

SQL поверх больших данных

Андрей Кузнецов
28.09.2022

Структура курса

1. Введение в Большие Данные
2. Hadoop экосистема и MapReduce
3. SQL поверх больших данных ←
4. Инструменты визуализации при работе с Большими Данными
5. Введение в Scala
6. Модель вычислений Spark: RDD
7. Распараллеливание алгоритмов ML
8. Spark Pipelines
9. Approximate алгоритмы для больших данных
10. Spark для оптимизации гиперпараметров
11. Поточковая обработка данных (Kafka, Spark Streaming, Flink)
12. Архитектуры в продакшене

План занятия

1. Обзор фреймворков для SQL-подобной работы с Большими данными
2. Apache Hive

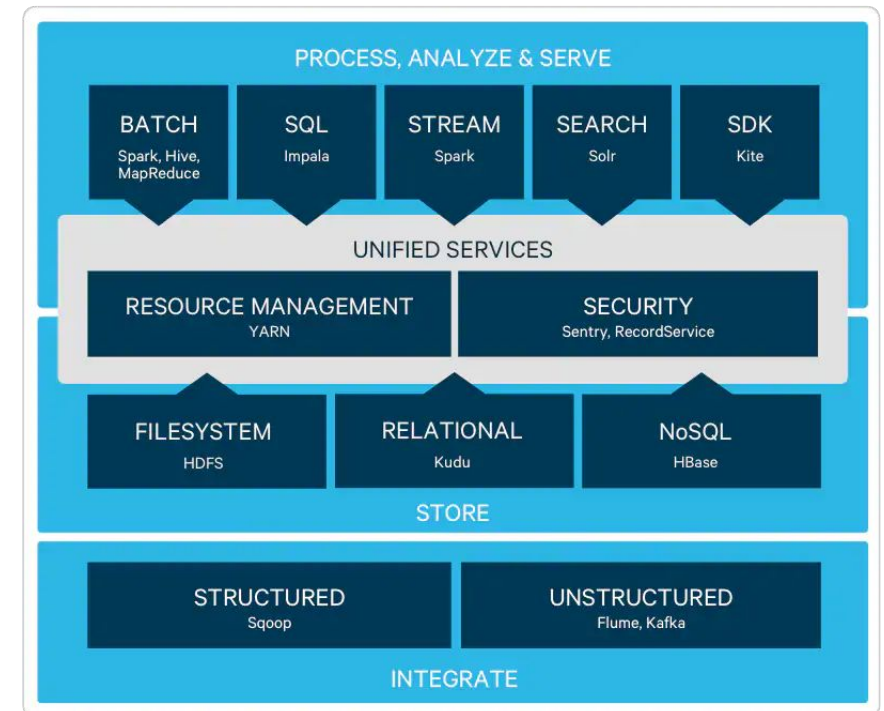
Where we are?


Big Data platform

- Hadoop Distributed Filesystem (NM, DN)
- Apache Hadoop YARN (RM, NM)

Big Data applications

- Hadoop MapReduce
- SQL-like processing frameworks ←
- Apache Spark
- Stream processing frameworks + Apache Kafka





Обзор фреймворков для SQL-подобной работы с Большими данными

SQL over Big Data. Motivation

- 1) Есть разные данные в разных форматах, которые хранятся в разных хранилищах и каждый из компонентов имеет свои интерфейсы.
- 2) Есть понятный стандартизированный SQL since 1974.
- 3) Хотели (хотят) сделать инструмент, который мог бы уметь под капотом работать с большим зоопарком и управляться через стандартизированные интерфейсы

SQL over Big Data. Landscape. On-premise



Distributed processing



MPP (Massively parallel processing)

Apache Impala

Impala обеспечивает быстрые интерактивные SQL-запросы, хранящихся в HDFS, HBase или Amazon S3. Состоит из следующих компонентов:

- Клиенты - **Hue**, ODBC, JDBC и Impala Shell
- Hive Metastore - хранит информацию о данных, доступных для Impala. Metastore позволяет Impala знать, какие базы данных доступны и какова структура этих баз данных.
- **Impalad** - процесс, который выполняется на DN, координирует и выполняет запросы. Каждый экземпляр Impala может получать, планировать и координировать запросы от клиентов Impala. Запросы распределяются между узлами Impala, и эти узлы действуют как рабочие, выполняя параллельные фрагменты запроса.
- HBase и HDFS - хранят данные для запроса.

Написана на C++/Java, очень быстрая. Общается с YARN через **Llama**.

Apache Impala. Architecture

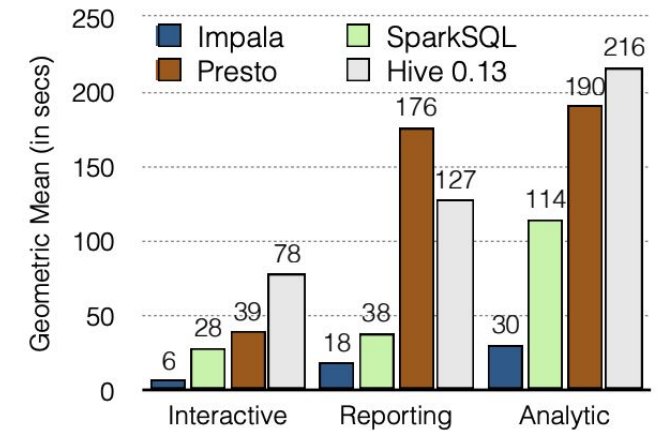
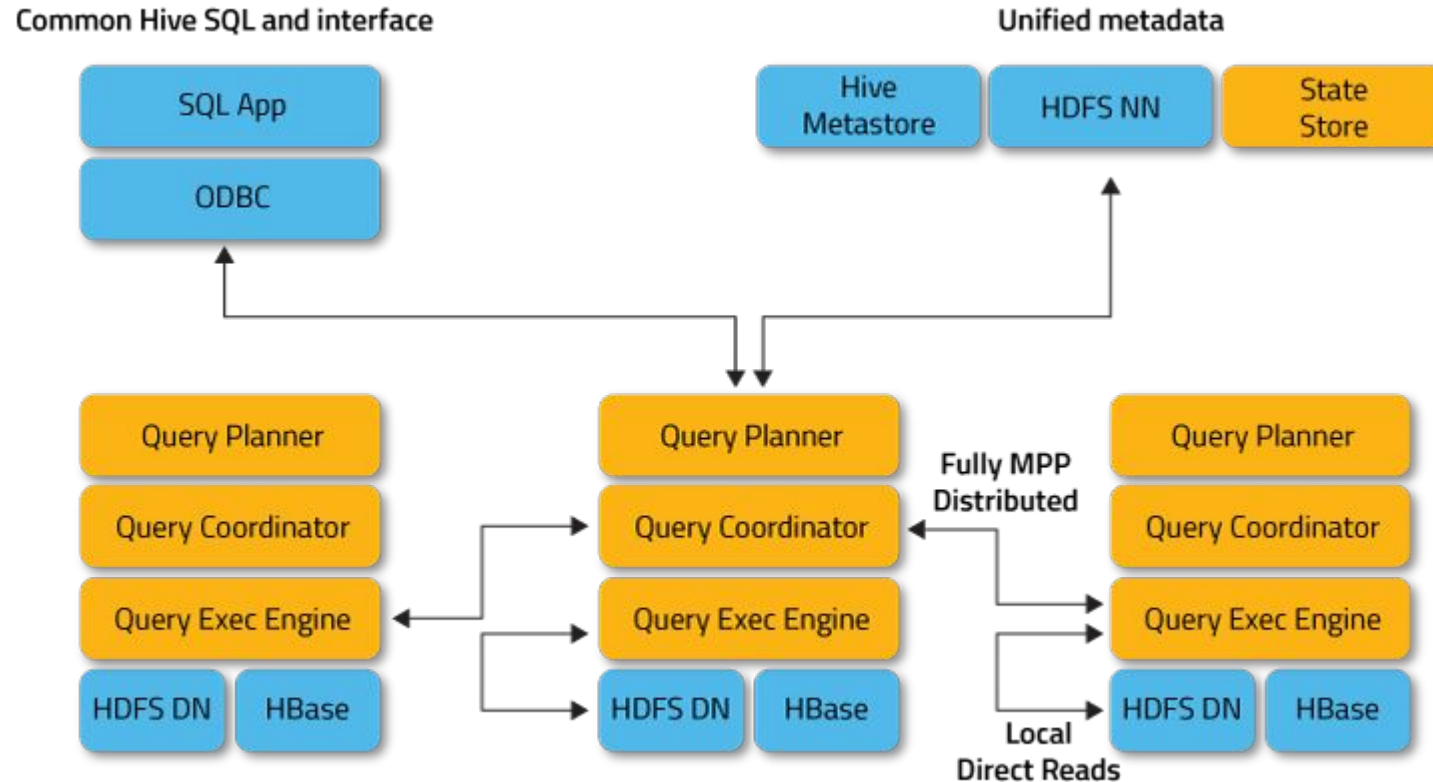


Figure 6: Comparison of query response times on single-user runs.

Apache Drill

Supports:

- ANSI SQL
- ODBC/JDBC
- RESTful APIs



Storages:

- Apache Hadoop, MapR, CDH and Amazon EMR
- NoSQL: MongoDB, Apache HBase, Apache Cassandra
- Online Analytical Processing: Apache Kudu, Apache Druid, OpenTSDB
- Cloud storage: Amazon S3, Google Cloud Storage, Azure Blob Storage, Swift, IBM Cloud Object Storage
- Diverse data formats, including Apache Avro, Apache Parquet and JSON
- RDBMs storage plugins (Using JDBC to connect to MySQL, PostgreSQL, and others)

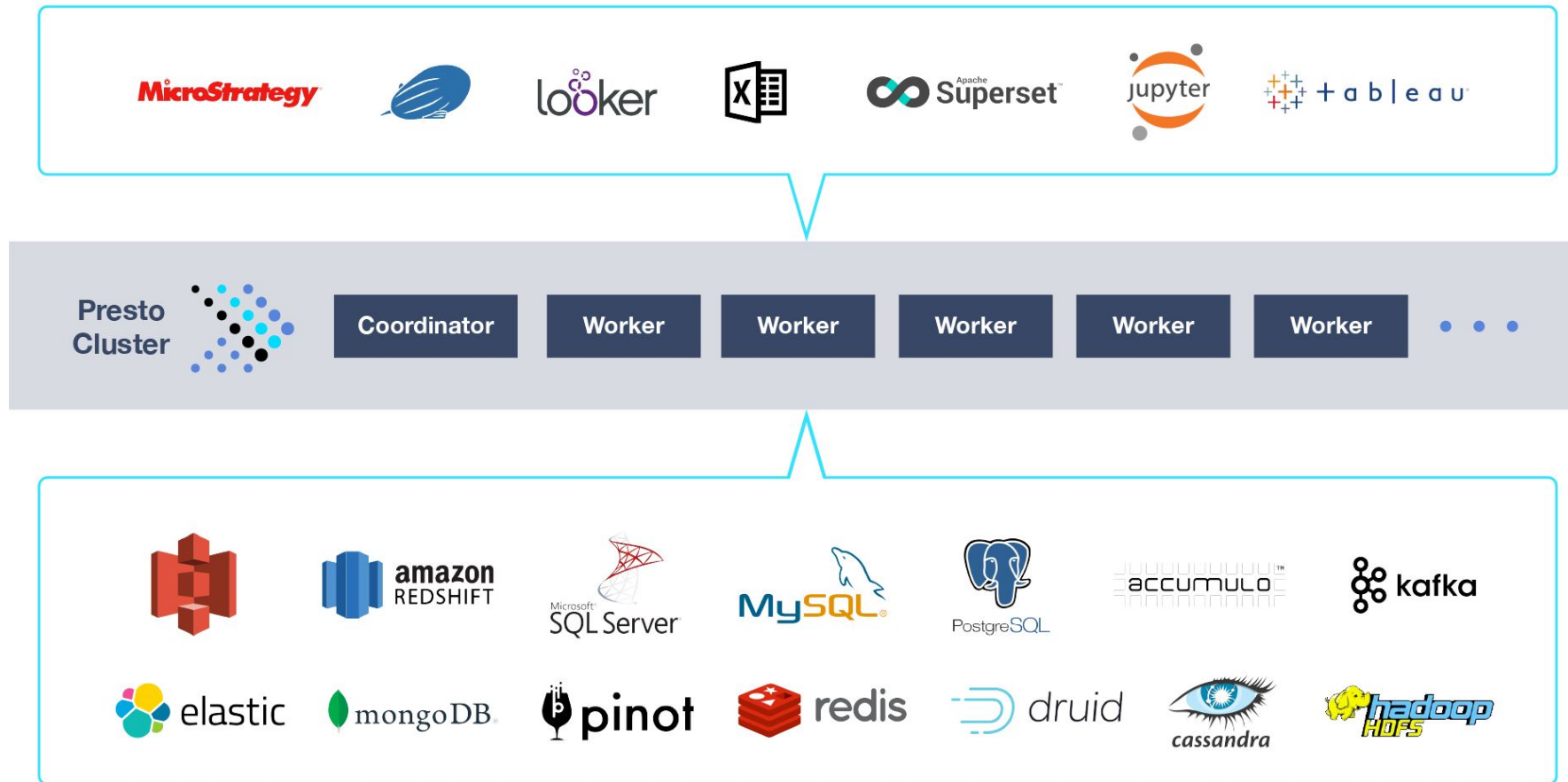
Apache Drill. Workflow

Drillbit - приложение, которое устанавливается на каждой ноде. Может доставляться через YARN, может работать с RM и даже имеет свой AM, но пока не рекомендуют его использовать.

Последовательность обработки запросов Apache Drill:

1. Клиент посылает запрос в Drill через JDBC, ODBC, CLI или REST API. Любая нода с Drillbit в кластере может его принять.
2. Drillbit парсит запрос, оптимизирует его, и генерирует план его распределенного выполнения.
3. Нода Drillbit, которая приняла запроса получает список доступных Drillbit нод из **ZooKeeper** и выбирает подходящие для соблюдения data locality.
4. Ведущая нода Drillbit рассылает запросы на исполнение.
5. Ведомые ноды исполняют запрос и отсылают ведущей ноде ответы.
6. Ведущая Drillbit нода отдает результат клиенту.

Presto / Trino. Architecture



Presto / Trino. Architecture

Coordinator - парсит запросы, планирует выполнение, управляет воркерами. Координаторы управляют воркерами через REST API.

Worker - исполняет задачи, забирает данные из источников через коннекторы. Отдает результат на координатор.

Connector - интерфейс взаимодействия с другими системами. Некоторые коннекторы, которые есть из коробки:

- | | |
|----------------------------|--------------------------|
| 1. Cassandra Connector | 8. Oracle Connector |
| 2. Druid Connector | 9. PostgreSQL Connector |
| 3. Elasticsearch Connector | 10. Redis Connector |
| 4. Hive Connector | 11. Redshift Connector |
| 5. Kafka Connector | 12. SQL Server Connector |
| 6. MongoDB Connector | 13. TPCDS Connector |
| 7. MySQL Connector | |



Presto. Use case



Ashish Singh

Tech Lead, Big Data Platform at Pinterest · Nov 27, 2019 | 33 upvotes · 534.2K views



Shared insights



Amazon EC2



Kubernetes



Presto



Apache Hive



Amazon S3



Pinterest

To provide employees with the critical need of interactive querying, we've worked with [Presto](#), an open-source distributed SQL query engine, over the years. Operating Presto at Pinterest's scale has involved resolving quite a few challenges like, supporting deeply nested and huge thrift schemas, slow/ bad worker detection and remediation, auto-scaling cluster, graceful cluster shutdown and impersonation support for ldap authenticator.

Our infrastructure is built on top of [Amazon EC2](#) and we leverage [Amazon S3](#) for storing our data. This separates compute and storage layers, and allows multiple compute clusters to share the S3 data.

We have hundreds of petabytes of data and tens of thousands of [Apache Hive](#) tables. Our Presto clusters are comprised of a fleet of 450 r4.xl EC2 instances. Presto clusters together have over 100 TBs of memory and 14K vcpu cores. Within Pinterest, we have close to more than 1,000 monthly active users (out of total 1,600+ Pinterest employees) using Presto, who run about 400K queries on these clusters per month.

Each query submitted to Presto cluster is logged to a [Kafka](#) topic via Singer. Singer is a logging agent built at Pinterest and we talked about it in a [previous post](#). Each query is logged when it is submitted and when it finishes. When a Presto cluster crashes, we will have query submitted events without corresponding query finished events. These events enable us to capture the effect of cluster crashes over time.

Each Presto cluster at Pinterest has workers on a mix of dedicated AWS EC2 instances and [Kubernetes](#) pods. Kubernetes platform provides us with the capability to add and remove workers from a Presto cluster very quickly. The best-case latency on bringing up a new worker on Kubernetes is less than a minute. However, when the Kubernetes cluster itself is out of resources and needs to scale up, it can take up to ten minutes. Some other advantages of deploying on Kubernetes platform is that our Presto deployment becomes agnostic of cloud vendor, instance types, OS, etc.

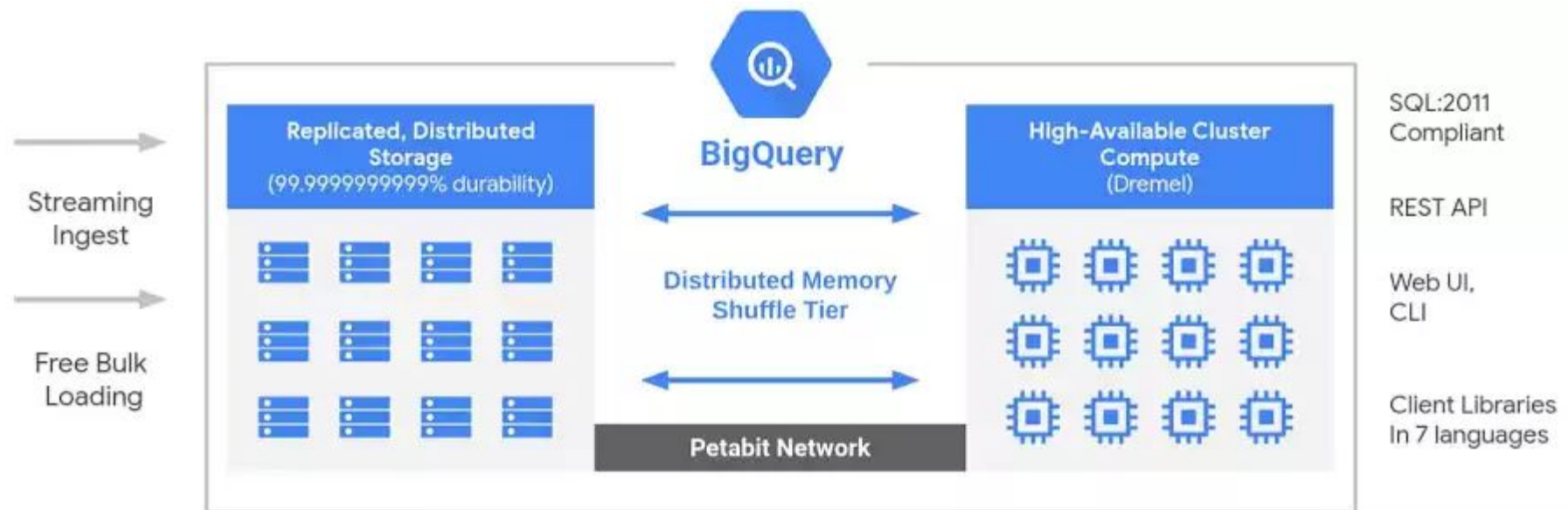
SQL over Big Data. Landscape. Cloud



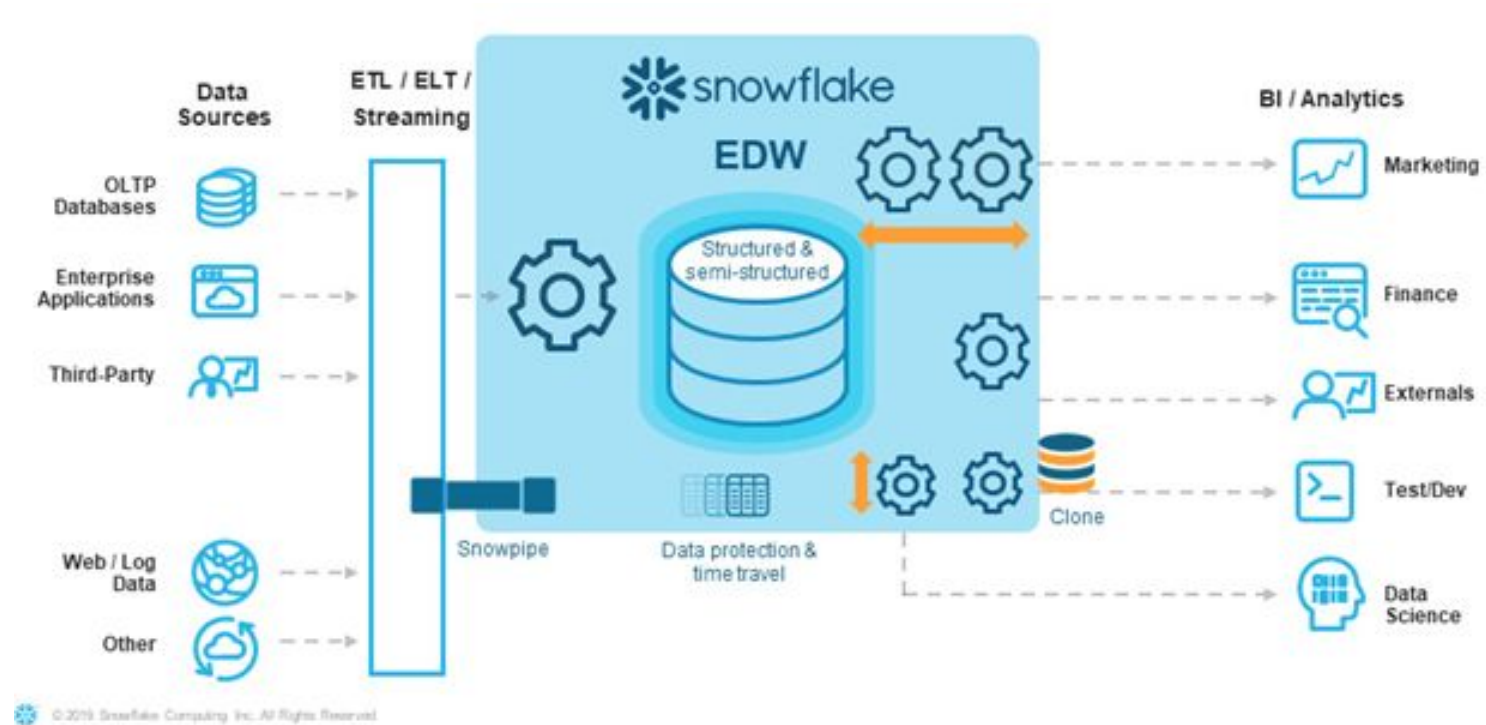
BigQuery



Bigquery



Snowflake



Cloud DWH providers comparison

CLOUD DATA WAREHOUSE PRODUCTS COMPARED					
Vendors	Snowflake	Redshift	BigQuery	Teradata	Azure
Architecture	Hybrid (shared-disk and shared-nothing elements)	Shared-nothing MPP architecture	Shared-nothing MPP architecture	Shared-nothing MPP architecture	Shared-disk MPP architecture
Server management	More serverless	More self-managed	Serverless	More self-managed	More self-managed
Deployment	Cloud-based	Cloud-based	Cloud-based	Cloud-based, On-premises	Cloud-based
Performance	High	Good	Good	High	High
Scalability	Scales horizontally and vertically				

Cloud DWH providers comparison

CLOUD DATA WAREHOUSE PRODUCTS COMPARED					
Vendors	Snowflake	Redshift	BigQuery	Teradata	Azure
Integrations	Data integration, BI, and analytics tools	AWS ecosystem, data integration, BI, and analytics tools	Google Workplace, data integration, BI and AI tools	Cloud providers, data integration, BI, and analytics tools	Microsoft software, data integration, BI, and ML tools
Data loading	ETL/ELT, data streaming support				
Data backup and recovery	Yes				
Implementation	Intuitive and simple-to-use. Requires solid SQL and DW architecture knowledge	Knowing PostgreSQL or similar RDMSs facilitates deployment	User-friendly. Requires knowledge of SQL commands and ETL tools	Easy and fast. Requires a background in using SQL syntax and working with RDBMs	Easy-to-use. Requires SQL and Spark use experience
Pricing	On-demand, pre-purchase	On-demand, managed storage	Flat rate, on-demand	Blended, on-demand	Compute charge, storage charge
Suitable for those who	Need easy deployment and configuration	Process large data sets	Deal with varied workloads	Look for flexible deployment	Need enterprise DWHs



Apache Hive

Hive

Hive - движок, который превращает SQL-запросы в джобы. ACID compatible!

Хорош для обработки больших запросов и ETL, плох для OLTP как и все подобные системы.

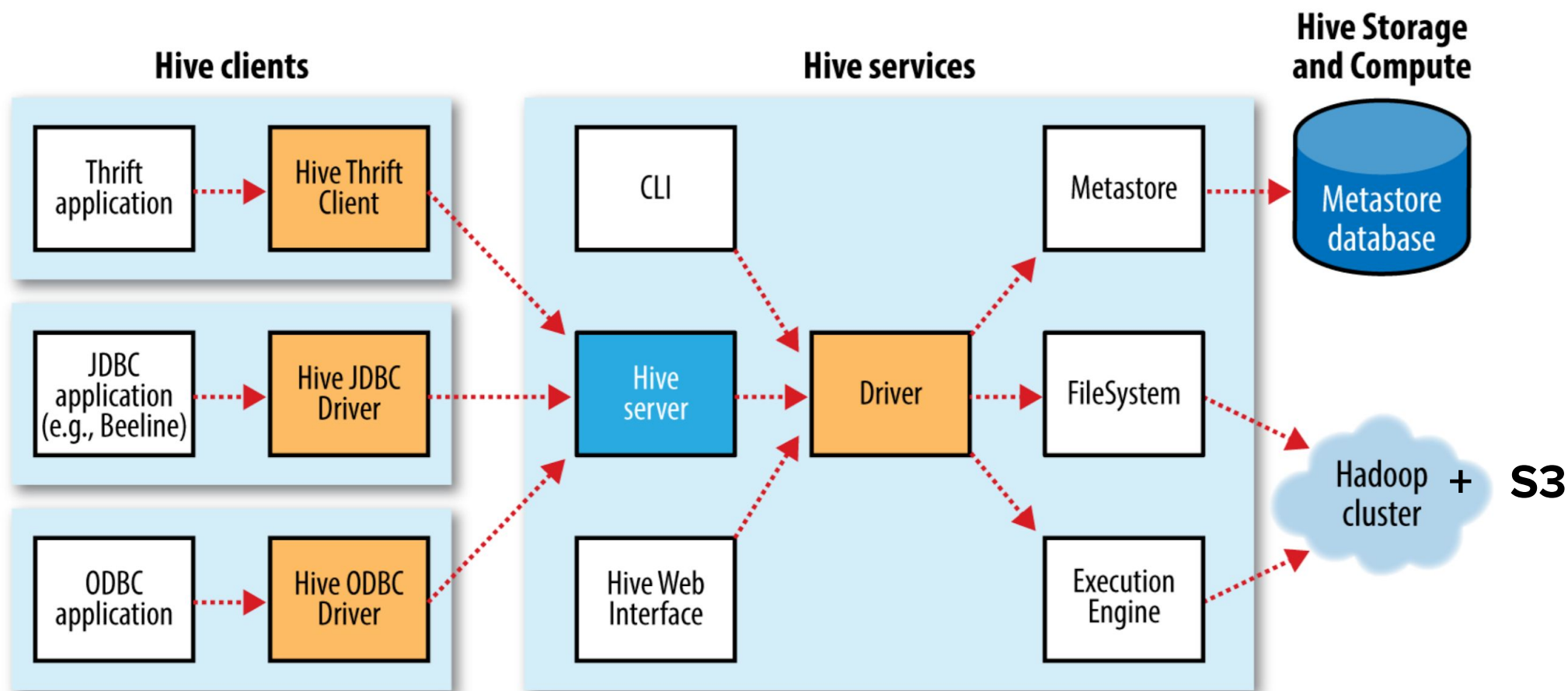
Движок включает в себя:

1. **Server** - универсальный коннектор
2. **Driver** - разбирает входящие SQL-запросы
3. **Execution Engine** - запускает задачи (**MR / Tez / Spark**)

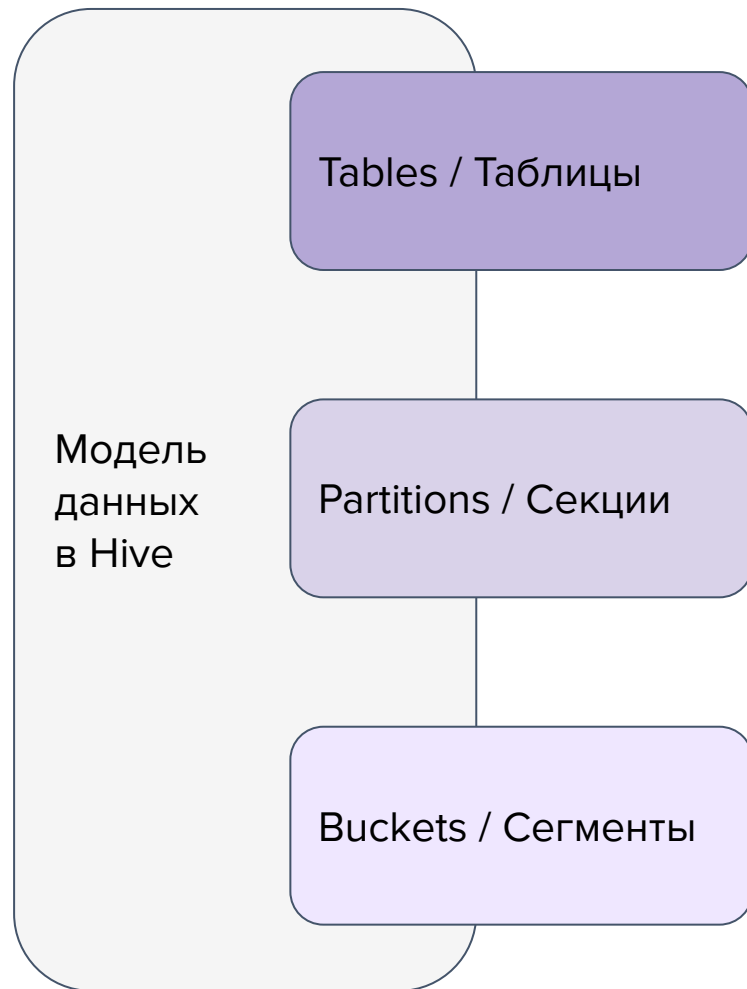
Hive использует хранилище метаданных **Metastore** для матчинга сущностей SQL (база данных, таблица, колонки, строки, ячейки) с объектами, хранящимися в HDFS или S3.

В качестве **Metastore** можно использовать RDBMS вроде MySQL, PostgreSQL или Oracle.

Hive. Architecture



Data model



Таблицы создаются таким же образом, как в классических реляционных базах данных. Являются физическим представлением данных, хранящихся в директории внутри файловой системы:

`/user/hive/warehouse/mytable`

`/user/hive/warehouse/mytable/City=Moscow`

`/user/hive/warehouse/mytable/City=Moscow/part-00000`

Hive tricks. Partitioning & Bucketing

1. Разбиение на **партиции** дает возможность делать запросы быстрее.
2. Когда партиций мало, или есть сильный дисбаланс партиций делается **бакетизация**

```
/user/hive/warehouse/logs
├── dt=2001-01-01/
│   ├── country=GB/
│   │   ├── file1
│   │   └── file2
│   └── country=US/
│       └── file3
└── dt=2001-01-02/
    ├── country=GB/
    │   └── file4
    └── country=US/
        ├── file5
        └── file6
```

HiveQL

Hive Query Language (HiveQL) - диалект SQL с рядом отличий

Сложные обработки, которые нельзя написать на SQL можно реализовать через написание своих функций: User Defined Function (UDF), User Defined Aggregate Function (UDAF), User Defined Tabular Function (UDTF)

```
hive> SHOW FUNCTIONS;
```

```
hive> DESCRIBE FUNCTION length;
```

```
length (str | binary) - Returns the length of  
str or number of bytes in binary data
```

Feature	SQL	HiveQL
Updates	UPDATE, INSERT, DELETE	UPDATE, INSERT, DELETE
Transactions	Supported	Limited support
Indexes	Supported	Supported
Data types	Integral, floating-point, fixed-point, text and binary strings, temporal	Boolean, integral, floating-point, fixed-point, text and binary strings, temporal, array, map, struct
Functions	Hundreds of built-in functions	Hundreds of built-in functions
Multitable inserts	Not supported	Supported
CREATE TABLE...AS SELECT	Not valid SQL-92, but found in some databases	Supported
SELECT	SQL-92	SQL-92. SORT BY for partial ordering, LIMIT to limit number of rows returned
Joins	SQL-92, or variants (join tables in the FROM clause, join condition in the WHERE clause)	Inner joins, outer joins, semi joins, map joins, cross joins
Subqueries	In any clause (correlated or noncorrelated)	In the FROM, WHERE, or HAVING clauses (uncorrelated subqueries not supported)
Views	Updatable (materialized or nonmaterialized)	Read-only (materialized views not supported)

Hive. Commands

DDL Command	Use With
CREATE	Database, Table
SHOW	Databases, Tables, Table Properties, Partitions, Functions, Index
DESCRIBE	Database, Table, view
USE	Database
DROP	Database, Table
ALTER	Database, Table
TRUNCATE	Table

DML Command
LOAD
SELECT
INSERT
DELETE
UPDATE
EXPORT
IMPORT

DDL

```
CREATE TABLE mytable(id INT, name STRING, age INT, city
```

объявляем схему

```
STRING)
```

```
COMMENT `This is a sample table`
```

комментарий для читаемости

```
PARTITIONED BY (city STRING)
```

партиционирование

```
ROW FORMAT DELIMITED
```

строки разделяются `n`

```
FIELDS TERMINATED BY ``,`
```

поля разделяются запятой

```
STORED AS TEXTFILE;
```

хранится в виде текстового файла

Таблица создается в специальной директории warehouse и полностью находится под управлением Hive

DDL

```
CREATE EXTERNAL TABLE my_external_table(id INT, name STRING, age INT,  
city STRING)  
LOCATION `/user/ivanov/mytable`;
```

Таблица не создается в warehouse. У Hive сохраняется только ссылка на неё

DDL

```
CREATE TABLE mytable(id INT, name STRING, age INT, city
```

объявляем схему

```
STRING)
```

```
COMMENT `This is a sample table`
```

комментарий для читаемости

```
PARTITIONED BY (city STRING)
```

партиционирование

```
ROW FORMAT DELIMITED
```

строки разделяются `n`

```
FIELDS TERMINATED BY ``,`
```

поля разделяются запятой

```
STORED AS TEXTFILE;
```

хранится в виде текстового файла

Таблица создается в специальной директории warehouse и полностью находится под управлением Hive

DML

```
LOAD DATA LOCAL INPATH `/home/ivanov/peoples.txt`  
INTO TABLE mytable;
```

Файл peoples.txt находится в локальной файловой системе и будет скопирован в директорию warehouse

```
LOAD DATA INPATH `/user/ivanov/peoples.txt`  
INTO TABLE mytable;
```

Файл peoples.txt находится в HDFS и будет скопирован в директорию warehouse

DML

```
SELECT * FROM mytable;
```

Показать все данные таблицы

```
SELECT COUNT(DISTINCT city ) FROM mytable;
```

Агрегация

```
SELECT COUNT(*) FROM mytable GROUP BY city;
```

```
SELECT * FROM mytable SORT BY id DESC;
```

```
FROM mytable SELECT * ORDER BY id ASC;
```

Агрегация и сортировка

Apache Hue

The screenshot displays the Apache Hue web interface. On the left is a dark sidebar with navigation links: Editor, Dashboard, Scheduler, Documents, Files, S3, Tables, Indexes, Jobs, Streams, HBase, Security, and Importer. The main area is divided into three panels. The left panel shows a 'Tables' list for the 'default' database, including 'customers', 'k8s_logs', 'sample_07', 'sample_08', and 'web_logs'. The center panel is the SQL editor, showing a query that filters for shipping orders from zip code 76710 and calculates the total amount per customer. The right panel shows the 'Tables' section for the 'default' database, listing 'customers' with columns 'id', 'name', 'email_preferences', 'addresses', and 'orders'. Below the query editor, a 'Query History' section shows three completed queries. At the bottom, a 'Results (106)' table displays the output of the query, showing customer details and order totals.

Query

Search saved documents...

Impala Add a name... Add a description... 0.92s Database default

```
15 WHERE a.key = 'shipping' and a.zip_code = '76710';
16
17
18
19 -- Compute total amount per order for all customers
20 SELECT
21   c.id AS customer_id,
22   c.name AS customer_name,
23   o.order_id,
24   v.total
25 FROM
26   customers c,
27   c.orders o,
28   (SELECT SUM(price * qty) total FROM o.items) v;
```

Tables Statement 3/3

Filter...

default.customers

id	int
name	string
email_preferences	struct
addresses	map
orders	array

Query 034d413ec7474ed5:4249de1000000000 100% Complete 034d413ec7474ed5:4249de1000000000
Query 034d413ec7474ed5:4249de1000000000 100% Complete 034d413ec7474ed5:4249de1000000000
Query 034d413ec7474ed5:4249de1000000000 100% Complete (1 out of 1)

Query History **Saved Queries** **Results (106)** **Execution Analysis**

	customer_id	customer_name	order_id	total
1	75012	Dorothy Wilk	4056711	918
2	75012	Dorothy Wilk	J882C2	96
3	17254	Martin Johnson	I72T39	18
4	12532	Melvin Garcia	PB6268	68
5	12532	Melvin Garcia	B8623C	2507
6	12532	Melvin Garcia	R9S838	1278
7	42632	Raymond S. Vestal	HS3124	1944
8	42632	Raymond S. Vestal	BS5902	2798
9	77913	Betty J. Giambrone	DN8815	1320
10	77913	Betty J. Giambrone	XR2771	4315

Frameworks comparison

Open Source SQL on Hadoop Solutions								
What are Your Needs?	 High Performance for Predefined Queries	✗	✓	✓	✓	✓	✓	✓
	 High Performance for Dynamic/Interactive Queries	✗	✗	✗	✗	✗	✗	✗
	 Support for All Hadoop Distros	✓	✓	✓	✓	✓	✓	✓
	 Support for HDFS/HBase Storage	✓	✓	✓	✓	✓	✓	✓
	 Support for S3 Storage	✗	✓	✓	✓	✓	✓	✗
	 Support for WASB Storage	✗	✗	✓	✗	✗	✗	✗
	 Support for other Storage Systems	Hypertable	Cassandra	✗	Cassandra	RDBs / Redis	Kudu / GoogleCS / MongoDB	✗
	 Optimized Storage Formats	RC / ORC	SEQ / Parquet	ORC	SEQ / Parquet	ORC	JSON / Parquet	Parquet
	 HiveQL Support	✓	✓	✓	✗	✗	✗	✗
	 SQL Support	SQL-92	SQL-92	SQL-92	SQL-92 Subset	SQL-92	ANSI	SQL-99
Common Use Cases		Integration with BI tools Not performance sensitive	Need high performance Multiple data streams	High performance Complex distributed processing	High performance Multiple data streams	Very large data scale Interactive analytics	Both schema-declared and schema-on-read Arbitrary formats	Greenplum user Advanced analytics with MADlib

HIVE vs RDBMS

Apache Hive

- Проверяет схему данных при **чтении**, что ускоряет *добавление* данных и позволяет работать с *неизвестной* схемой данных
- Размер данных измеряется петабайтами
- Ориентирован на модель “один раз записал, много раз прочитал”
- Несмотря на поддержку SQL является хранилищем данных
- Легко масштабируется при расширении кластера

RDBMS

- Проверяет схему при **записи**, что ускоряет *чтение* данных и повышает уровень их *консистентности*
- Размер данных измеряется терабайтами
- Поддерживается инструментарий как записи данных, так и чтения
- Традиционная база данных, основанная на реляционной модели данных
- Для масштабирования требуются соответствующие навыки

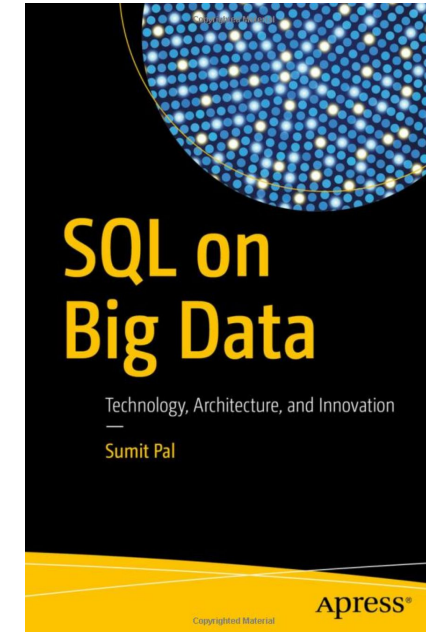
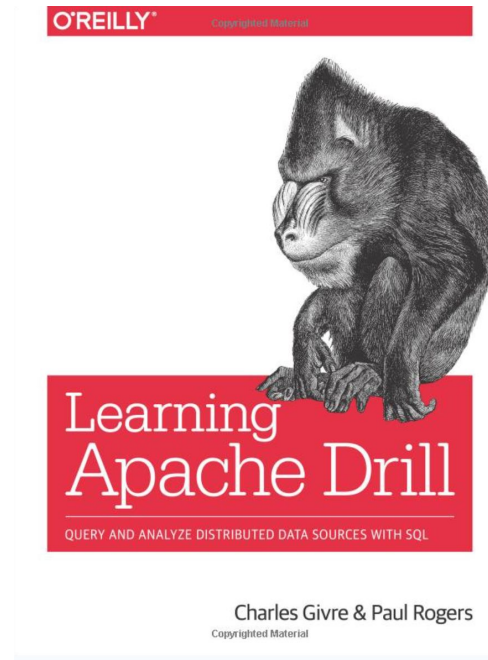
Additional topics

1. Форматы хранения данных Orc, Avro, Parquet
2. Безопасность и разделение прав
3. Тюнинг запросов

Lecture summary

1. SQL на больших данных нужен и возможен в двух парадигмах MPP и DP.
2. Impala, Drill, Presto - MPP фреймворки.
3. Hive, SparkSQL - DP фреймворки
4. Hive с точки зрения пользователя похож на MySQL, но со своими особенностями.

Recommended links and literature



- 1) [OLAP Query Engines for Big Data](#)
- 2) [Big Data File Formats](#)
- 3) <https://www.sql-ex.ru/>