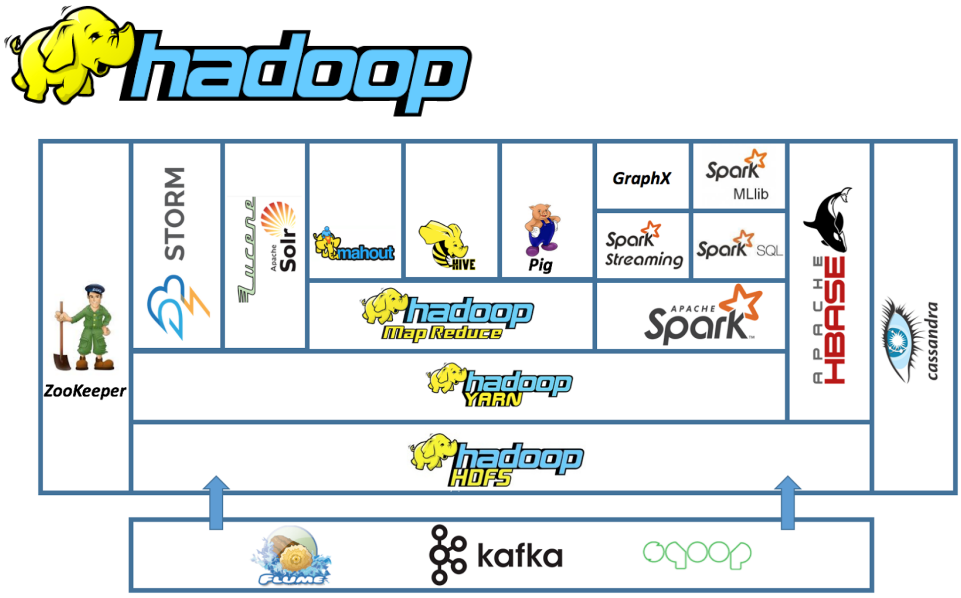
Duże zbiory danych
big data

dr hab. inż. Przemysław Korytkowski

Wykład 3

1



2

- Stworzony przez Facebook w styczniu 2007 roku, od sierpnia 2008 open source.
- HIVE jest zbliżonym do SQL-u interfejsem wykorzystującym Hadoop MapReduce, aby uwolnić użytkowników od zajmowania się niskopoziomowymi szczegółami implementacji równoległych zadań przetwarzania wsadowego.
- Hive koncentruje się głównie na ekstrakcji - transformacji - ładowaniu (ETL) oraz przetwarzaniu wsadowym (w partiach):
  - odczytywanie ogromnych ilości danych,
  - dokonywanie przekształceń tych danych (np. przemieszanie danych, konsolidacja, agregowanie)
  - załadowanie danych wyjściowych do innych systemów, które są wykorzystywane do dalszej analizy.

3

Hive – A Petabyte Scale Data Warehouse Using Hadoop

Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy

Facebook Data Infrastructure Team

2010

- Hive, an open-source data warehousing solution built on top of Hadoop.
- Hive supports queries expressed in a SQL-like declarative language - **HiveQL**, which are compiled into mapreduce jobs that are executed using Hadoop. In addition, HiveQL enables users to plug in custom map-reduce scripts into queries.
- The language includes a type system with support for tables containing primitive types, collections like arrays and maps, and nested compositions of the same. The underlying IO libraries can be extended to query data in custom formats.
- Hive also includes a system catalog - Metastore – that contains schemas and statistics, which are useful in data exploration, query optimization and query compilation.
- In Facebook, the Hive warehouse contains tens of thousands of tables and stores over 700TB of data and is being used extensively for both reporting and ad-hoc analyses by more than 200 users per month.

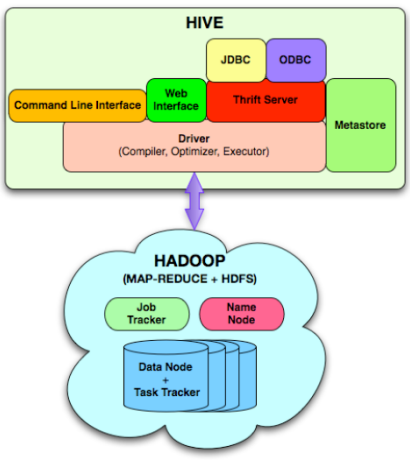
4

Możliwości

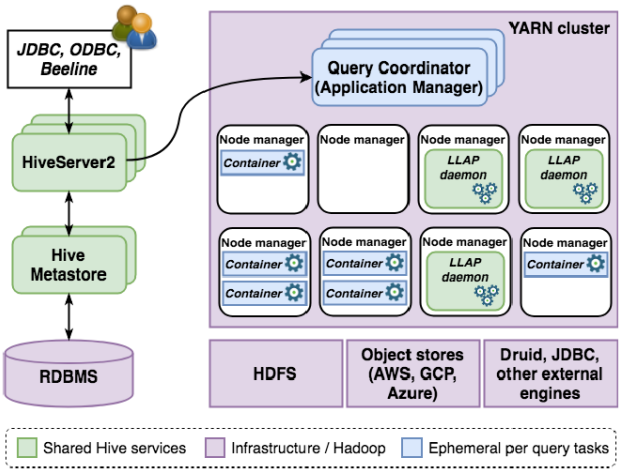
SQL Data types	SQL Semantics
INT	SELECT, LOAD, INSERT from query
TINYINT/SMALLINT/BIGINT	Expressions in WHERE and HAVING
BOOLEAN	GROUP BY, ORDER BY, SORT BY
FLOAT	CLUSTER BY, DISTRIBUTE BY
DOUBLE	Subqueries in FROM clause
STRING	GROUP BY, ORDER BY
BINARY	ROLLUP and CUBE
TIMESTAMP	UNION
ARRAY, MAP, STRUCT, UNION	LEFT, RIGHT and FULL INNER/OUTER JOIN
DECIMAL	CROSS JOIN, LEFT SEMI JOIN
CHAR	Windowing functions (OVER, RANK, etc.)
VARCHAR	Subqueries for IN/NOT IN, HAVING
DATE	EXISTS / NOT EXISTS

5

Architektura



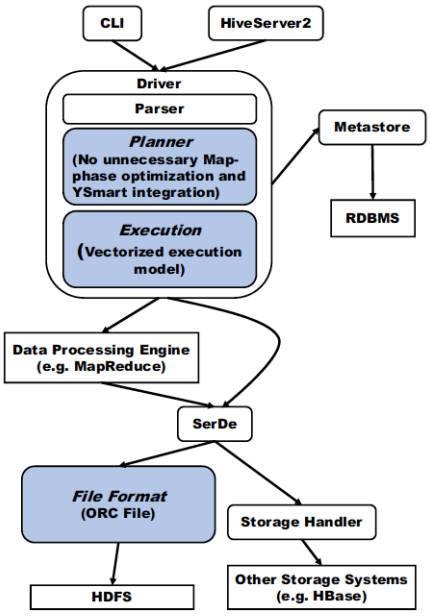
2010



2019

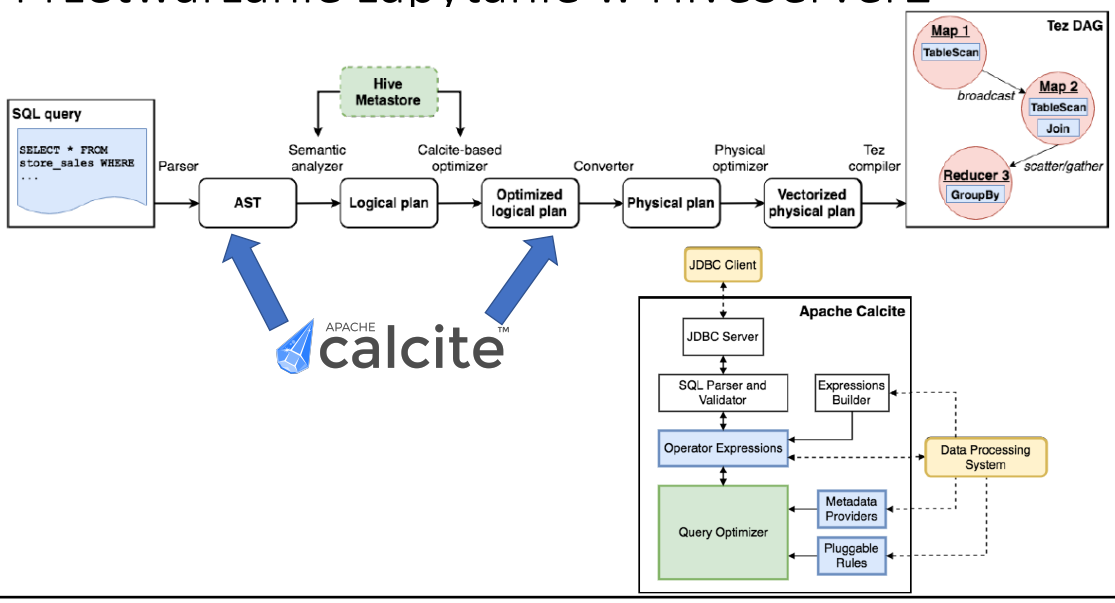
6

Przetwarzanie zapytań



7

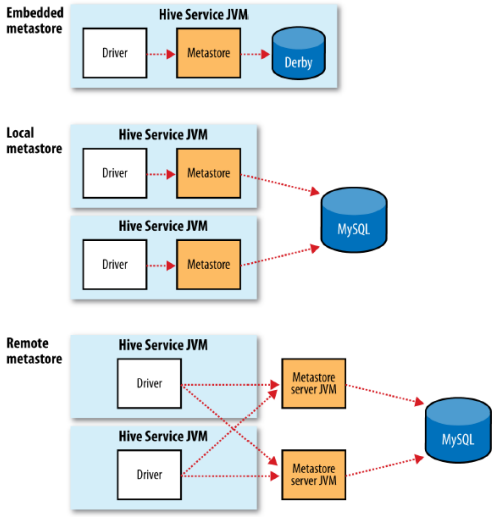
Przetwarzanie zapytanie w HiveServer2



8

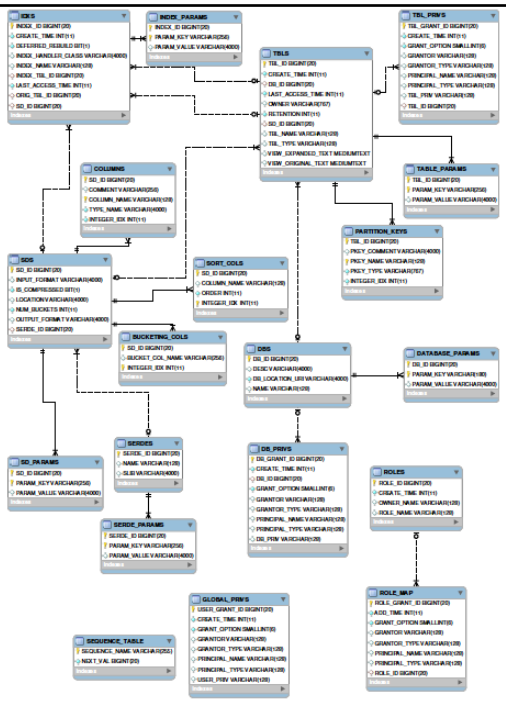
MetaStore

- Metastore jest centralnym repozytorium metadanych Hive. Przechowuje ono metadane dla tabel (takie jak ich schemat i lokalizacja) oraz partycje.
- Wszystkie implementacje Hive'a potrzebują usługi metastore.
- Jest on zaimplementowany przy użyciu tabel w relacyjnej bazie danych.
- Domyślnie Hive korzysta z wbudowanego serwera Derby SQL. Zapewnia on obsługę jednego procesu, nie możemy uruchomić kilku instancji Hive CLI.
- Kiedy chcemy uruchomić Hive w klastrze, to MySQL lub inna podobna relacyjna baza danych jest wymagana.
- Obecnie MetaStore może działać na Hbase



9

MetaStore



10

Tabele

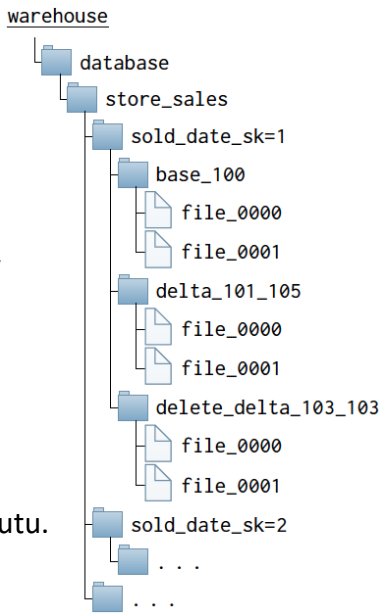
- Zarządzanie (ang. **managed**), Hive kontroluje cykl życia ich danych, przechowuje dane dla tych tabel w podkatalogu zdefiniowanym przez hive.metastore.warehouse.dir (np. /user/hive/warehouse). Gdy skasujemy (`DROP`) zarządzaną tabelę, Hive usunie dane w tabeli (pliku).
- Zewnętrzne (ang. **external**), Hive zakłada, że nie jest właścicielem danych. Dlatego też, skasowanie (`DROP`) tabeli nie powoduje usunięcia danych, chociaż metadane dla tabeli zostaną usunięte.

```
CREATE EXTERNAL TABLE IF NOT EXISITS table_name (…)
LOCATION `/user/XXXX/file_name.orc`
```

11

Struktura plików

- /user/hive/warehouse
- **Tabela** jest przechowywana w katalogu w HDFS.
- **Partycja** tabeli przechowywana jest w podkatalogu w katalogu tabeli. Polega na podziale na partycje wg wartości atrybutu. Najważniejszym powodem do partycjonowania danych jest szybsze wykonywanie zapytań.
- **Kubełek** (ang. bucket) przechowywany jest w pliku w katalogu partycji lub tabeli w zależności od tego, czy tabela jest tabelą partycjonowaną, czy nie. Polega na podziale na kubełki dzięki haszowaniu wartości atrybutu.
- Kompaktowanie



12

CREATE TABLE

```
CREATE TABLE IF NOT EXISTS mydb.employees (
  name         STRING COMMENT 'Employee name',
  salary       FLOAT  COMMENT 'Employee salary',
  subordinates ARRAY<STRING> COMMENT 'Names of subordinates',
  deductions   MAP<STRING, FLOAT>
               COMMENT 'Keys are deductions names, values are percentages',
  address      STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
               COMMENT 'Home address')
COMMENT 'Description of the table'
TBLPROPERTIES ('creator'='me', 'created_at'='2012-01-02 10:00:00', ...)
LOCATION '/user/hive/warehouse/mydb.db/employees';
```

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

```
hive> CREATE TABLE weblog (user_id INT, url STRING, source_ip STRING)
    > PARTITIONED BY (dt STRING)
    > CLUSTERED BY (user_id) INTO 96 BUCKETS;
```

13

Wgrywanie danych

```
CREATE TABLE managed_table (dummy STRING);
LOAD DATA LOCAL INPATH '/user/tom/data.txt' OVERWRITE INTO TABLE
managed_table;
```

```
CREATE EXTERNAL TABLE external_table (dummy STRING)
LOCATION '/user/tom/external_table';
LOAD DATA INPATH '/user/tom/data.txt' INTO TABLE external_table;
```

- LOCAL – kopiuje dane.
- bez LOCAL przenosi, kasuje dane w źródle.

14

Wgrywanie danych

```
INSERT OVERWRITE INTO TABLE managed_table
PARTITION (xx, yy)
SELECT * FROM my_table_name WHERE y >= 5
```

```
CREATE TABLE ca_employees
AS SELECT name, salary, address
FROM employees
WHERE se.state = 'CA';
```

15

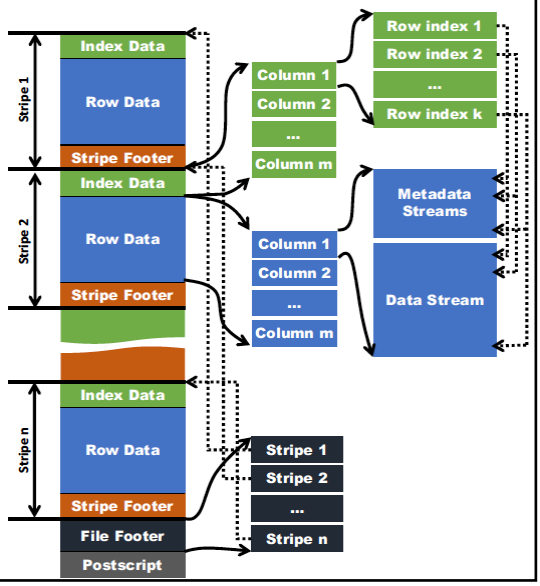
Przechowywanie danych w Hive

- Formaty plików: Text File, SequenceFile, RCFile, Avro, ORC, Parquet
- Formaty plików nieuwzględniające struktury danych oraz jednowierszowa serializacja uniemożliwiają efektywną kompresję wartości danych.
- Efektywność odczytu danych jest ograniczona przez brak indeksów i niezdekomponowanych kolumn o złożonych typach danych.

16

Plik ORC (Optimized Record Columnar)

- W przypadku tabeli przechowywanej w pliku ORC, jest ona najpierw dzielona poziomo na wiele segmentów. Następnie, w segmencie, wartości danych zapisywane są w kolumnach, jedna po drugiej.
- Wszystkie kolumny w segmencie są zapisywane w tym samym pliku. Ponadto, aby dostosować się do różnych wzorców zapytań, zwłaszcza zapytań ad hoc, plik ORC nie umieszcza kolumn w grupach kolumn.
- Domyślny rozmiar segmentu to 256 MB.
- Indeksy: wskaźniki pozycji i statystyki

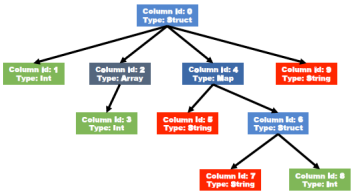


Plik ORC (Optimized Record Columnar)

- W przypadku kolumny o złożonym typie danych (np. Mapa), rozkłada się tę kolumnę na wiele kolumn podrzędnych.
- Można wybrać opcję dostosowania wielkości segmentu z wielkością blokiem HDFS. Jeden segment zawsze będzie w jednym bloku HDFS.
- Kompresja:
  - na poziomie kolumn, np. string można kodować słownikowo
  - cały plik: ZLIB, Snappy and LZO.

```
CREATE TABLE tbl (
  col1 Int,
  col2 Array<Int>,
  col4 Map<String,
            Struct<col7:String,
                   col8:Int>>,
  col9 String
)
```

(a) The schema of the table



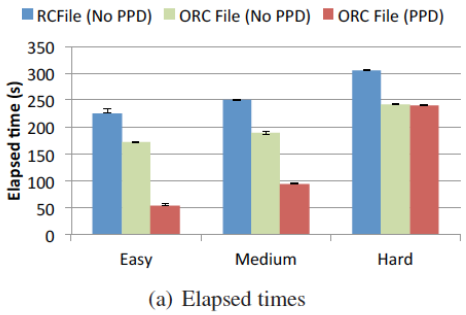
Statystyka danych w ORC

- Statystyka danych jest wykorzystywana przez ORC reader, aby nie odczytywać niepotrzebnie danych z HDFS.
- Te statystyki są tworzone, gdy ORC writer tworzy plik ORC.
- Statystyki obejmują: liczba wartości, min, max, suma, oraz długość (dla typów tekstu i typu binarnego). Dla złożonych typy danych, tablica, mapa, struktura i związek, ich kolumny potomne rejestrują również statystyki danych.
- W pliku ORC, statystyki danych mają trzy poziomy:
  1. Kolumny w pliku ORC Plik ORC posiada statystyki na poziomie plików, które są rejestrowane na końcu tego pliku. Statystyki te są wykorzystywane w optymalizacjach zapytań, oraz są również wykorzystywane do odpowiedzi na proste pytania dotyczące agregacji.
  2. ORC Plik przechowuje statystyki poziomu segmentu dla każdej kolumny. ORC reader używa te statystyki do analizy, które segmenty są potrzebne do wykonania zapytania. Niepotrzebne segmenty nie będą odczytywane z HDFS.
  3. statystyki drobnoziarniste wewnątrz segmentów, domyślnie dla 10 000 rekordów.

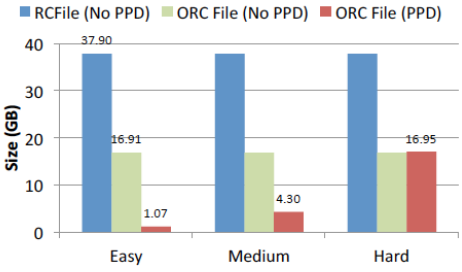
19

Porównanie RCFile z ORC

- PPD – z wykorzystaniem statystyki danych

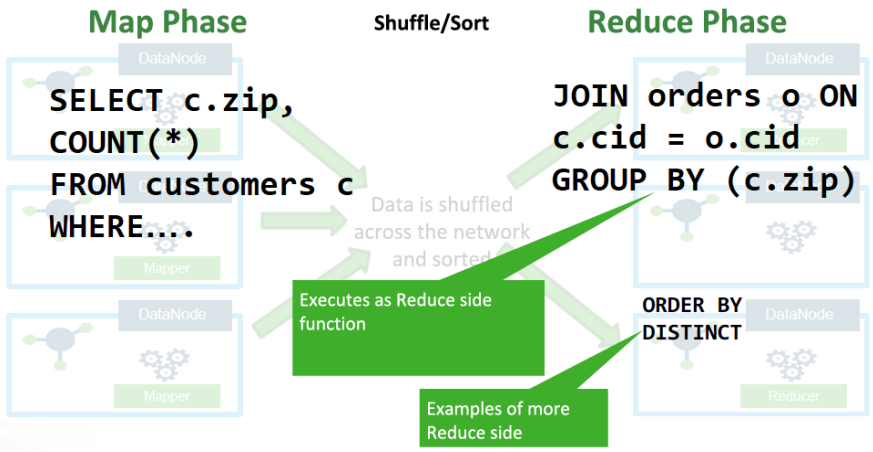


(a) Elapsed times



20

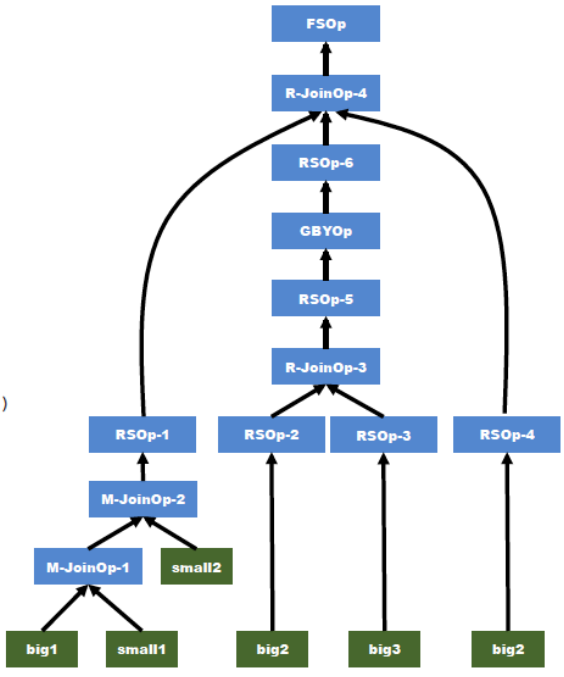
Hive jako MapReduce



21

Optymalizacja zapytań

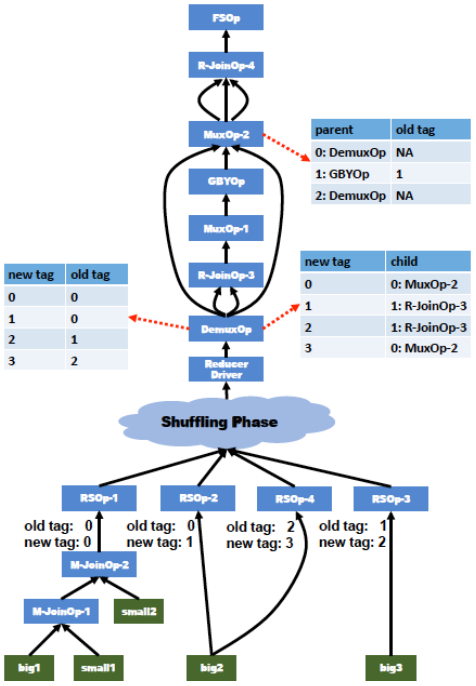
```
SELECT big1.key, small1.value1, small2.value1,
       big2.value1, sq1.total
FROM big1
JOIN small1 ON (big1.sKey1 = small1.key)
JOIN small2 ON (big1.sKey2 = small2.key)
JOIN (SELECT key,
             avg(big3.value1) AS avg,
             sum(big3.value2) AS total
      FROM big2 JOIN big3 ON (big2.key = big3.key)
      GROUP BY big2.key) sq1 ON (big1.key = sq1.key)
JOIN big2 ON (sq1.key = big2.key)
WHERE big2.value1 > sq1.avg;
```



22

Optymalizacja zapytań

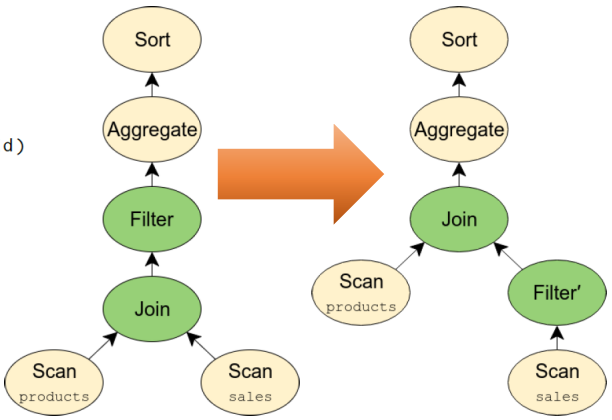
```
SELECT big1.key, small1.value1, small2.value1,
       big2.value1, sq1.total
FROM big1
JOIN small1 ON (big1.sKey1 = small1.key)
JOIN small2 ON (big1.sKey2 = small2.key)
JOIN (SELECT key,
             avg(big3.value1) AS avg,
             sum(big3.value2) AS total
      FROM big2 JOIN big3 ON (big2.key = big3.key)
      GROUP BY big2.key) sq1 ON (big1.key = sq1.key)
JOIN big2 ON (sq1.key = big2.key)
WHERE big2.value1 > sq1.avg;
```



23

Optymalizacja zapytań (Calcite)

```
SELECT products.name, COUNT(*)
FROM sales JOIN products USING (productId)
WHERE sales.discount IS NOT NULL
GROUP BY products.name
ORDER BY COUNT(*) DESC;
```



24

Cost Base Optimizer (CBO)

- Calcite oferuje dwa różne silniki planistyczne:
  - planista oparty na kosztach, który uruchamia przepisywanie reguł w celu zmniejszenia całkowitego kosztu wyrażenia
  - planista wyczerpujący, który uruchamia reguły w sposób wyczerpujący do momentu wygenerowania wyrażenia, które nie jest już modyfikowane przez żadne reguły. Reguły transformacji działają w sposób niezauważalny dla obu planistów.
- Calcite wykorzystuje statystyki udostępniana przez MetaStore oraz przechowywane w plikach ORC.

25

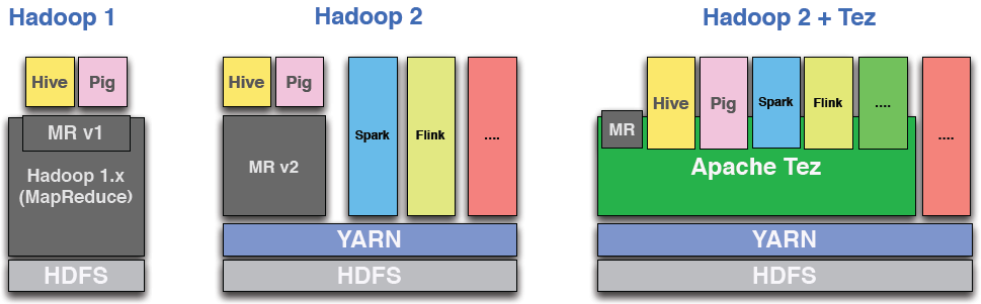
Wektoryzacja

- W wektorowym modelu wykonania, zestaw danych jest reprezentowany jako partia wierszy.
- W partii wierszy wartości danych danej kolumny są reprezentowane jako wektory kolumn. Ilość wierszy w partii jest konfigurowalna i powinna być wybrana tak, aby pasowała do wielkości pamięci podręcznej procesora.
- Domyślnie ta liczba jest ustawiona na 1024, co zazwyczaj pozwala zmieścić jeden wektor w pamięci podręcznej procesora.
- Wykonywanie zapytań postępuje poprzez zastosowanie każdego wyrażenia na całym wektorze kolumny.

26

Tez

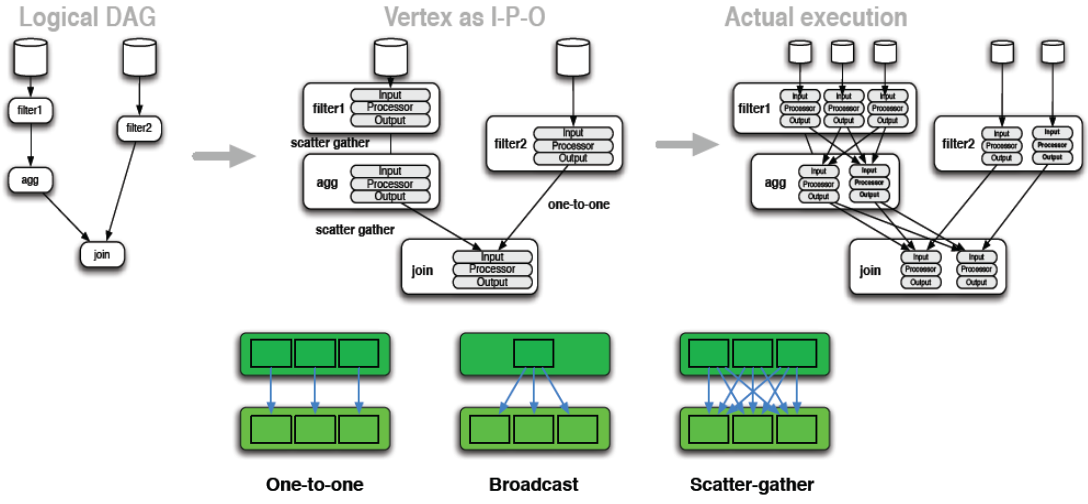
- Silnik wykonujący lepiej operacje MapReduce (ok. 10x szybciej)



27

Tez

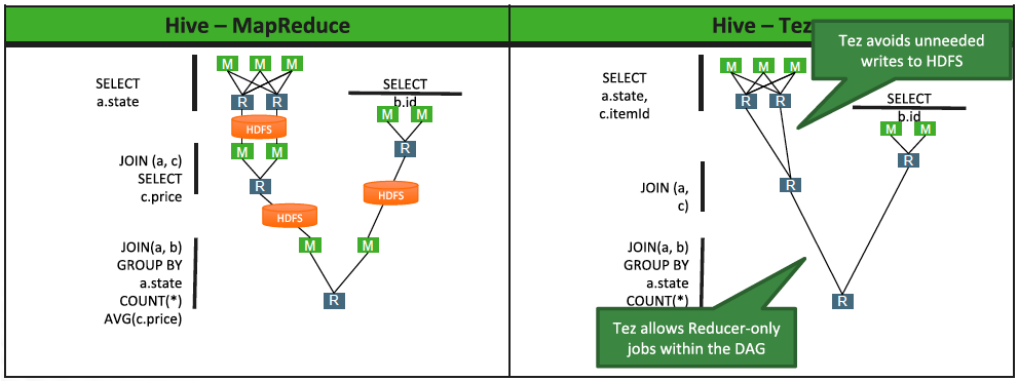
- DAG - (Directed- Acyclic-Graph) – skierowany graf acykliczny



28

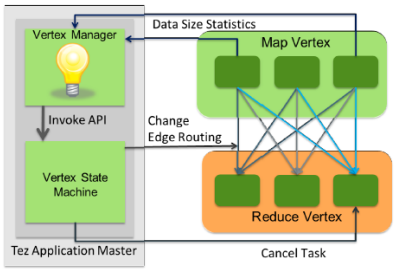
Tez

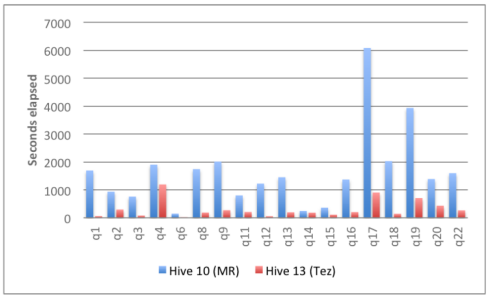
```
SELECT a.state, COUNT(*), AVG(c.price) FROM a
  JOIN b ON (a.id = b.id)
  JOIN c ON (a.itemId = c.itemId)
  GROUP BY a.state
```



29

Tez vs MapReduce





Wyniki testów Yahoo
(10TB danych, 4,200 serwerów, 46 PB HDFS i 90TB łącznej pamięci)

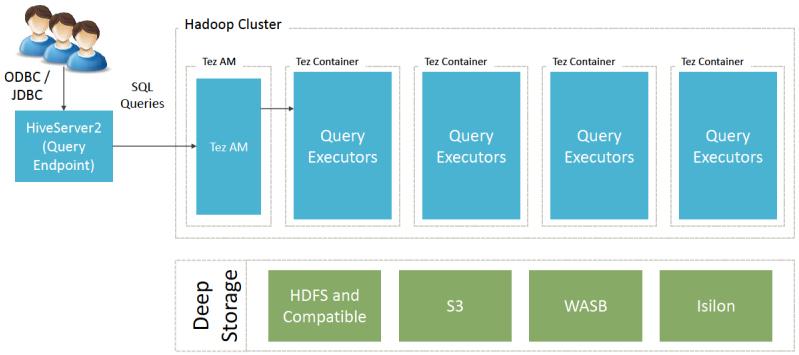
30

LLAP (Live Long And Prosper)

- **Trwałe demony** (kontenery) zapytań i inteligentne buforowanie w pamięci, aby dostarczać błyskawicznie szybkie zapytania SQL przy zachowaniu skalowalności.
- Dzięki trwałym (persistent) demonom zapytań unikamy **długiego czasu uruchamiania** kodu SQL.
  - Eliminacja przydzielania kontenerów i czasu uruchamiania JVM.
- Dzieli swoją pamięć podręczną pomiędzy wszystkich użytkowników SQL, maksymalizując wykorzystanie tego zasobu.
- Dysponuje drobnoziarnistym zarządzaniem zasobami i prewencją, co zapewnia równoczesny dostępu dla wielu użytkowników.
- Jeśli to możliwe, praca planowana jest na węźle z danymi w pamięci podręcznej, jeśli nie, zostanie wykonana w innym węźle.
- Jest w 100% kompatybilny z istniejącymi narzędziami Hive SQL i Hive.

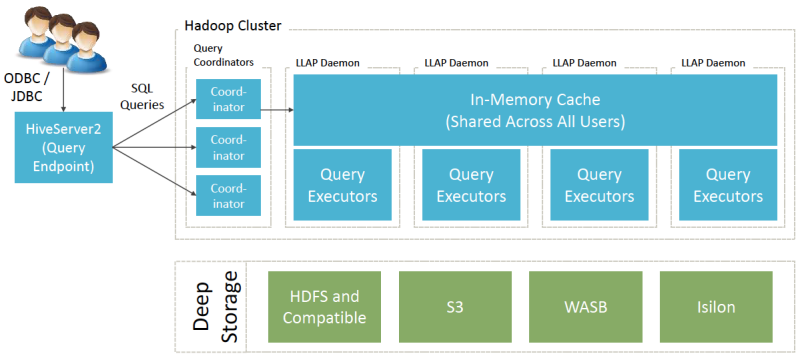
31

LLAP



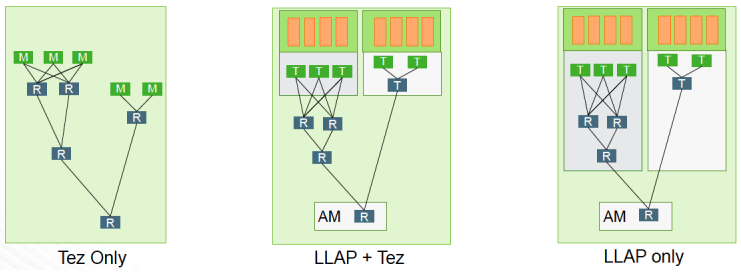
32

LLAP



33

LLAP + Tez



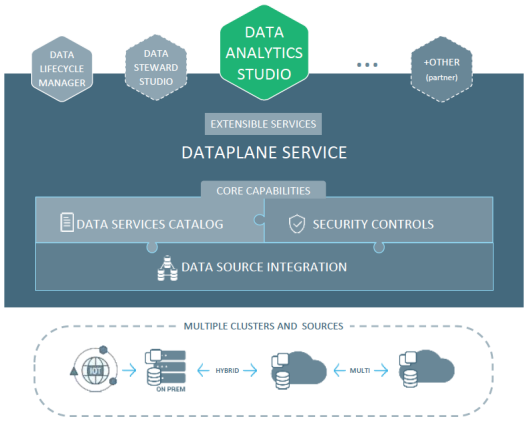
34

Inne

- Transakcje (informacja są przechowywane w MetaStore)
  - Transakcyjność zapewniona na poziomie jednego zapytania
  - Trwają pracę nad transakcyjnością dla bloku zapytań
- Zmaterializowane widoki (dane są przechowywane przez Hive w plikach)

35

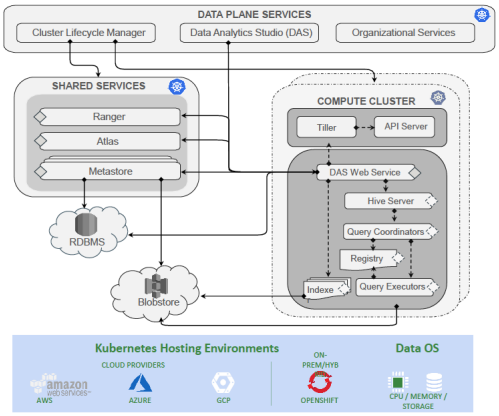
Data Analytics Studio



36

Hive na Kubernetes (trwają prace)

- Zalety:
  - Instalacja równoległa Hive/LLAP (do klastra głównego)
  - Wiele wersji Hive
  - Wielokrotne instancje magazynowe i obliczeniowe
  - Dynamiczna konfiguracja
  - Restarty zachowujące stan zadań (cache).
  - Restart kolejnych serwerów w celu aktualizacji. Szybki powrót do poprzedniego dobrego stanu.



37

Sqoop

- Narzędzie to transferu danych między Hadoop a:
- relacyjnymi bazami danych: Mysql, Oracle
- bazami półstrukturalnymi (semi-structured): Hbase, Cassandra
- nieustrukturyzowanymi: Hadoop
- działa z linii poleceń
  - `codegen` Generate code to interact with database records
  - `create-hive-table` Import a table definition into Hive
  - `eval` Evaluate a SQL statement and display the results
  - `export` Export an HDFS directory to a database table
  - `help` List available commands
  - `import` Import a table from a database to HDFS
  - `import-all-tables` Import tables from a database to HDFS
  - `import-mainframe` Import mainframe datasets to HDFS
  - `list-databases` List available databases on a server
  - `list-tables` List available tables in a database
  - `version` Display version information

38

Dziękuję za uwagę!

39