BIGDATA TEAM

MegaFon Course: Big Data

Big Data Layout and File Formats

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

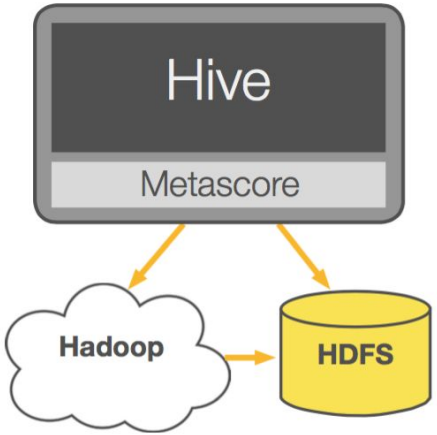
15.07.2019, Moscow, Russia

На этом занятии

- [18:30 - 19:20] Data Skew and Salting
  - или “Как правильно солить косые данные”
- -- перерыв, Q&A
- [19:30 - 20:20] Сжатие данных
  - Hive: File и Row Format
- -- перерыв, Q&A
- [20:30 - 21:30] RCFile, ORC, Parquet, Avro, ...

Hive DML: **Q&A**

Hive DML: Implementation **Q&A**

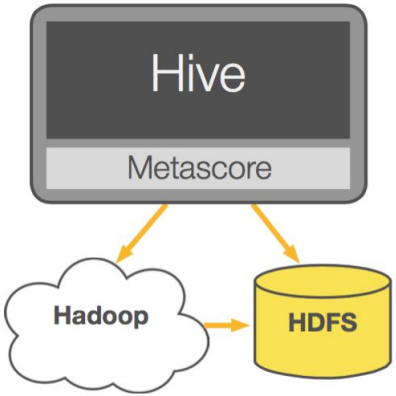


(1) DDL
(2) HiveQL (details)
(3) DML

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: +
```

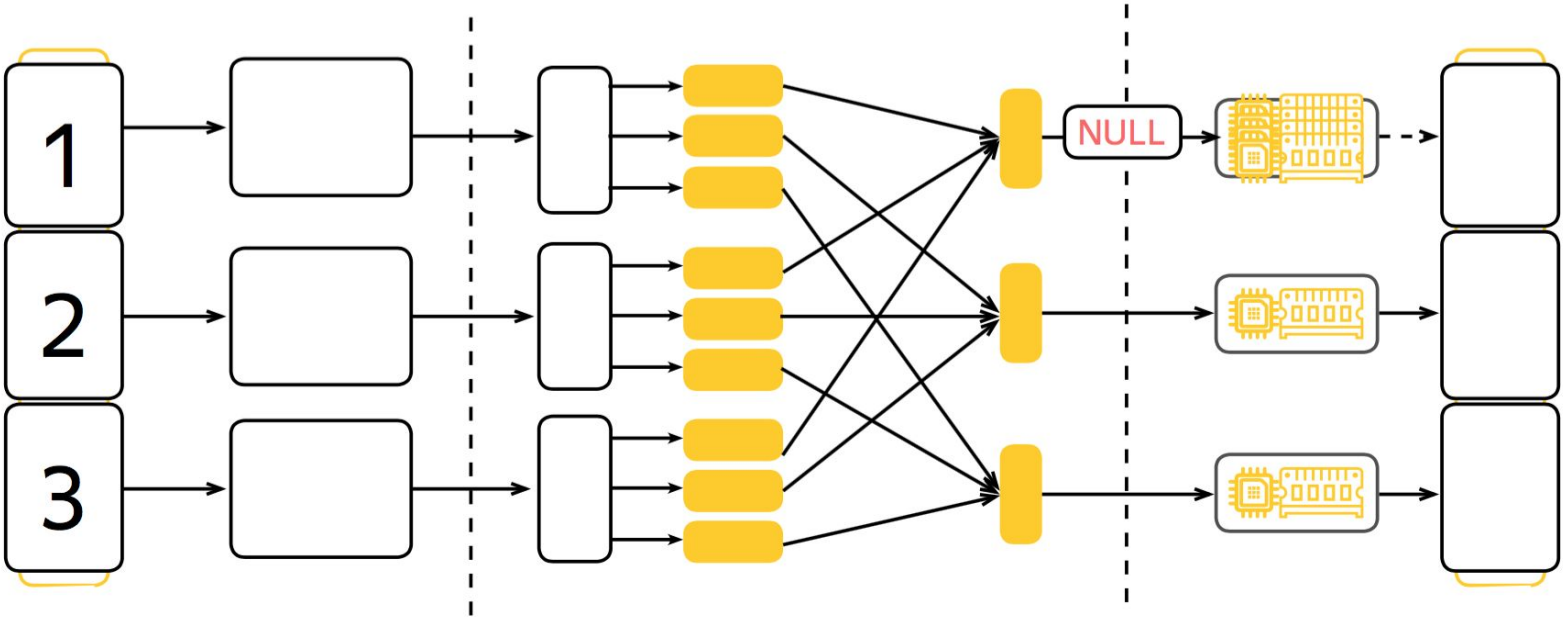
Data Skew and Salting

или

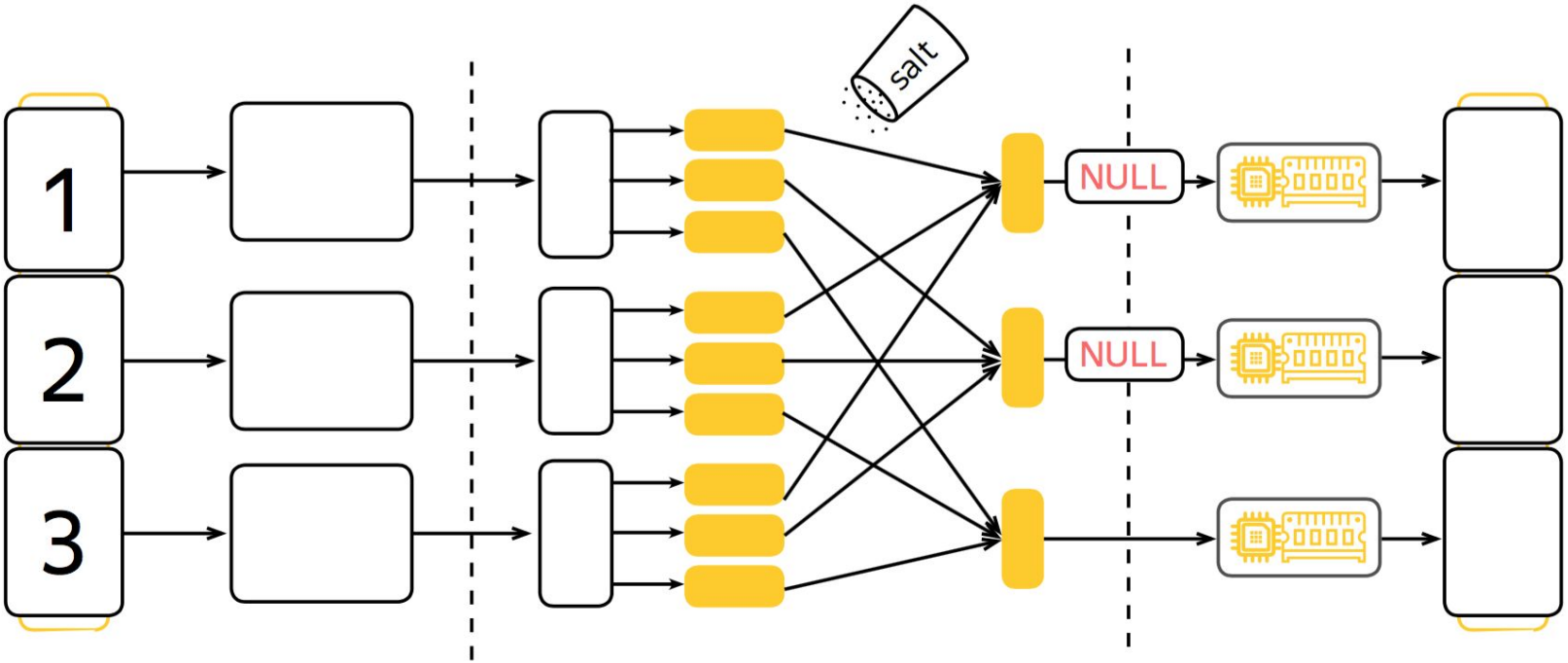
как правильно солить косые данные

BIGDATA TEAM

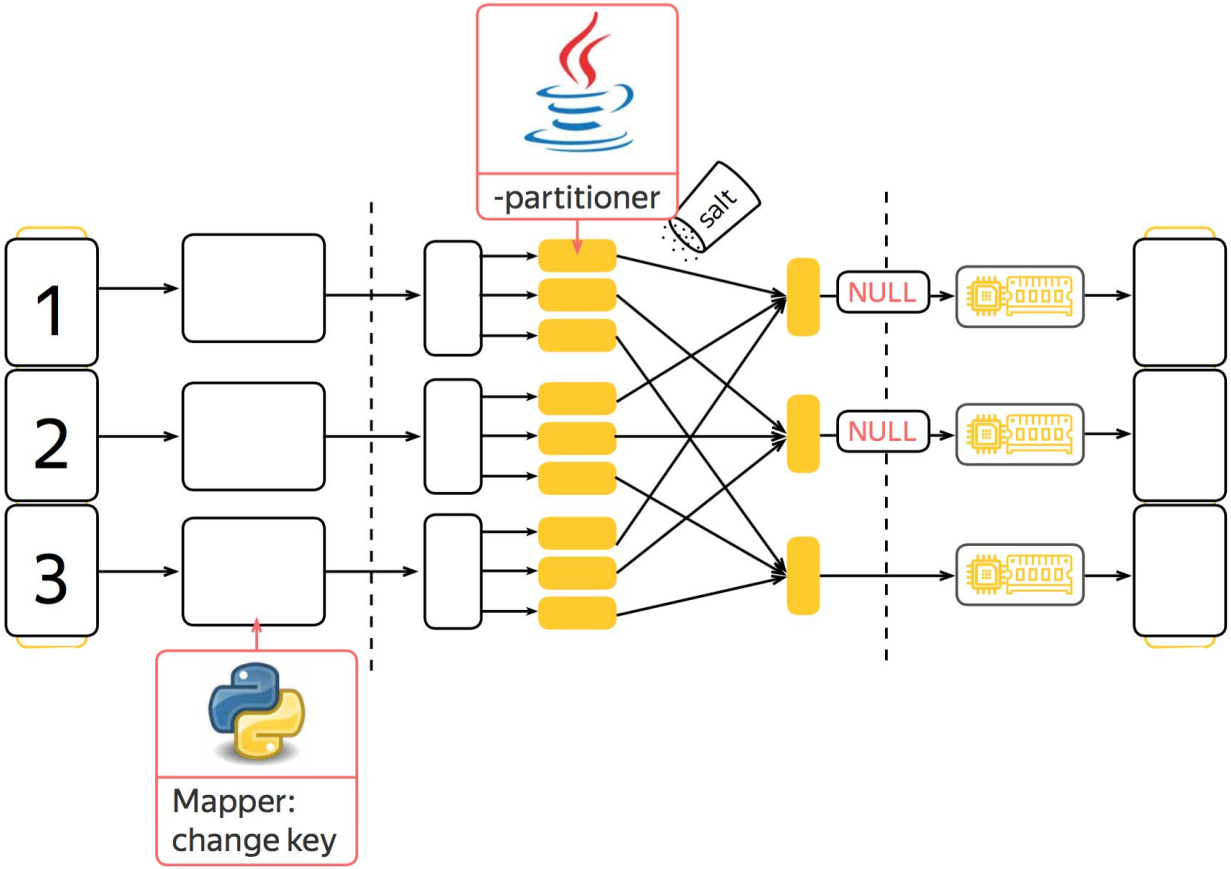
Dataset Skew (Nulls)



Соль ~~и перец~~



Solutions



Засол (Salting)

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

Пример вывода

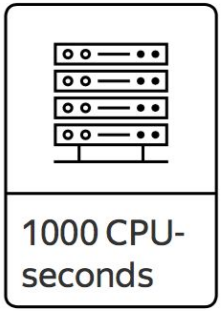
```
...
null_58  40989.56529872355
null_67  40775.58025775422
null_76  42430.98650098723
null_85  41811.88806991089
null_94  41086.03092382825
...
```

Вторая стадия соления

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

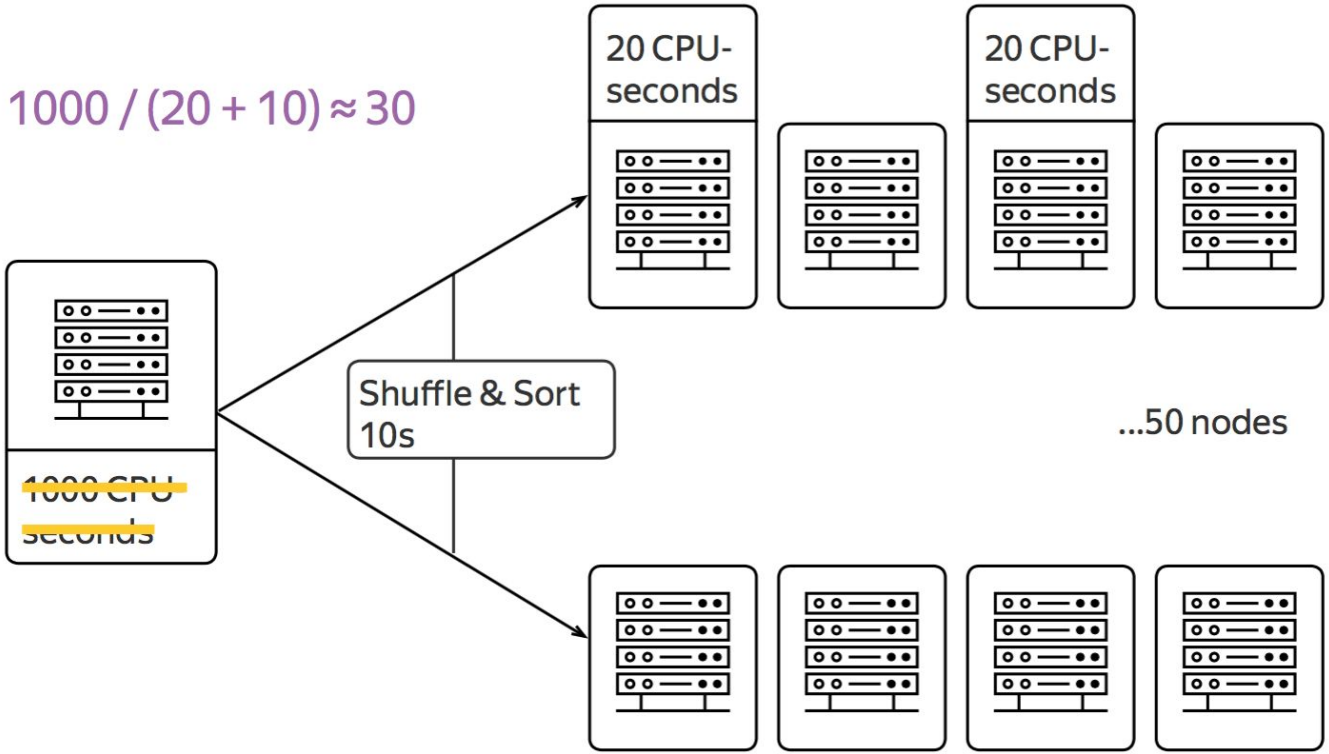
South 164302.58197312435
null 4145425.004916422
North 296659.74407499237

Оценка ускорения



BIGDATA TEAM

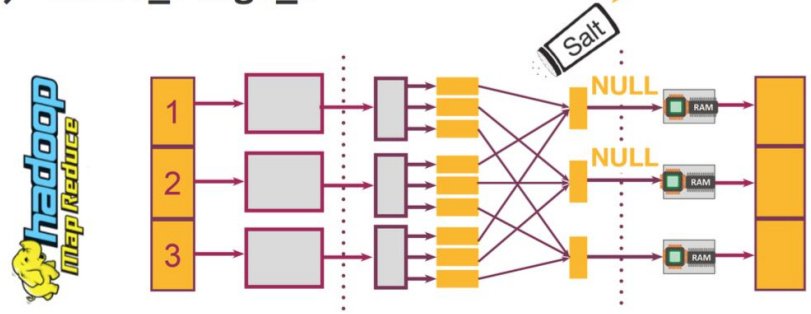
Оценка ускорения



Data Skew в Hive

Data Skew в Hive: версия 1

```
SET mapred.reduce.tasks = 128;
SELECT TRANSFORM(user_id, …)
    USING "./count.sh" AS user_id, some_stat
FROM (
    SELECT *
    FROM access_log
    DISTRIBUTE BY (
      hash(user_id)
      + IF(user_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
```

example

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

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

Data Skew 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

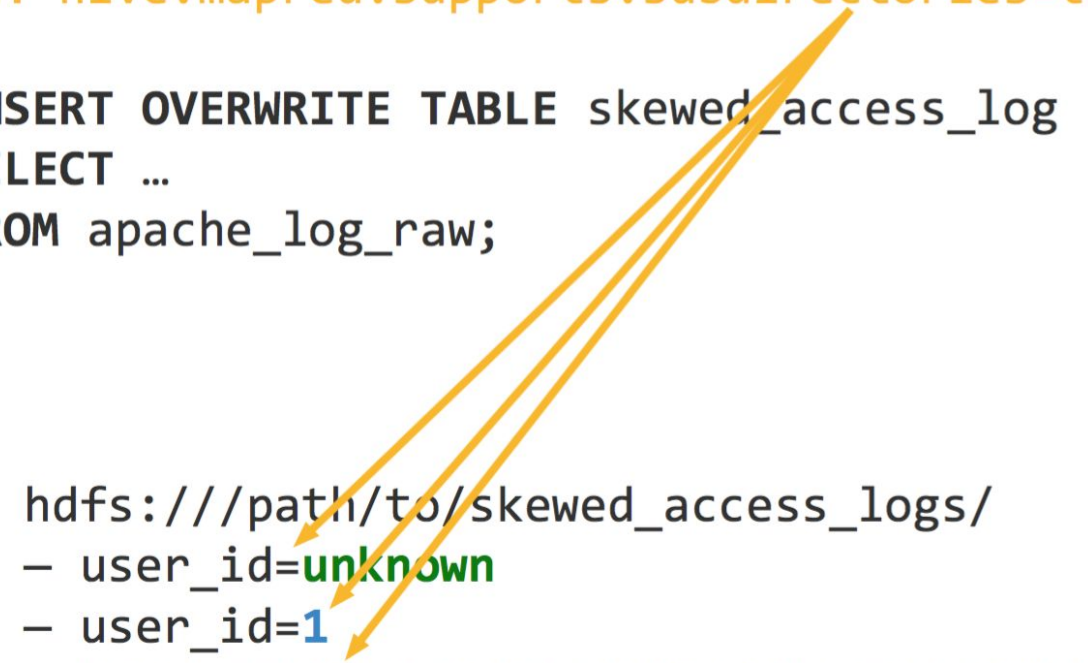
```
CREATE TABLE skewed_access_log (
    ip STRING,
    …
    user_id STRING,
    …
)
SKEWED BY (user_id) ON ("unknown", "1")
    STORED AS DIRECTORIES
…

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

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
```

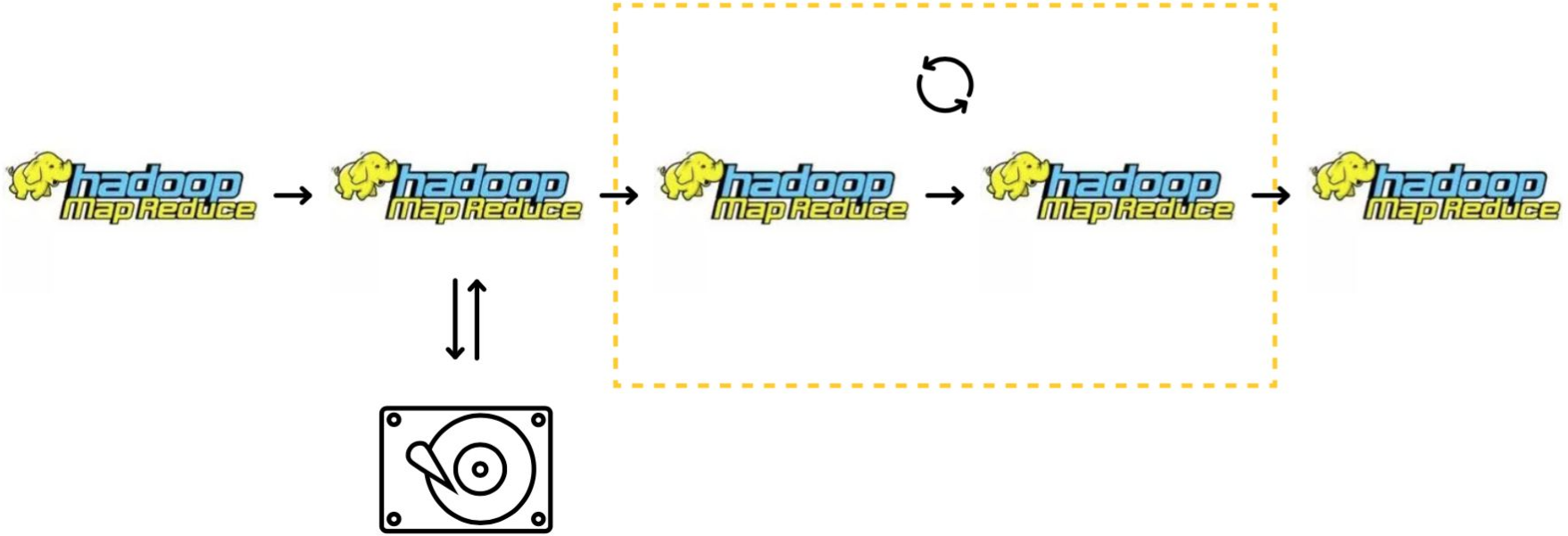
Skewed Table

```
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
```

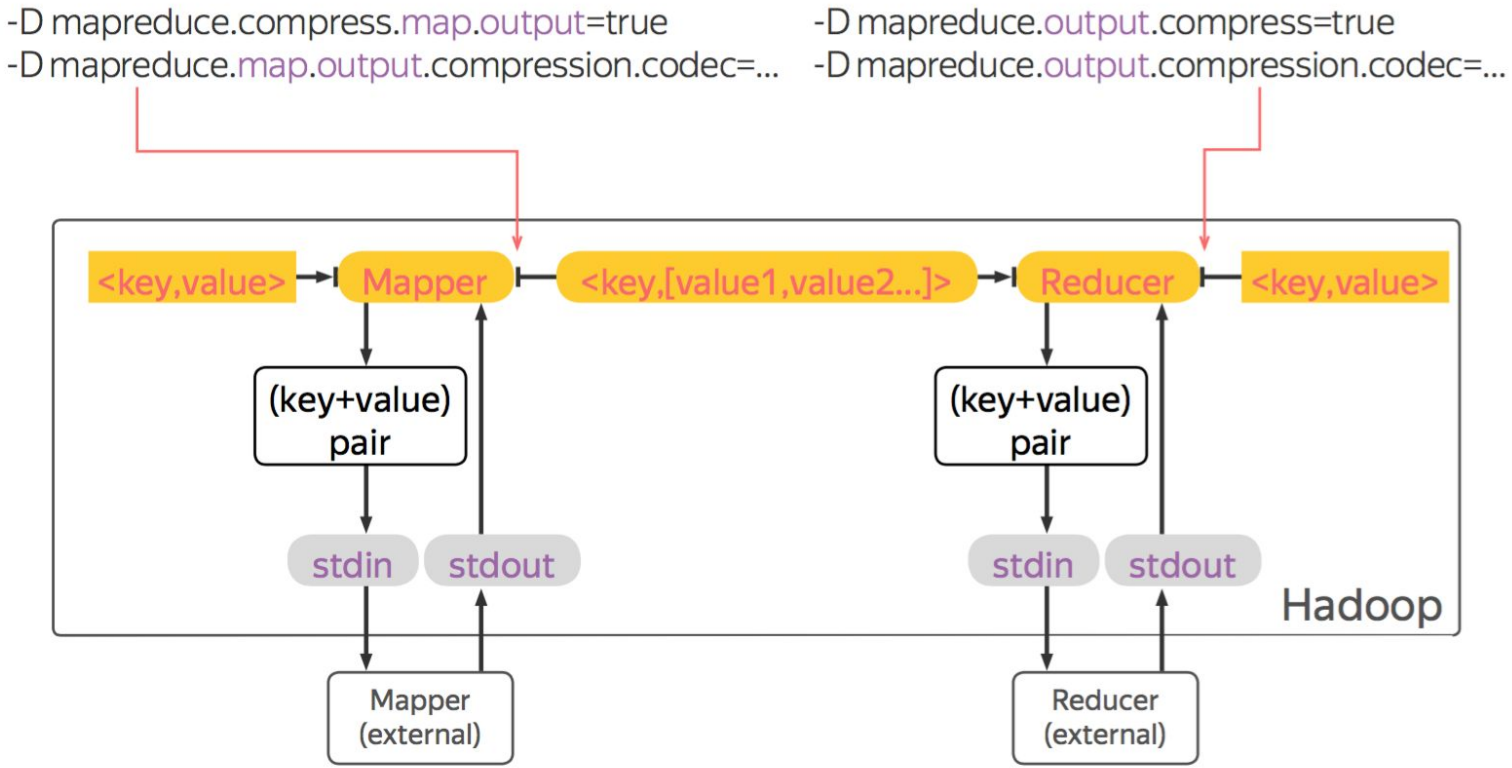
Сжатие данных

Disk I/O, Network Bandwidth



BIGDATA TEAM

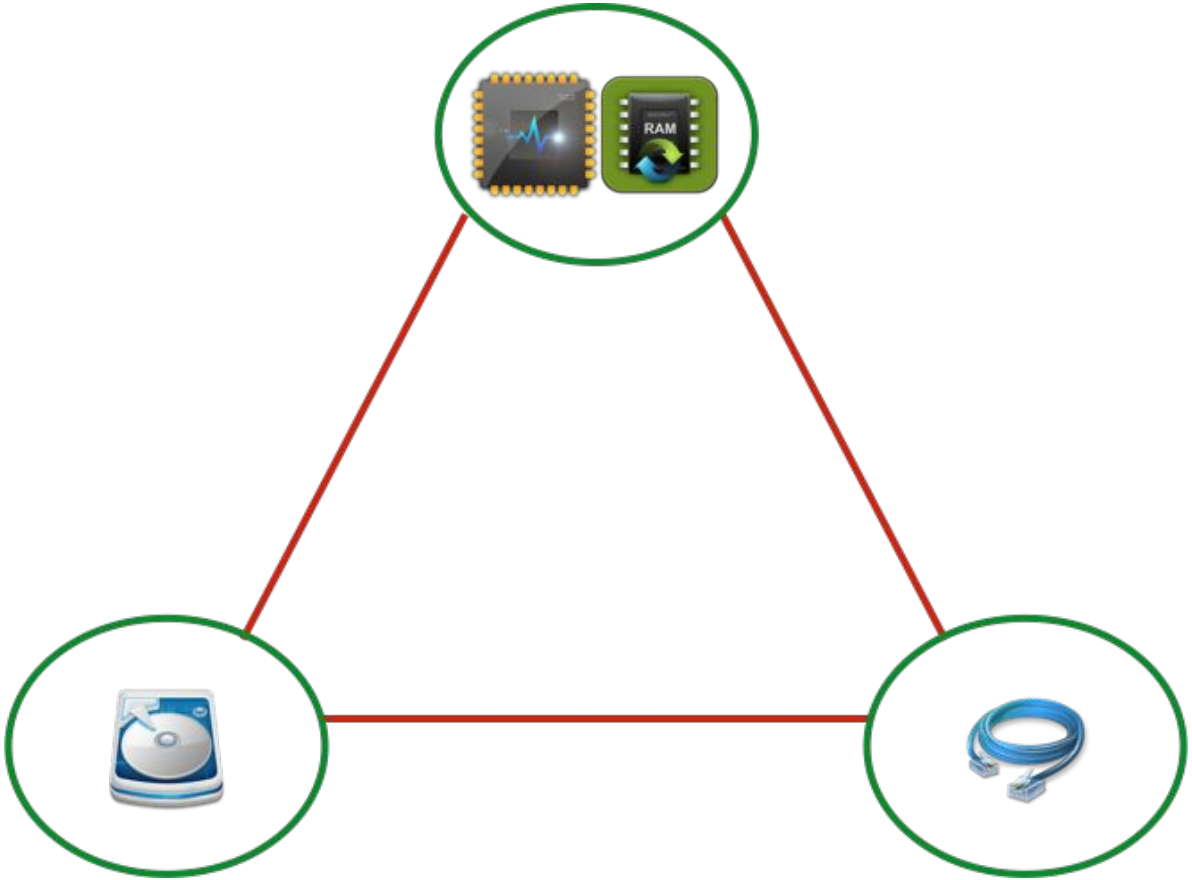
Сжатие данных в Hadoop



Spark: spark.io.compression.codec=..., spark.shuffle.compress=true

BIGDATA TEAM

Trade-off сжатия данных



Стандартные алгоритмы сжатия

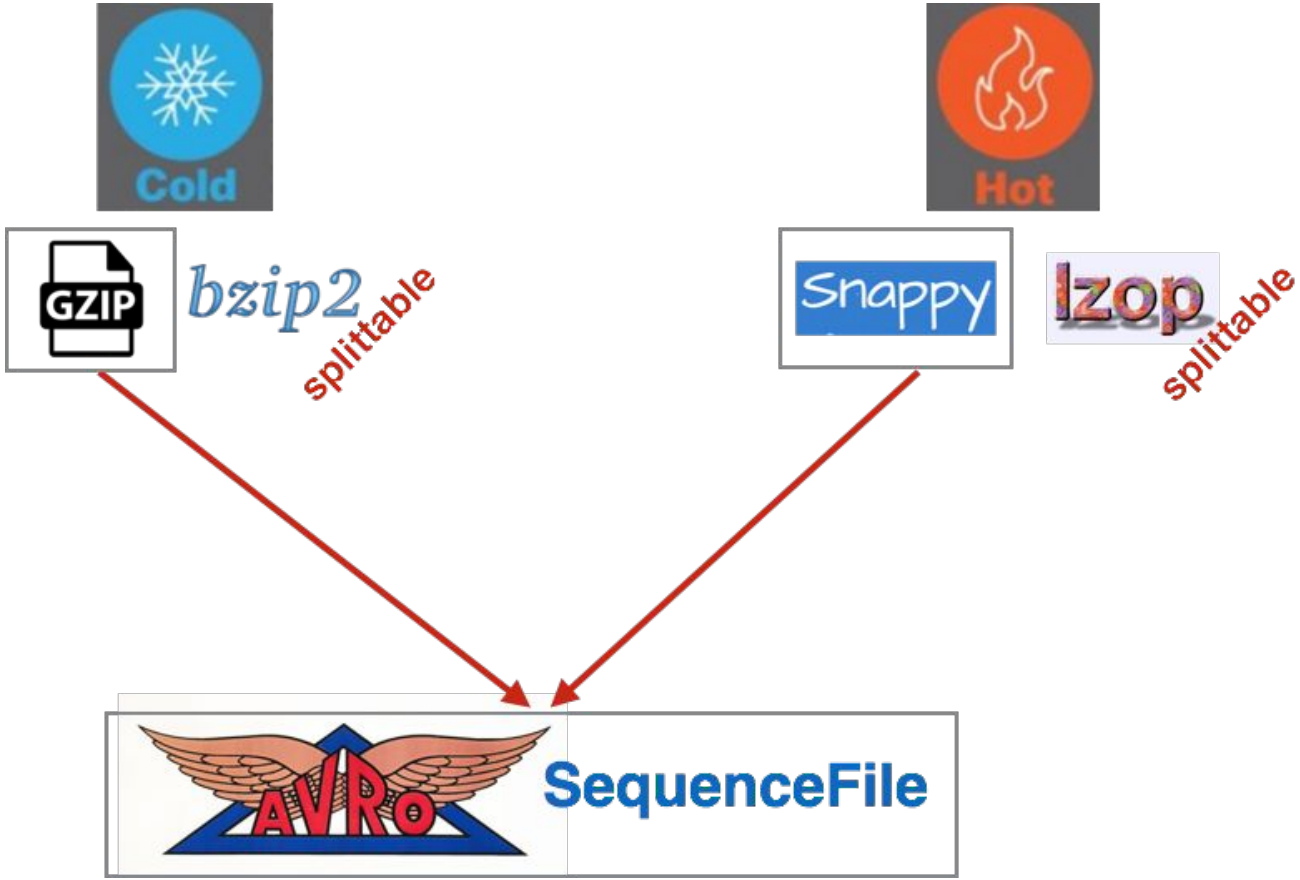
Compression Format	Splittable	Comments
.deflate .gz (gzip)	NO	Uses DEFLATE algorithm
.bz2 (bzip)	YES	more effective than gzip, but slower compression
.lzo	YES*	decompres way faster than gzip, compres less efficient
.snappy	NO	faster than LZO for decompression

flags: -1(optimised for speed)...-9(optimised for space)

Compression Codecs

Comptession format	Hadoop CompressionCodec
DEFLATE	org.apache.hadoop.io.compress.DefaultCodec
gzip	org.apache.hadoop.io.compress.GzipCodec
bzip2	org.apache.hadoop.io.compress.BZip2Codec
LZO	com.hadoop.compression.lzo.LzopCodec
LZ4	org.apache.hadoop.io.compress.Lz4Codec
Snappy	org.apache.hadoop.io.compress.SnappyCodec

Стандартные подходы



Hive: File и Row Format

Практика по Hive

```
create table tab_dataset (
    first_column string,
    second_column string,
    value int
) location '/user/<user>/hive_practice_data/';
```

```
first   line    1       NULL    NULL
second  line    3       NULL    NULL
last    line    5       NULL    NULL
\N      \N      10      NULL    NULL
Time taken: 3.805 seconds, Fetched: 4 row(s)
```

Hive DDL

<tab> or “\t”

Ctrl-A or “^A” (ASCII)

Hive DDL

Primitive

Category	Type	Description	Literal examples
Primitive	BOOLEAN	True/false value	TRUE
	TINYINT	1-byte (8-bit signed integer, from -128 to 127.	1Y
	SMALLINT	2-byte (16-bit) signed integer, from -32,768 to 32,767.	1S
	INT	4-byte (32-bit) signed integer, from -2,147,483,648 to 2,147,483,647.	1
	BIGINT	8-byte (64-bit) signed integer, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.	1L
	FLOAT	4-byte (32-bit) single-precision floating-point number.	1.0
	DOUBLE	8-byte (64-bit) double-precision floating-point nuber	1.0
	DECIMAL	Arbitrary-precision signed decimal number.	1.0
	STRING	Unbounded variable-length character string.	'a', "a"
	VARCHAR	Variable-length character string.	'a', "a"
	CHAR	Fixed-length character string.	'a', "a"
	BINARY	Byte array.	Not supported
	TIMESTAMP	Timestamp with nanosecond precision	1325502245000, '2012-01-02 03:04:05.123456789
	DATE	Date.	'2012-01-02'

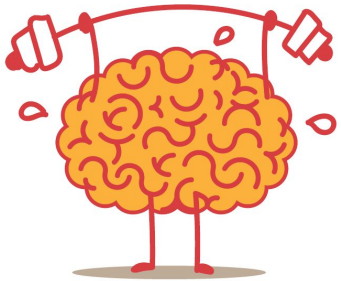
Complex

Category	Type	Description	Literal examples
Complex	ARRAY	An ordered collection of fieldss. The fields must all be of the same type.	array(1, 2)
	MAP	An unordered collection of key-value pairs. Keys must be primitives; values may be any type. For a particular map, the keys must be the same type, and the values must be the same type.	map('a', 1, 'b', 2)
	STRUCT	A collection of named fields. The fields may be of different types.	struct('a', 1, 1.0), named_ struct('col1', 'a', 'col2', 1, 'col3', 1.0)
	UNION	A value that may be one of a number of defined data types. The alue is tagged with an integer (zero-indexed) representing its data type in the union.	create_union(1, 'a', 63)

Hive DDL

```
CREATE EXTERNAL TABLE tab_dataset (
    first_column    STRING,
    second_column   STRING,
    value           INT
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\001'
  COLLECTION ITEMS TERMINATED BY '\002'
  MAP KEYS TERMINATED BY '\003'
  LINES TERMINATED BY '\n'
LOCATION '/user/adral/course2/week1/tab_dataset/';
```

Hive DDL



```
CREATE TABLE employees (
    name          STRING,
    salary        FLOAT,
    subordinates  ARRAY<STRING>,
    deductions    MAP<STRING, FLOAT>,
    address       STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>);
```

```
John Doe^A100000.0^AMary Smith^BTodd Jones^AFederal Taxes^C.2^BState
Taxes^C.05^BInsurance^C.1^A1 Michigan Ave.^BChicago^BIL^B60600

Mary Smith^A80000.0^ABill King^AFederal Taxes^C.2^BState
Taxes^C.05^BInsurance^C.1^A100 Ontario St.^BChicago^BIL^B60601
```

Programming Hive: Data Warehouse and Query Language for Hadoop

Hive DDL

```
CREATE EXTERNAL TABLE tab_dataset (
    first_column    STRING,
    second_column   STRING,
    value           INT
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\001'
  COLLECTION ITEMS TERMINATED BY '\002'
  MAP KEYS TERMINATED BY '\003'
  LINES TERMINATED BY '\n'
STORED AS file_format
LOCATION '/user/adral/course2/week1/tab_dataset/';
```

default: TEXTFILE

RCFile

BIGDATA TEAM

Record Columnar File (RCFile)

1. Быстрая загрузка данных в хранилище
2. Высокая скорость обработки запросов
3. Эффективное использование жесткого диска
4. Адаптивность к динамическому изменению паттернов аналитических запросов

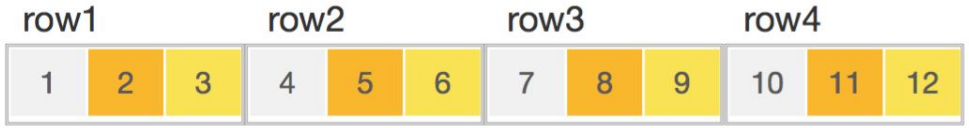
[Facebook] 2011 - RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems

Record Columnar File (RCFile)

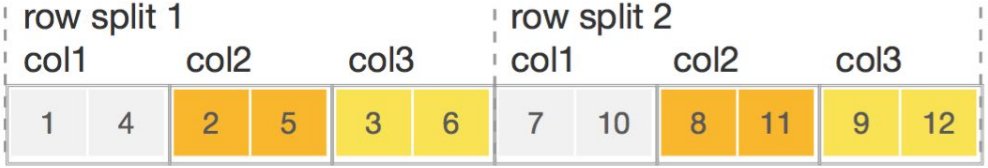
Logical table

	col1	col2	col3
row1	1	2	3
row2	4	5	6
row3	7	8	9
row4	10	11	12

Row-oriented layout



Column-oriented layout (RCFile)

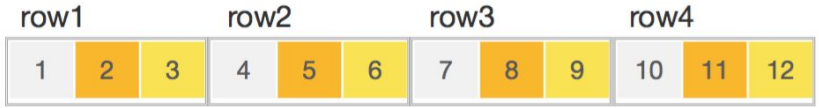


Record Columnar File (RCFile)

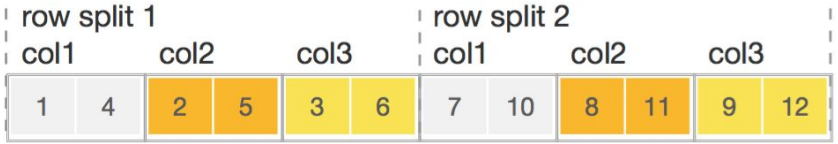
Logical table

	col1	col2	col3
row1	1	2	3
row2	4	5	6
row3	7	8	9
row4	10	11	12

Row-oriented layout



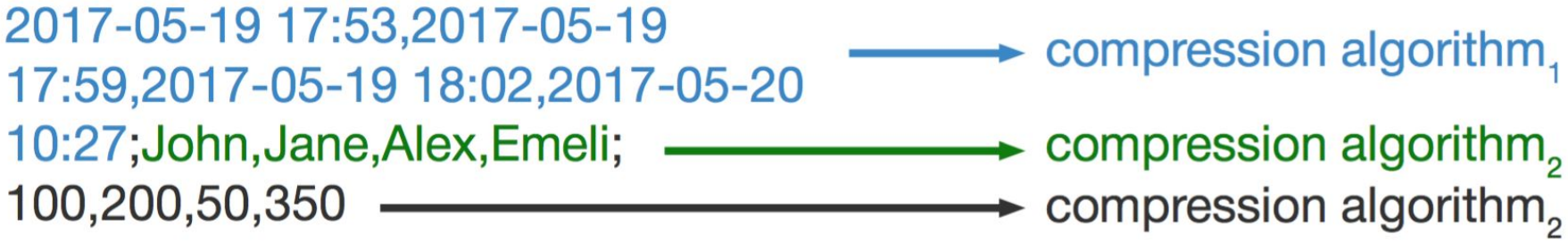
Column-oriented layout (RCFile)



date	user	order
2017-05-19 17:53	John	100
2017-05-19 17:59	Jane	200
2017-05-19 18:02	Alex	50
2017-05-20 10:27	Emeli	350

2017-05-19 17:53,2017-05-19 17:59,2017-05-19 18:02,2017-05-20 10:27;John,Jane,Alex,Emeli;
100,200,50,350

Кодирование и сжатие данных



BIGDATA TEAM

Кодирование Datetime

2017-05-19 17:53,2017-05-19 17:59,2017-05-19 18:02,2017-05-20 10:27;John,Jane,Alex,Emeli;
100,200,50,350

→ compression algorithm$_1$

2017-05-19 17:53
2017-05-19 17:59
2017-05-19 18:02
2017-05-20 10:27

↓

1495205580
1495205940
1495206120
1495265220

↓

Delta-кодирование

2017-05-19 17:53,2017-05-19 17:59,2017-05-19 18:02,2017-05-20 10:27;John,Jane,Alex,Emeli;
100,200,50,350

→ compression algorithm$_1$

2017-05-19 17:53
2017-05-19 17:59
2017-05-19 18:02
2017-05-20 10:27

↓

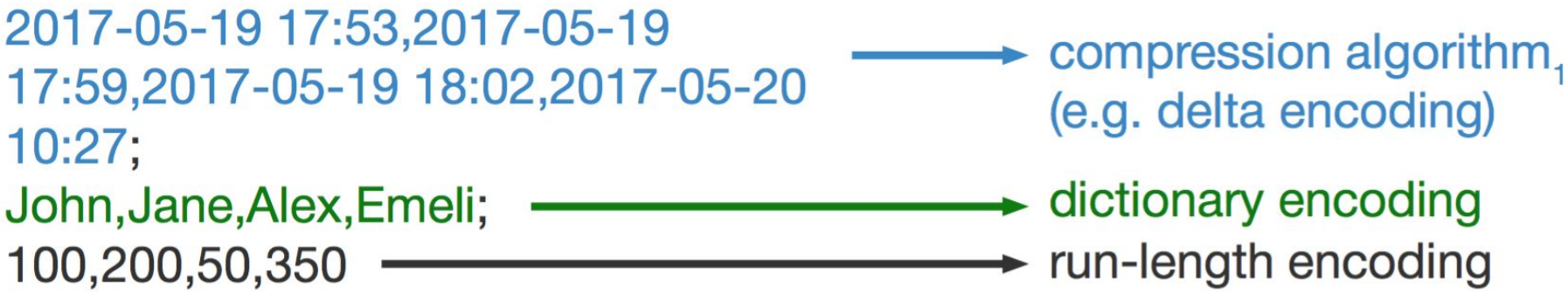
1495205580
1495205940
1495206120
1495265220

↓

δ-encoding

1495205580
+360
+180
+59100

Record Columnar File (RCFile)



BIGDATA TEAM

Record Columnar File (RCFile)

1. Быстрая загрузка данных в хранилище
2. Высокая скорость обработки запросов
3. Эффективное использование жесткого диска
4. Адаптивность к динамическому изменению паттернов аналитических запросов

ORC

ORC

1. Быстрая загрузка данных в хранилище
2. Высокая скорость обработки запросов
3. Эффективное использование жесткого диска
4. Адаптивность к динамическому изменению паттернов аналитических запросов

ORC = Optimized Row Columnar (File Format)

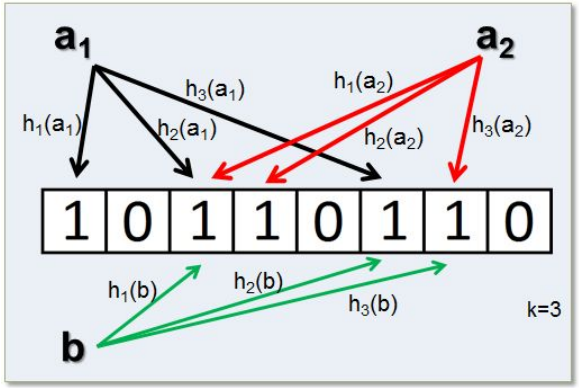
ORC и Parquet

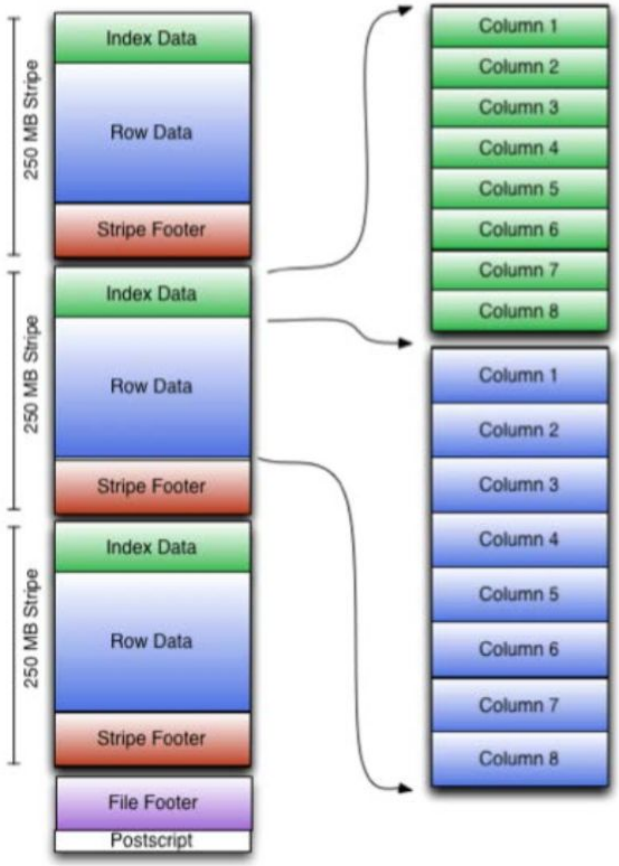
- ORC = Optimized Row Columnar (File Format)

- Parquet - основан на статье Google Dremel (для вложенных структур данных, nested structures)

ORC

Bloom Filters, extra stats, ...





https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

ORC Stats

```
message IntegerStatistics {
 optional sint64 minimum = 1;
 optional sint64 maximum = 2;
 optional sint64 sum = 3;
}
```

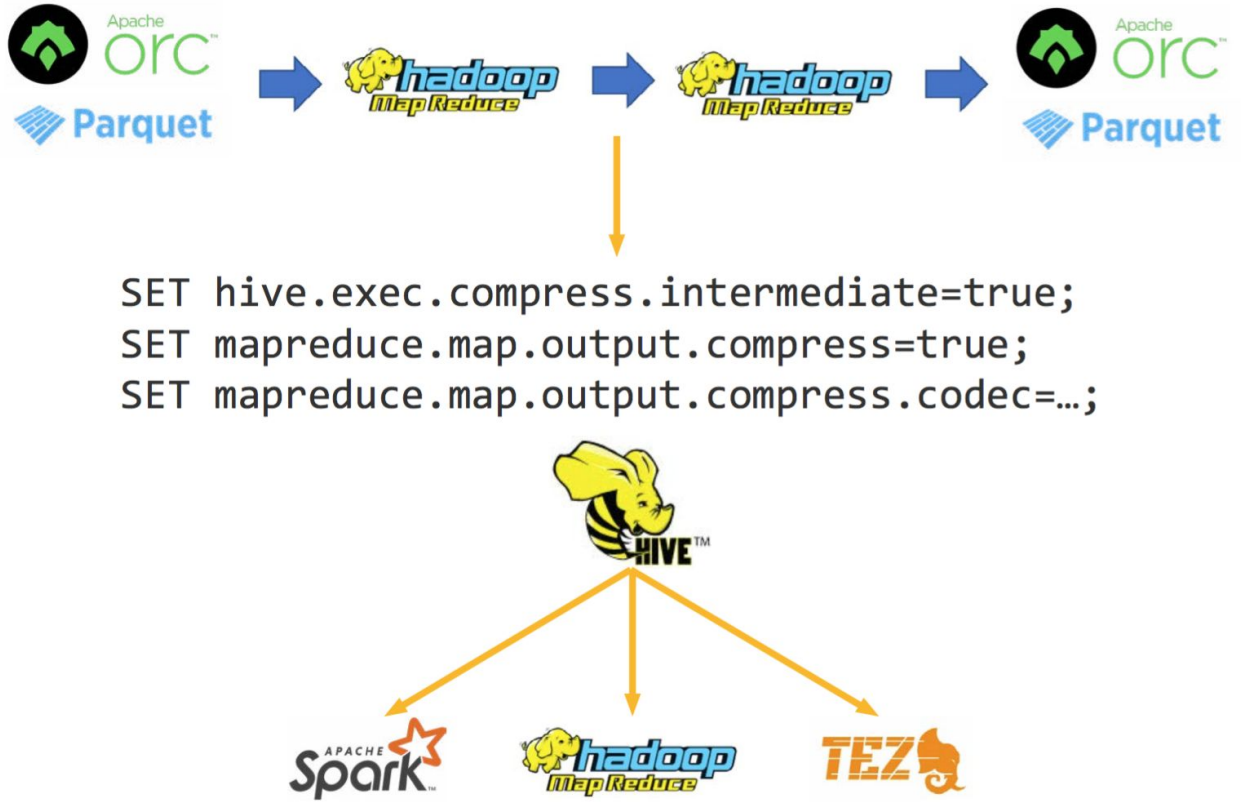
source: https://orc.apache.org/specification/ORCv1/

ORC Parameters

Key	Default	Notes
orc.compress	ZLIB	high level compression (one of NONE, ZLIB, SNAPPY)
orc.compress.size	262,144	number of bytes in each compression chunk
orc.stripe.size	67,108,864	number of bytes in each stripe

```
CREATE TABLE my_orc_table (
    …
)
STORED AS orc
    TBLPROPERTIES ("orc.compress"="NONE")
…;
```

Сжатие в Hive



ORC, Parquet, Avro, …

Google Dremel

- https://ai.google/research/pubs/pub36632 - Dremel: Interactive Analysis of Web-Scale Datasets (2010)

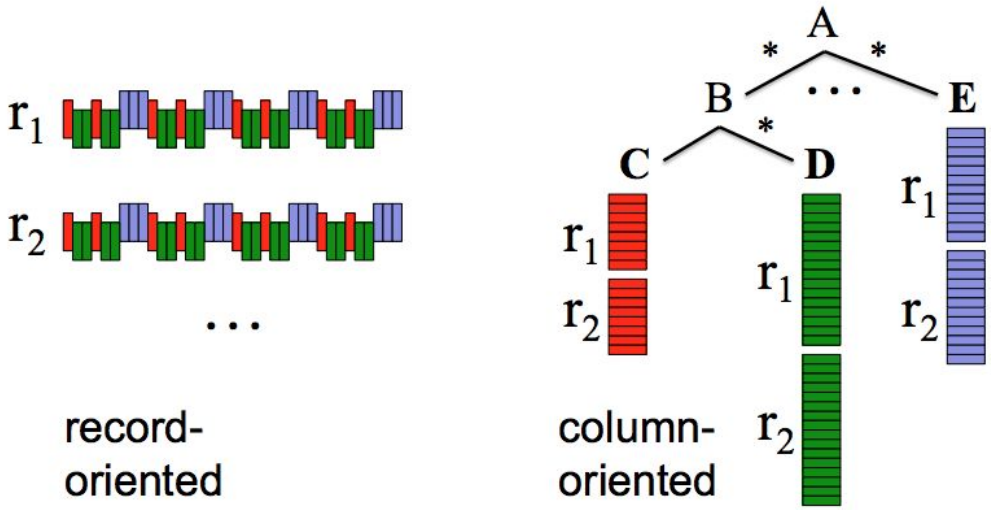


Figure 1: Record-wise vs. columnar representation of nested data

Google Dremel*

```
DocId: 10                    r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
```

```
DocId: 20                    r2
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```

Figure 2: Two sample nested records and their schema

DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

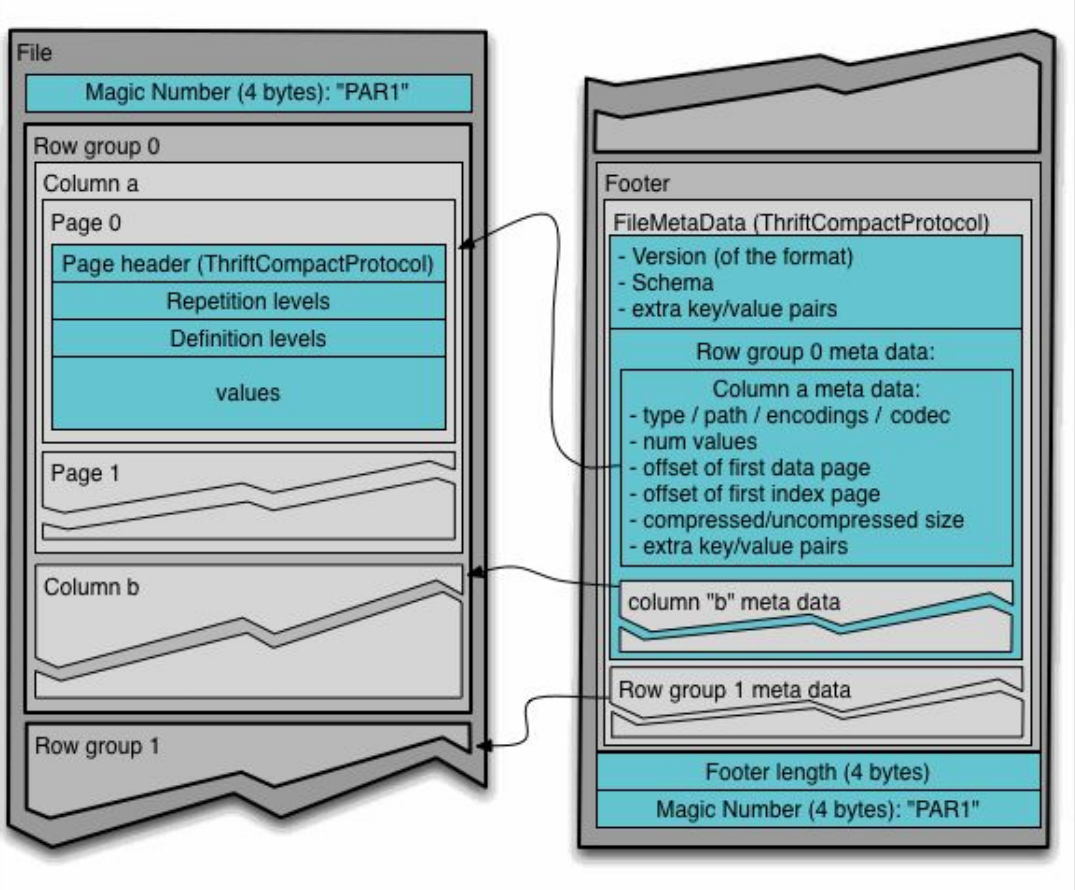
Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

Figure 3: Column-striped representation of the sample data in Figure 2, showing repetition levels (r) and definition levels (d)

BIGDATA TEAM

Внутренности Parquet



- File ~ блок в HDFS
- Row group ~ stripe
- Column Chunk
- Page

Parquet vs ORC

Parquet

- поддерживает версионирование
- поддерживает вложенные структуры данных, но поддерживает только 7 примитивных типов
- Ориентирован на мир во вне Hadoop

ORC

- поддерживает статистику и Bloom-фильтры
- меньше вероятность порчи данных
- ориентирован на Hadoop (см. транзакции в Hive)

Avro, Thrift и Protobuf

Apache Thrift ™

BIGDATA TEAM

Примеры из практики*

Яndex маркет

Яндекс Новости

Apache orc™

Parquet

protobuf
Protocol Buffers

*личный опыт

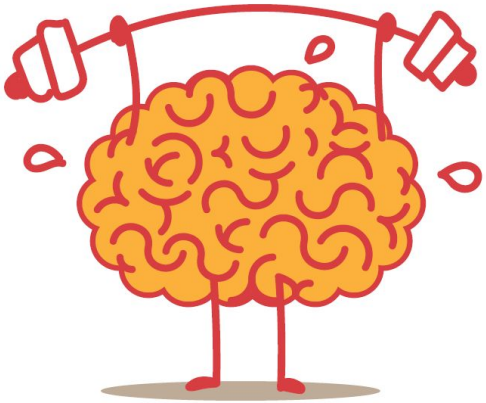
Резюме

- Вы умеете бороться с Data Skew (и в Hadoop и в Hive)
- Вы умеете настроить параметры сжатия данных для входных, выходных и промежуточных данных Hadoop, Hive, Spark приложений
- Вы умеете пользоваться ORC / Parquet форматом файлов в Hive
- Вы можете перечислить 4 design goals Big Data приложений, которым следуют поколоночные типы хранения данных (RCFile, ORC, Parquet)
- Вы можете привести кейсы использования Avro, Parquet, ORC

BIGDATA TEAM

Загрузка данных в хранилище

- Пример: имена людей



Thank you! Questions?

Feedback: http://rebrand.ly/mf2019q2_feedback_05_datalayout

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

BIGDATA TEAM

Материалы для погружения

Полезная статья: Scaling the Facebook data warehouse to 300 PB

Полезные книги про оптимизацию кодеков:

- “Hadoop in Practice” и “Hadoop the Definitive Guide”

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

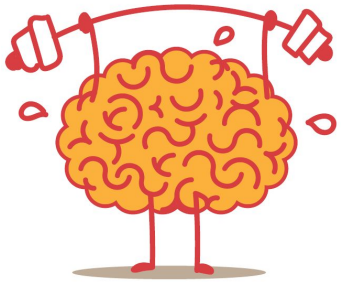
Материалы для погружения

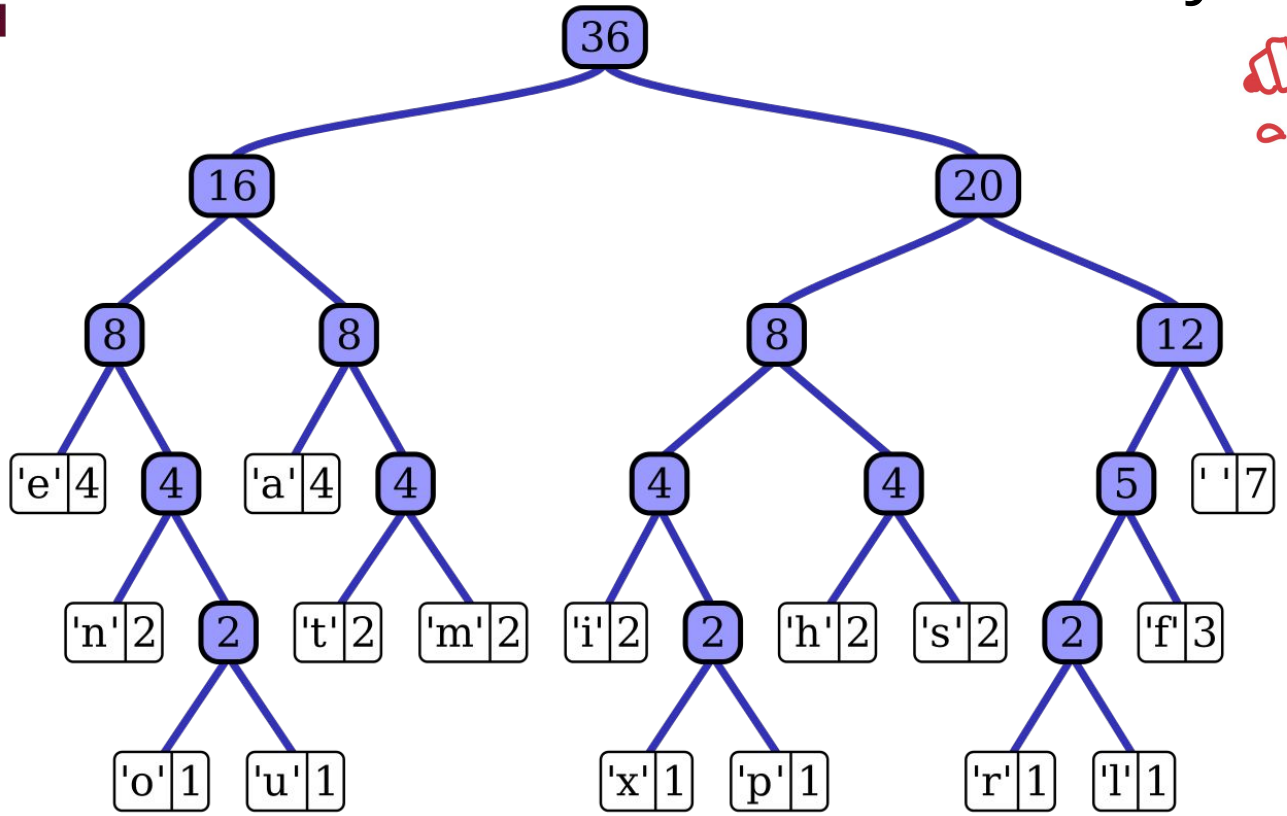
- Настраиваем Hive таблицы поверх данных в HDFS: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL
- Пользуемся клаузой “EXPLAIN” для того, чтобы понять план выполнения Hive-запроса: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Explain
- ORC Language Manual: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC\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

Энтропийная кодировка

BIGDATA TEAM

Dictionary Encoding





Код Хаффмана, LZW, ...