SECON'2017

Provectus

#SECONRU

IX МЕЖРЕГИОНАЛЬНАЯ КОНФЕРЕНЦИЯ РАЗРАБОТЧИКОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Колоночные структуры данных на примере Parquet

Блохин Леонид

BigDataEngineer at Provectus

21-22 АПРЕЛЯ | ПЕНЗА

Содержание доклада:

Отличия строковых и колоночных баз данных.
- особенности
- области применения
- преимущества и недостатки

Apache Parquet,
- области применения,
- преимущества и недостатки.

Apache Spark,
- области применения, отличительные особенности,
- приемущества и недостатки, работа с parquet файлами в Hadoop File System.

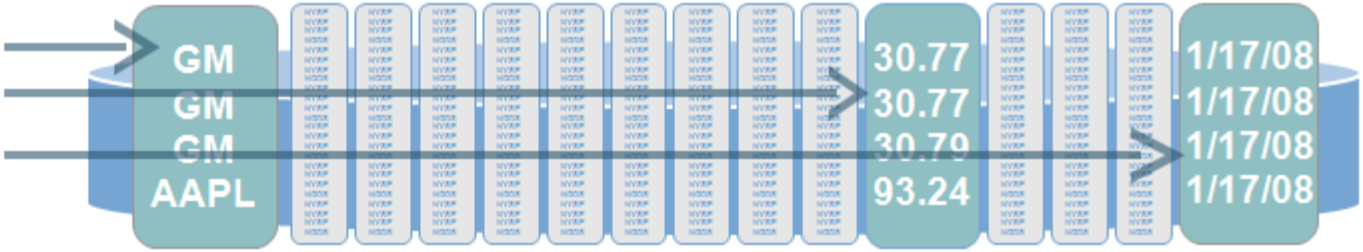
RDD, DataFrames, и Datasets в Apache Spark,
- зачем они нужны,
- как ими пользоваться,
- какие профиты.

Mist, используем Spark, как сервис

Отличия колоночных БД от строковых

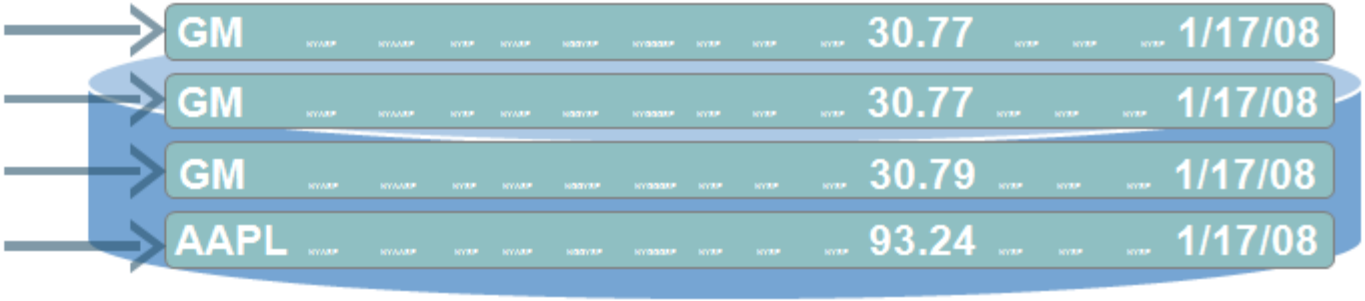
Хранение по столбцам

Чтение 3х столбцов



Хранение по строкам

Чтение всех столбцов



Области применения колоночных БД

Событийные таблицы, над которыми выполняются сложные выборки (агрегации, фильтры, сортировки). Например, добавление товаров в корзину в Интернет магазине. Метрики рекламы, нагрузок ...

Агрегатные таблицы с большим количеством данных для аналитических выборок. Это часто таблицы, которые строятся из событийных таблицы. Это может быть таблицы со статистикой добавления товаров в корзину по дням, категориям, ценам и другим параметрам.

Преимущества

Колоночные базы данных позволяют эффективно делать сложные выборки на больших таблицах.

Изменение структуры больших таблиц происходит мгновенно.

Сжатие данных позволяет сэкономить место

Недостатки

Медленно работают на запись

Не поддерживают транзакции

Имеют ряд ограничений для разработчика (sql синтаксис не полный)

Обычные выборки по ключу всех столбцов без профита


HIVE
Spark SQL
VERTICA
PARACCEL
SAND TECHNOLOGY
kognitio
INFOBRIGHT

Hadoop Distributed File System,

hbase,

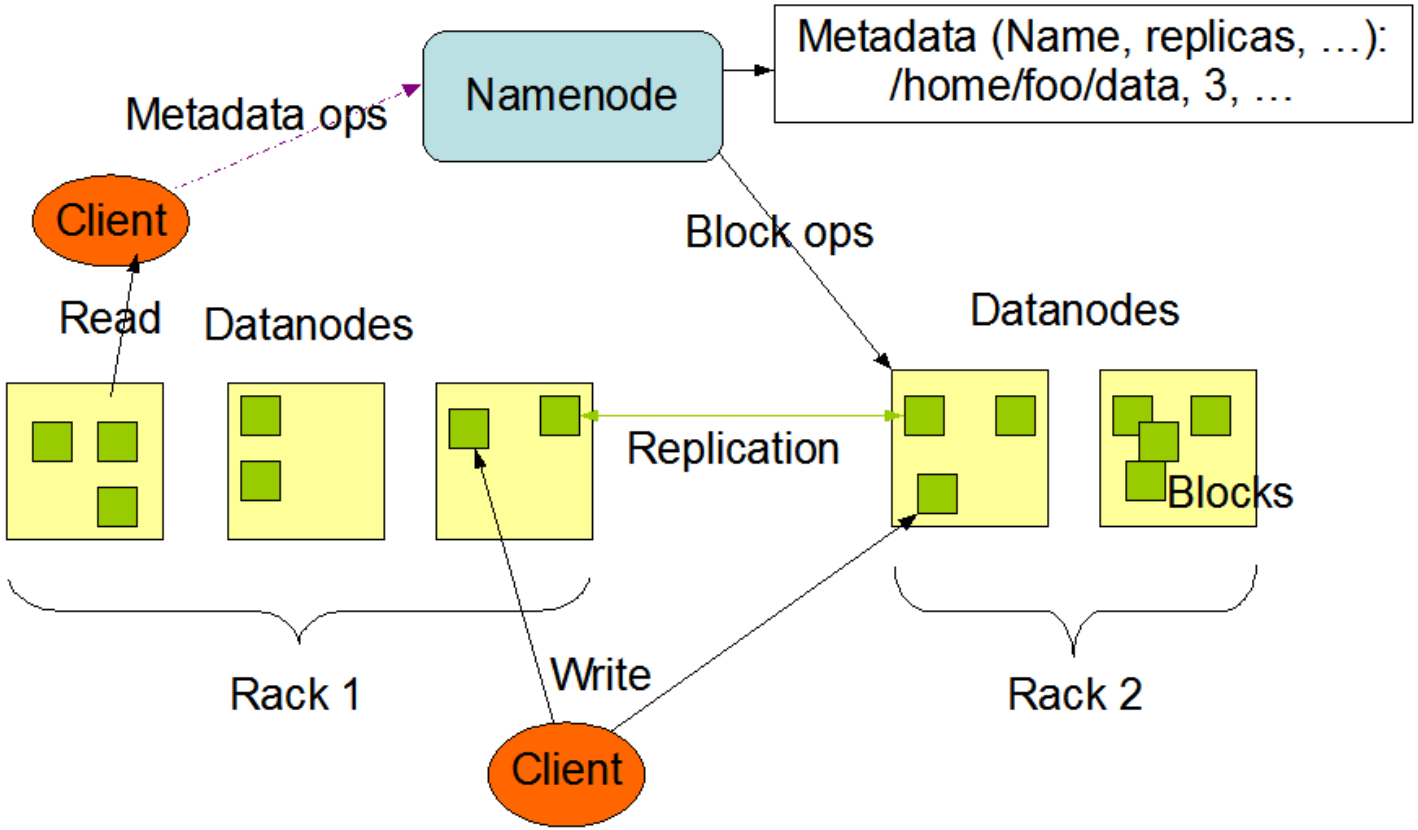
S3,

cassandra,

gfs,

ceph ...

HDFS Architecture



CSV, JSON, XML	Sequence file	Avro	Parquet	ORC
	Строко ориентированный формат	Бинарный с компрессией и сериализацией	Колоночно ориентированный формат	Смешанный строко и колоночно ориентированный

Parquet

Parquet

Колоночный формат, Работает в HDFS

Поддерживает сжатие(snappy, gzip, lzo, brotli)

Метадата кодируется Apache Thrift

Поддерживает вложенные данные

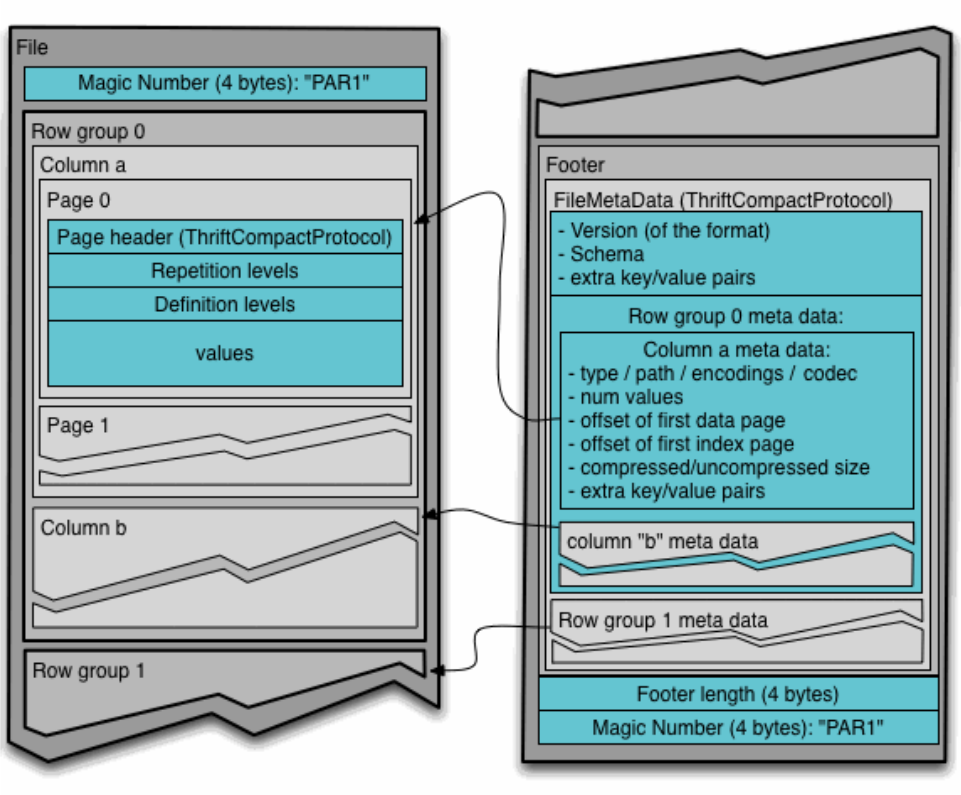
The striping and assembly algorithms from the Dremel paper

Файлы имеют несколько уровней разбиения на части, благодаря чему возможно довольно эффективное параллельное выполнение операций поверх них:

Row-group — это разбиение, позволяющее параллельно работать с данными на уровне Map-Reduce

Column chunk — разбиение на уровне колонок, позволяющее распределять IO операции

Page — Разбиение колонок на страницы, позволяющее распределять работу по кодированию и сжатию



типы

BOOLEAN: 1 bit boolean

INT32: 32 bit signed ints

INT64: 64 bit signed ints

INT96: 96 bit signed ints

FLOAT: IEEE 32-bit floating point values

DOUBLE: IEEE 64-bit floating point values

BYTE_ARRAY: arbitrarily long byte arrays.

Достоинства Parquet

Эффективное хранение

Высокая скорость сложных выборок, если не нужны все колонки сразу

Быстрая работа на чтение, по сравнению с использованием других файловых форматов

HDFS, NFS, GlusterFS

Недостатки parquet

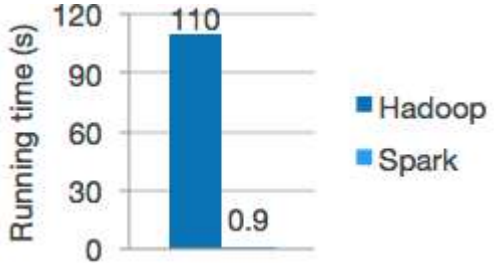
parquet ведёт себя как неизменяемая таблица или БД

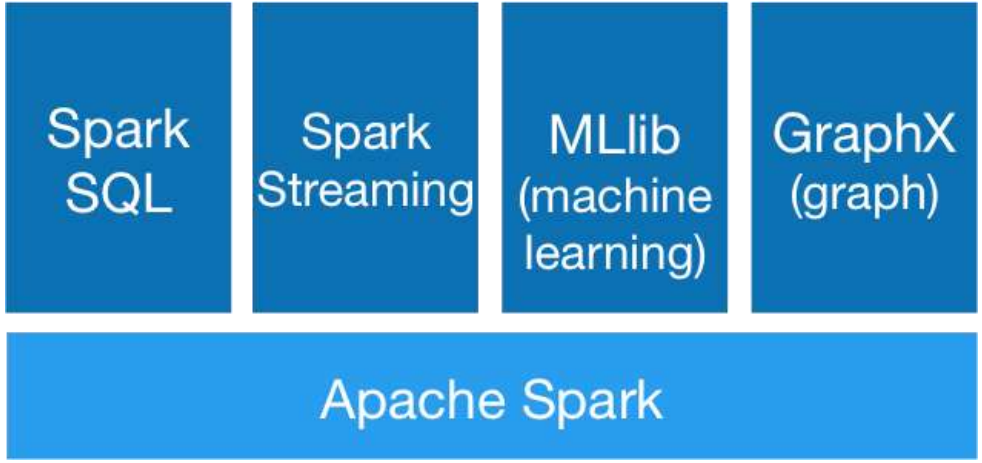
Для колонок определён тип, нельзя комбинировать сложный тип данных с простым.

Не поддерживаются транзакции


ЭКОСИСТЕМЫ.
ОЗЕРО
БОЛОТО
ЛУГ
ТОРФ


APACHE
Spark
TM


Running time (s)
120
90
60
30
0
110
0.9
Hadoop
Spark


Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

```
val peopleDF = spark.read.json("examples/src/main/resources/people.json")
```


```
peopleDF.write.parquet("people.parquet")
val parquetFileDF = spark.read.parquet("people.parquet")
```

```
parquetFileDF.createOrReplaceTempView("parquetFile")
val namesDF = spark.sql("SELECT name FROM parquetFile WHERE age BETWEEN 13 AND 19")
namesDF.map(attributes => "Name: " + attributes(0)).show()
```

```
+----------------+
|      value     |
+----------------+
|Name: Justin|
+----------------+
```

Resilient Distributed Dataset

Распределенный набор данных, который знает как себя вычислить.

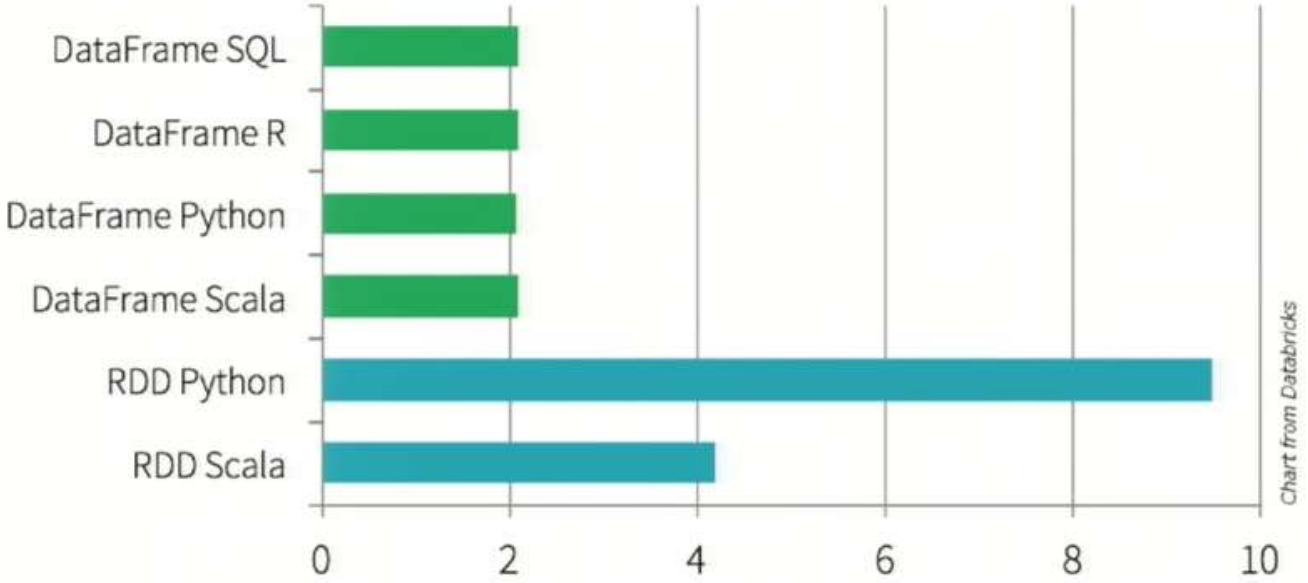
RDD может быть лениво вычисляемой при запросе

может быть и материализована — распределенно, в памяти или на диске (или в памяти с вытеснением на диск)

разбита на партиции — это минимальный объем RDD, который будет обработан каждым рабочим узлом.

DataFrames are faster

Because of the optimization, they tend to outperform RDDs.



Example Optimization

```
users.join(events, users("id") === events("uid"))
     .filter(events("date") > "2015-01-01")
```

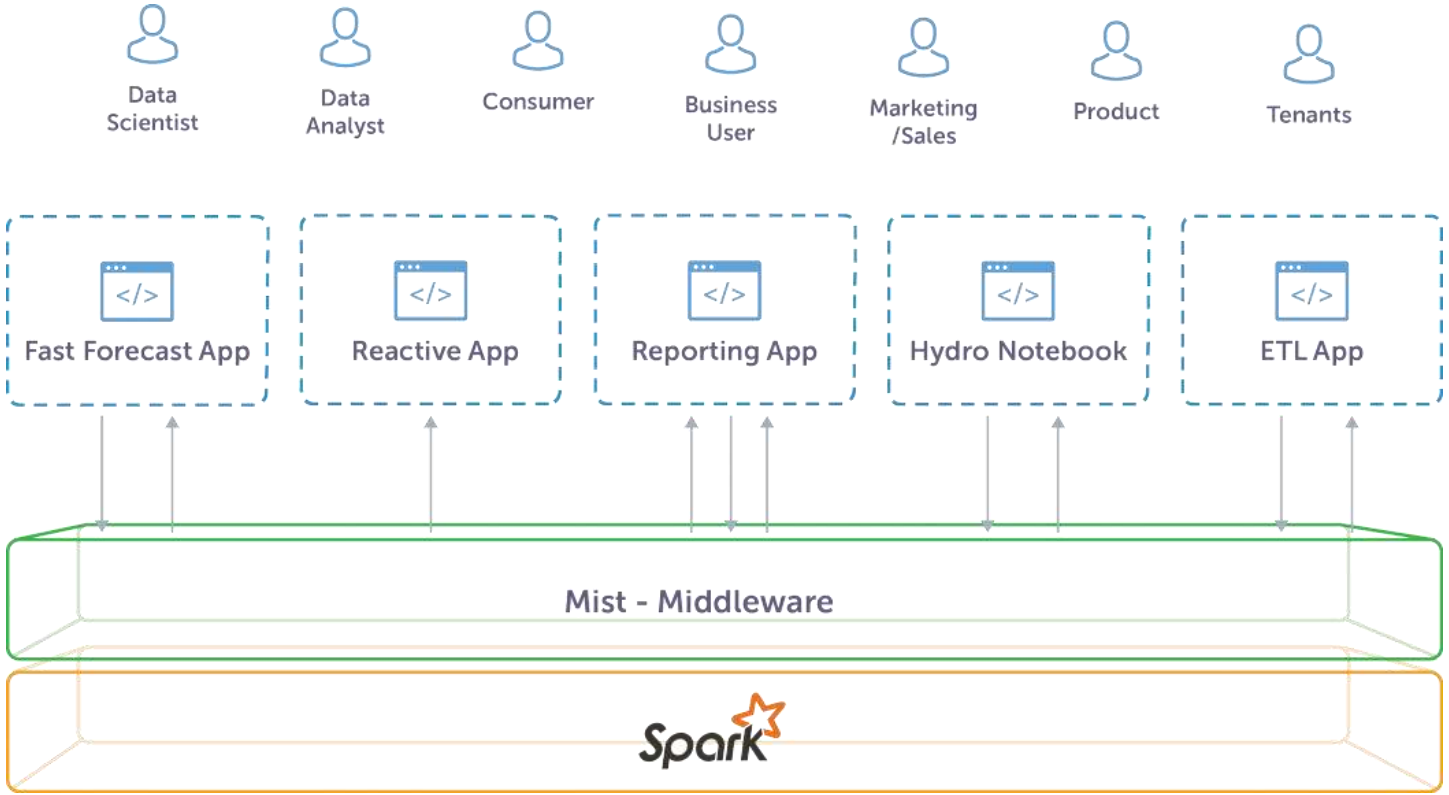
Enter Datasets

Like an RDD, Dataset has a *type*.

```
// Read a DataFrame from a JSON file
val df = sqlContext.read.json("people.json")
// Convert the data to a domain object.
case class Person(name: String, age: Long)
val ds: Dataset[Person] = df.as[Person]
```

hydrosphere.io

Provectus



SECON'2017

IX МЕЖРЕГИОНАЛЬНАЯ КОНФЕРЕНЦИЯ РАЗРАБОТЧИКОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Блохин Леонид

BigDataEngineer at Provectus

Blokhin@provectus.com

8 917 295-40-49