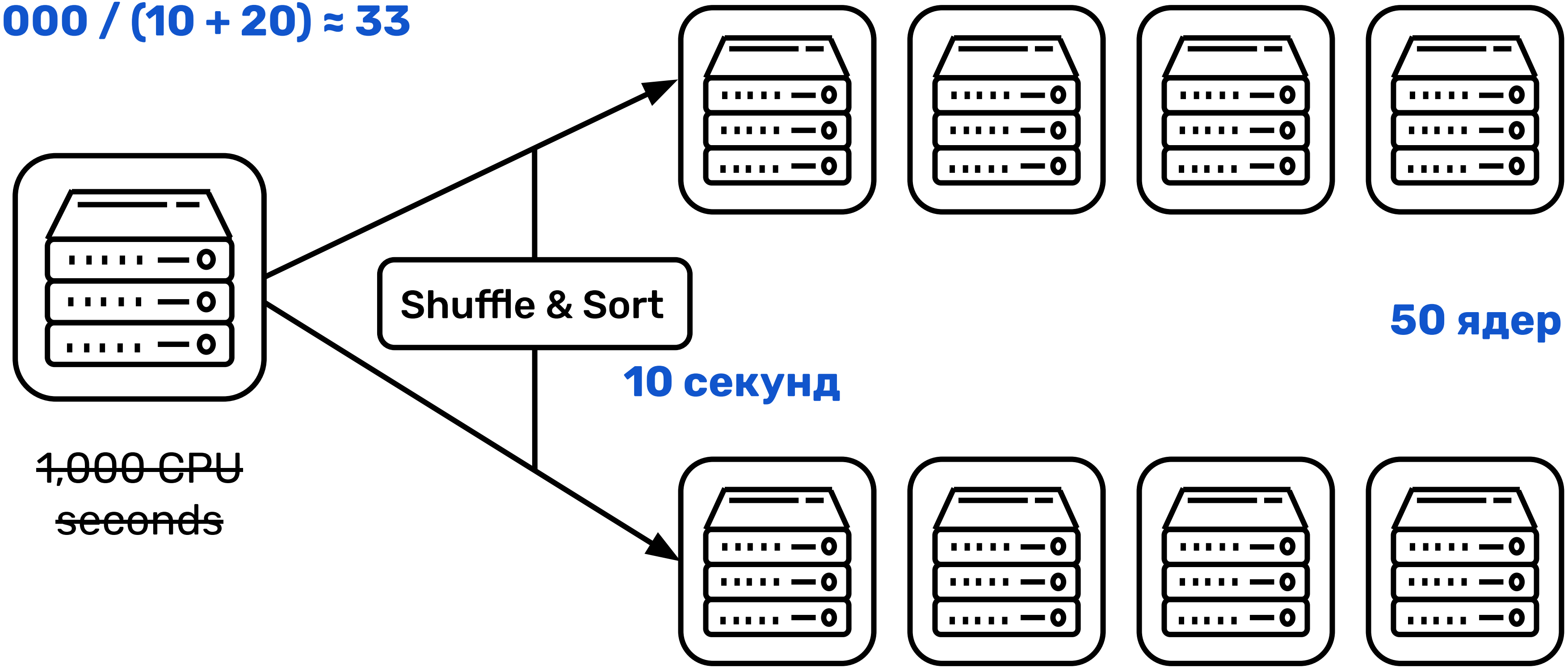
# Оценка ускорения

20 CPU секунд

20 CPU секунд

...

$1000 / (10 + 20) \approx 33$



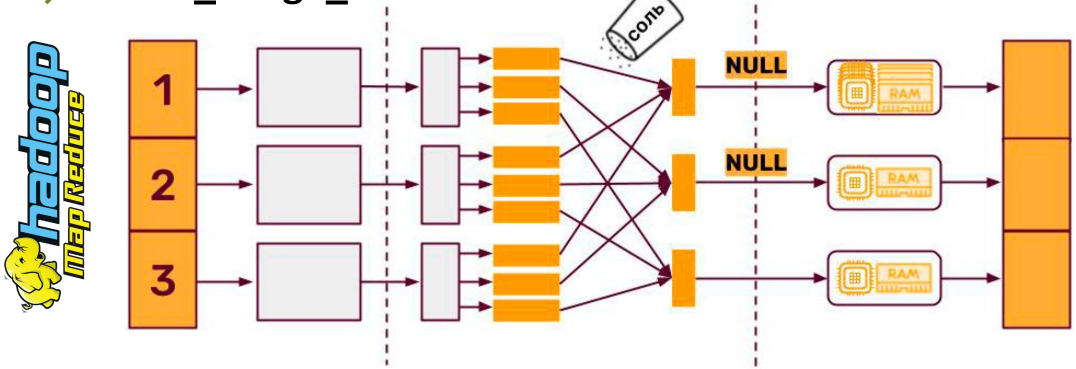
10 секунд

50 ядер

~~1,000 CPU seconds~~

# Data Skew в Hive: версия 1

```
SET mapred.reduce.tasks = 128;
SELECT TRANSFORM(grid_id, ...)
    USING "./count.sh" AS grid_id, some_stat
FROM (
    SELECT *
    FROM access_log
    DISTRIBUTE BY (
      hash(grid_id)
      + IF(grid_id IS NULL, my_salt_UDF(), 0)
    )
) table_stage_0
```



# Data Skew в Hive: версия 1

```
FROM (
    SELECT *
    FROM access_log
    DISTRIBUTE BY (
      hash(user_id)
      + IF(user_id IS NULL, my_salt_UDF(), 0)
    )
) table_stage_0
```

пример

```
SELECT CONCAT("none-", SUBSTR(
       reflect("java.util.UUID", "randomUUID"), 0, 8))
FROM some_table ...;
```

```
...
none-0a1a15ac
none-29e78368
none-3daa8e36
...
```

# Skewed Table DDL

```
CREATE TABLE skewed_access_log (
    ip STRING,
    ...
    ~~request_date STRING~~,
    user_id STRING,
    ...
)
PARTITIONED BY (request_date STRING)
SKEWED BY (user_id) ON ("unknown", "1")
...
```

# List Bucketing

```
CREATE TABLE skewed_access_log (
    ip STRING,
    ...
    user_id STRING,
    ...
)
SKEWED BY (user_id) ON ("unknown", "1")
    STORED AS DIRECTORIES
...
```

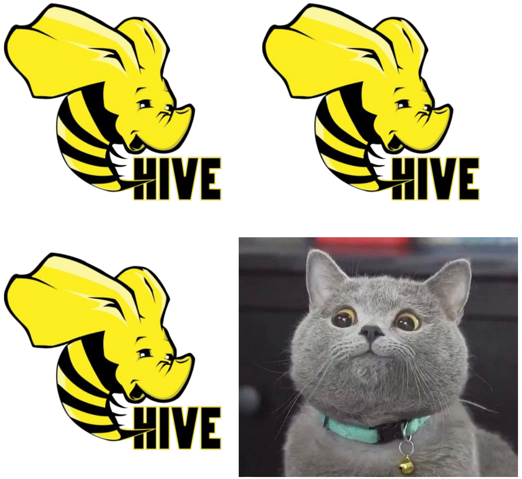
# List Bucketing

```
CREATE TABLE skewed_access_log (
    ip STRING,
    ...
    user_id STRING,
    ...
)
SKEWED BY (user_id) ON ("unknown", "1")
    STORED AS DIRECTORIES
...
```

```
hdfs:///path/to/skewed_access_logs/
 -  user_id = unknown  ←
 -  user_id = 1        ←
 -  HIVE_DEFAULT_LIST_BUCKETING_DIR_NAME
```

# schema-on-*

## schema-on-read



## schema-on-write

cassandra
SQLite
PostgreSQL
MySQL®

# List Bucketing (on-write)

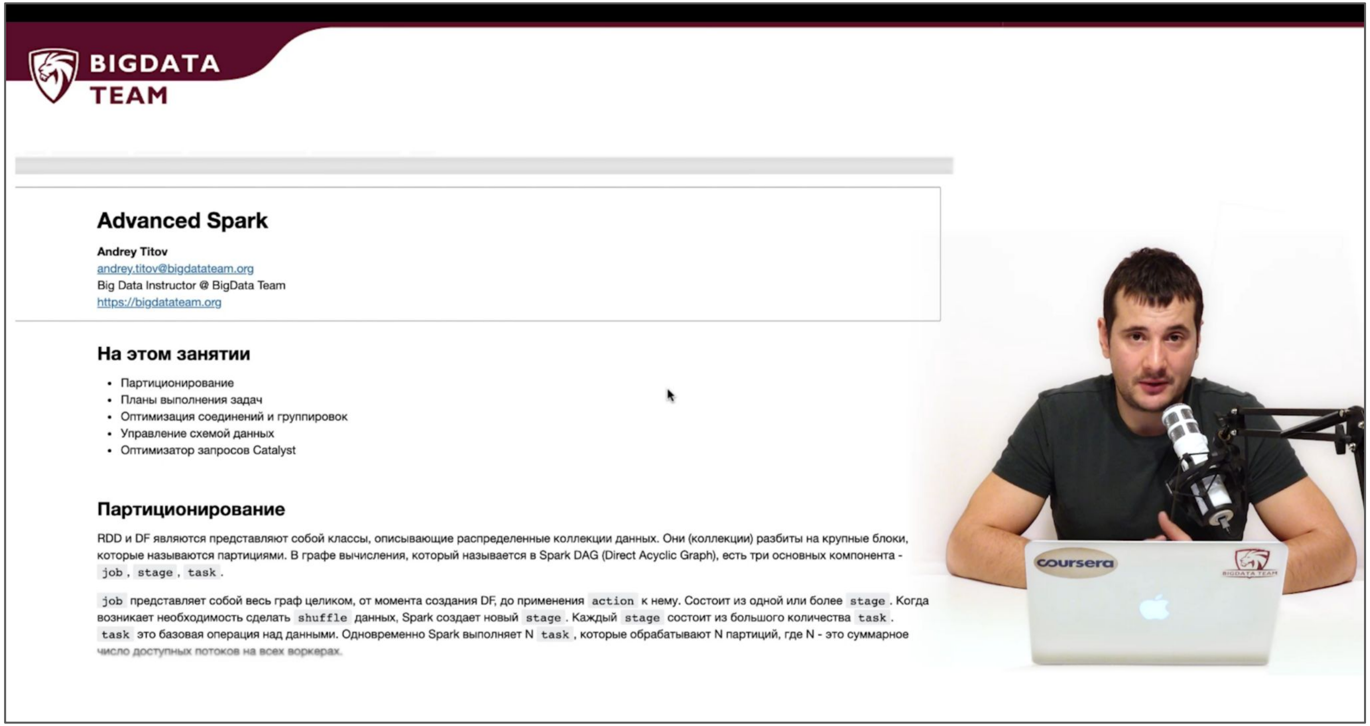
```
SET hive.mapred.supports.subdirectories=true;

INSERT OVERWRITE TABLE skewed_access_log
SELECT ...
FROM apache_log_raw;
```

```
hdfs:///path/to/skewed_access_logs/
   - user_id=unknown
   - user_id=1
   - HIVE_DEFAULT_LIST_BUCKETING_DIR_NAME
```

# А что в Spark?

- v.2 - ручной труд (см. MapReduce)



## Advanced Spark

**Andrey Titov**
andrey.titov@bigdatateam.org
Big Data Instructor @ BigData Team
https://bigdatateam.org

### На этом занятии

- Партиционирование
- Планы выполнения задач
- Оптимизация соединений и группировок
- Управление схемой данных
- Оптимизатор запросов Catalyst

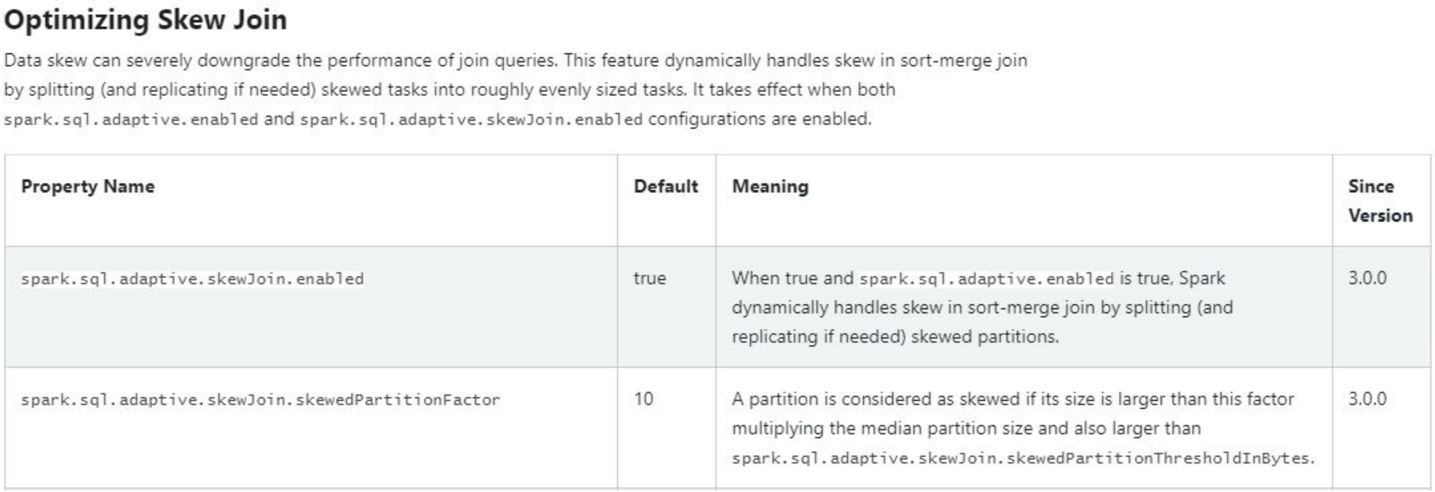
### Партиционирование

RDD и DF являются представляют собой классы, описывающие распределенные коллекции данных. Они (коллекции) разбиты на крупные блоки, которые называются партициями. В графе вычисления, который называется в Spark DAG (Direct Acyclic Graph), есть три основных компонента - `job` , `stage` , `task` .

`job` представляет собой весь граф целиком, от момента создания DF, до применения `action` к нему. Состоит из одной или более `stage` . Когда возникает необходимость сделать `shuffle` данных, Spark создает новый `stage` . Каждый `stage` состоит из большого количества `task` . `task` это базовая операция над данными. Одновременно Spark выполняет N `task` , которые обрабатывают N партиций, где N - это суммарное число доступных потоков на всех воркерах.

# А что в Spark?

- v.2 - ручной труд (см. MapReduce)
- v.3 - см. https://spark.apache.org/docs/latest/sql-performance-tuning.html#optimizing-skew-join



### Optimizing Skew Join

Data skew can severely downgrade the performance of join queries. This feature dynamically handles skew in sort-merge join by splitting (and replicating if needed) skewed tasks into roughly evenly sized tasks. It takes effect when both `spark.sql.adaptive.enabled` and `spark.sql.adaptive.skewJoin.enabled` configurations are enabled.

| Property Name | Default | Meaning | Since Version |
|---|---|---|---|
| `spark.sql.adaptive.skewJoin.enabled` | true | When true and `spark.sql.adaptive.enabled` is true, Spark dynamically handles skew in sort-merge join by splitting (and replicating if needed) skewed partitions. | 3.0.0 |
| `spark.sql.adaptive.skewJoin.skewedPartitionFactor` | 10 | A partition is considered as skewed if its size is larger than this factor multiplying the median partition size and also larger than `spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes`. | 3.0.0 |