Hadoop

Maidics Barnabás
b.maidics@gmail.com

Tartalom

- Motiváció: “Big Data” adatfeldolgozás
- HDFS - Hadoop Distributed File System
- Apache Hive - SQL engine Hadoop fölött
- Fájlformátumok - optimalizálás

Motiváció

- Nagyvállalat: többszáz Terabyte adat keletkezhet naponta
- Use-case: lekérdezések futtatása Petabyte-os adatmennyiségen
- Hagyományos RDBMS-nek vannak korlátai
- Megoldás: Hadoop alapú adattárolás/feldolgozás

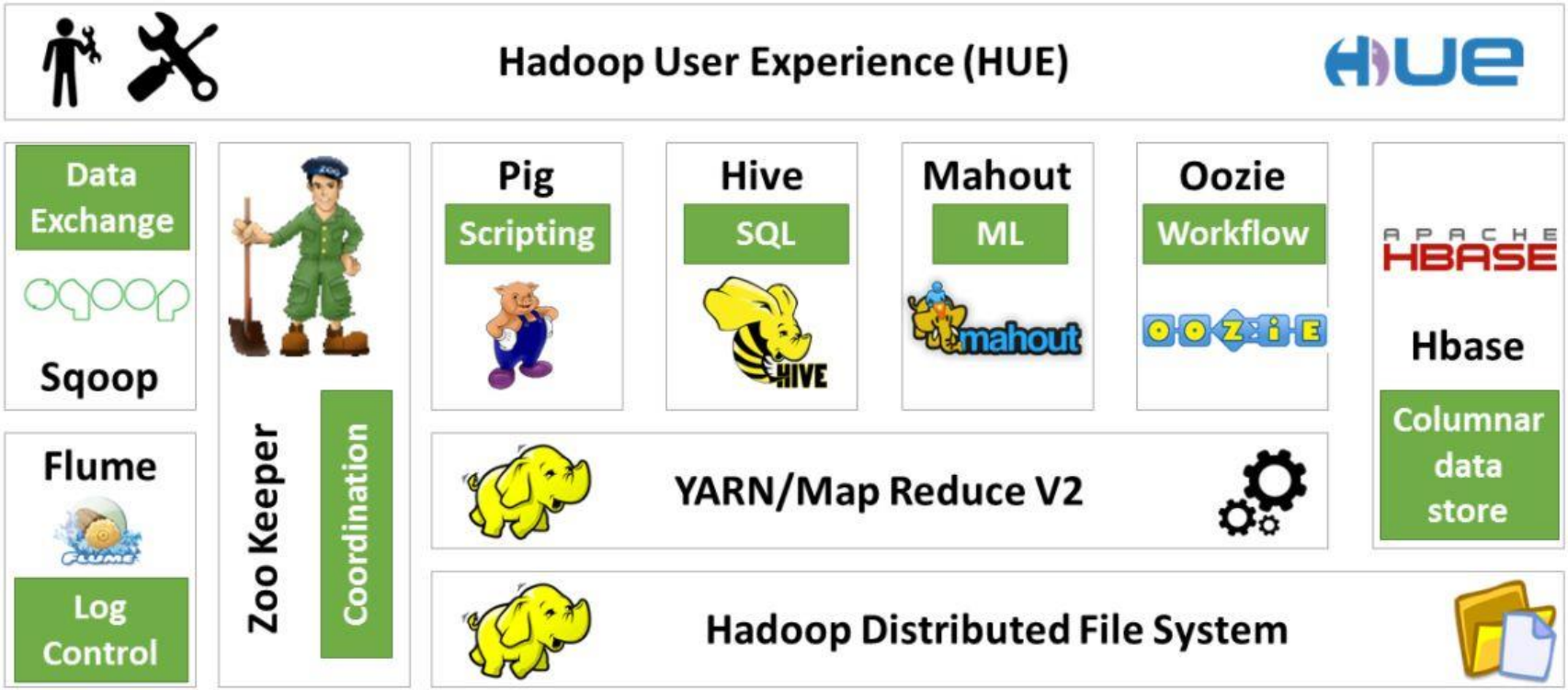
Amit eddig tudunk

- Centralizált architektúra: felesleges nekiállni
- Gyorsabb adatfeldolgozás => párhuzamosítás, ehhez elosztott architektúra
- MapReduce: programozási paradigma nagy adathalmazok párhuzamos feldolgozására

Ami hiányzik

Elosztott adattárolás

Hadoop ökoszisztéma



Apache Hadoop

- Modulok:
  - **HDFS (Hadoop Distributed FileSystem):** eloszott fájlrendszer
  - **Hadoop Mapreduce:** MapReduce paradigma implementációja
  - **YARN (Yet Another Resource Negotiator):** erőforrás-menedzselés, job ütemezés (nem fér bele)
  - **Hadoop Commons:** könyvtárak a többi Hadoop modulnak/komponensnek

HDFS

Hadoop Distributed File System

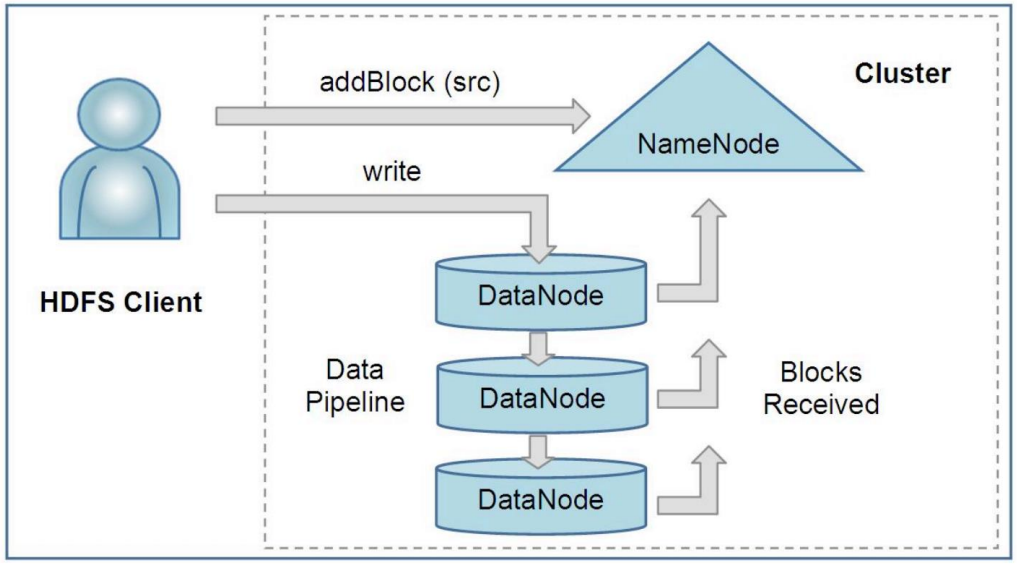
- Két féle node:
  - NameNode: fájlrendszer metaadatok tárolása
  - DataNode(s): alkalmazás adatok tárolása
- Célok, alapelvek:
  - Hardvermeghibásodás normál, nem pedig kivétel: 1 HDFS példány több 1000 Node-ból áll => mindig lesz nem működő Node.
  - “Simple Coherency”: WORM: fájlt írás után nem szabad módosítani, csak hozzáfűzni
  - Feldolgozóegység mozgatása az adat helyett
  - Megbízhatóság: adatvesztés minimalizálása -> replikációs faktor
  - “Commodity” hardverből építkezés => könnyű, olcsó horizontális skálázás

NameNode

- Fában tárolja az összes fájl helyét a fájlrendszerben (a fájlt közvetlenül NEM)
- Egyéb metaadatok: fájlok mérete, jogosultságok stb.
- Vele kommunikálnak a kliensek, ha keresnek egy fájlt: visszaadja azokat a DataNode-okat ahol megtalálható a fájl
- Követelmény a NameNode felé: NAGY rendelkezésreállás
- SPOF

DataNode / kliens

- DataNode-on tároljuk a fájlokat
- HDFS blokk (128 MB)
- Slave Node-ok, Heartbeat küldése a NameNode-nak
- Kezeli a HDFS kliensek írási/olvasási kéréseit



Kész a Big Data rendszer?

- Van
  - Elosztott adattárolást támogató fájlrendszer
    - Nagy rendelkezésre állás
    - Megbízható
  - MapReduce paradigma
    - Elosztott, párhuzamos adatfeldolgozás
    - Használhatóság: map/reduce függvény megírása java nyelven
    - Bonyolult, közepesen bonyolult probléma => több száz soros java kód
- Felhasználók az SQL világából jönnek => Hive

Ami hiányzik

SQL

Apache Hive

- MapReduce program: nehéz karbantartani, újrahasználni
- SQL lekérdezések futtatása Hadoopon
- Hive QL (HQL): SQL “szerű” lekérdező nyelv
- Hive vs. RDBMS
  - WORM
  - Schema on Read
  - Nincs határ a feldolgozandó adatmennyiségnél
  - Olcsón skálázható
  - Hive data volume: petabytes
  - ACID
    - Kezdetben nem, jelenleg csak sor szintű, csakis ORC fájlformátummal, sok limitációval

Alap ötlet

- MapReduce jobok létrehozása és futtatása SQL lekérdezésből
- select * from tablename limit 1;
  - Erre felesleges MapReduce job -> HDFS-ről kiolvasható könnyen
  - select col1 from tablename limit 2 where col2 > 10;
    - Csak Map job!
  - select col1, count(col2) from tablename group by col3 where col1 >10;
    - Map job: kiolvasás, filter
    - Reduce job: aggregálás (Map eredményeinek összefűzése)

Hive adattárolás

- HDFS-en tárol
- Egységek:
  - **Database**: névtér, a táblák névütközésének elkerülése érdekében
  - **Table**: tárolási egység azonos sémával rendelkező adatokra
  - **Partition**: keresés felgyorsítására szolgáló egység. Tábláinkat partíciókra oszhatjuk. “Virtuális” oszlop.

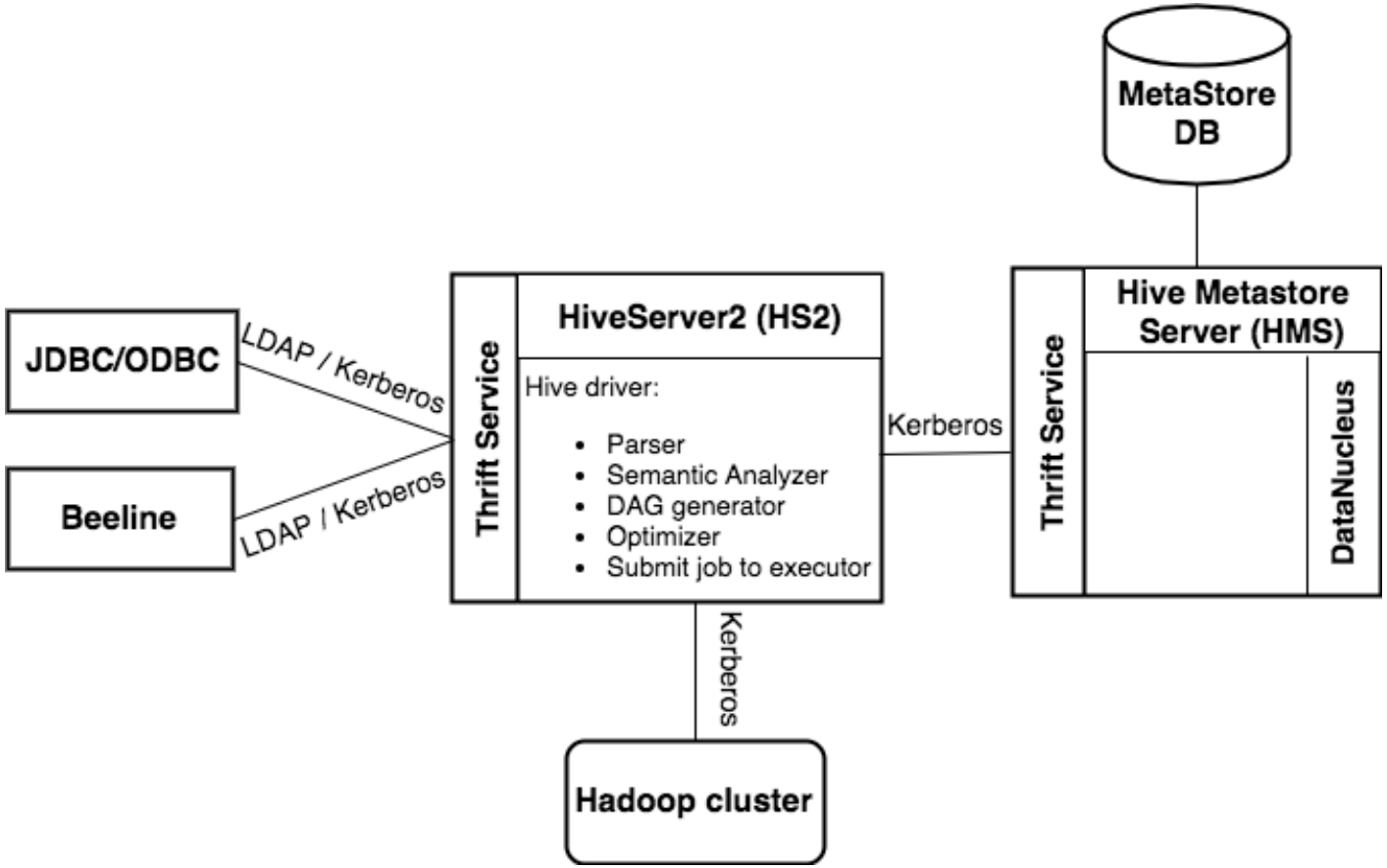
Hive adattárolás 2

- Példa:
  - CREATE TABLE test_table(c1 string, c2 int) PARTITIONED BY (date string, hour int);
    - HDFS-en: /user/hive/warehouse/test_table
  - INSERT OVERWRITE TABLE test_table PARTITION(date='2018-01-01', hour=12) SELECT * FROM t;
    - HDFS-en: /user/hive/warehouse/test_table/date=2018-01-01/hour=12
  - SELECT * FROM test_table WHERE date='2018-01-01';
    - Csak a date=2018-01-01 mappát kell kiolvasni

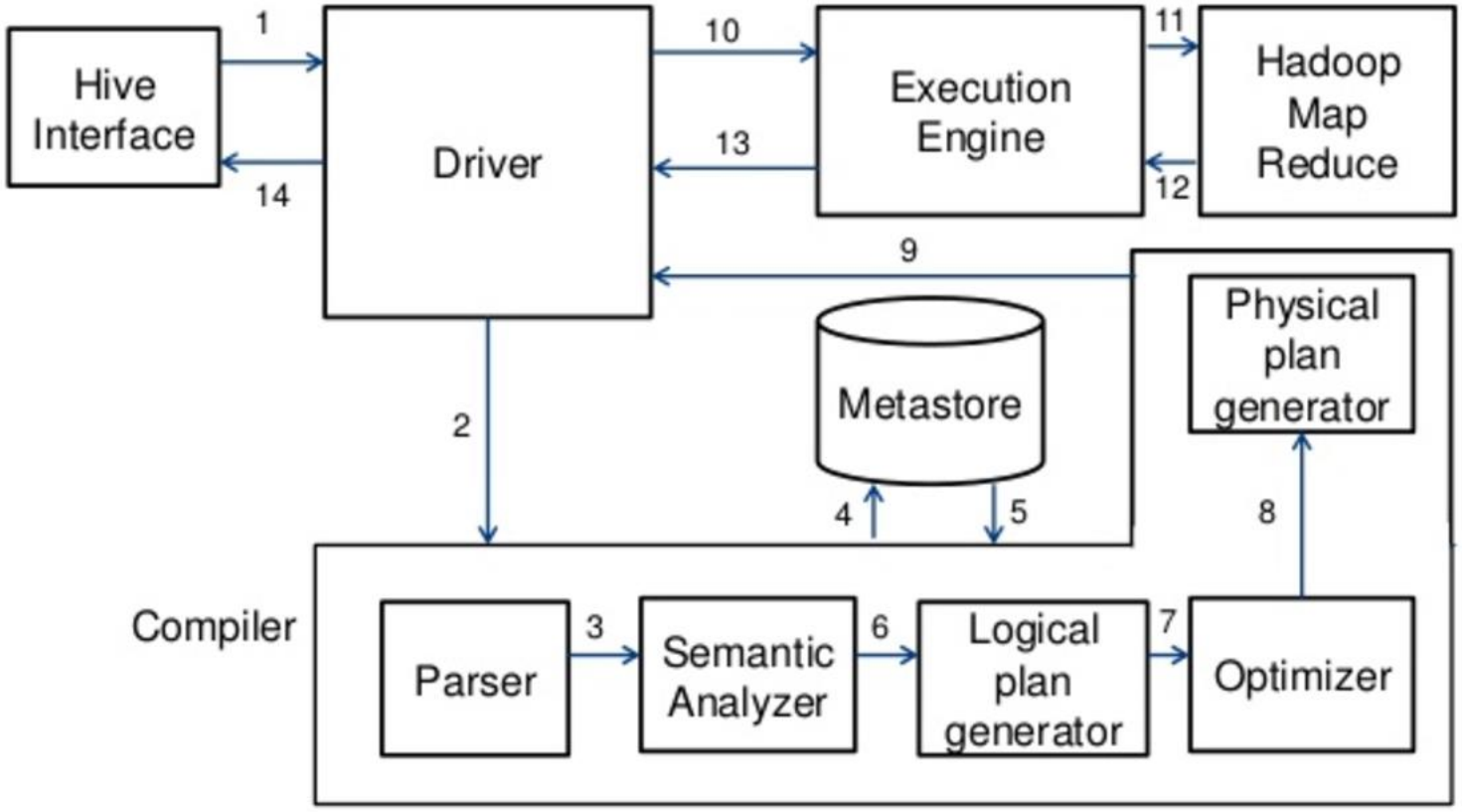
Hive adattárolás 3

- Kétfajta tábla:
  - Managed: <warehouse_root_directory>/table_name
  - External:
    - Nem a Hive HDFS könyvtárában található táblák
    - CREATE EXTERNAL TABLE test_external(c1 string, c2 int) LOCATION '/user/example_table/example_data';
    - Hive a sémát tárolja, feltételezi hogy illeszkedik rá a tábla
    - Különbség:
  - Drop table: Managed tábla esetén az adat is törlődik, External esetén nem

Hive architektúra



Egy lekérdezés élete

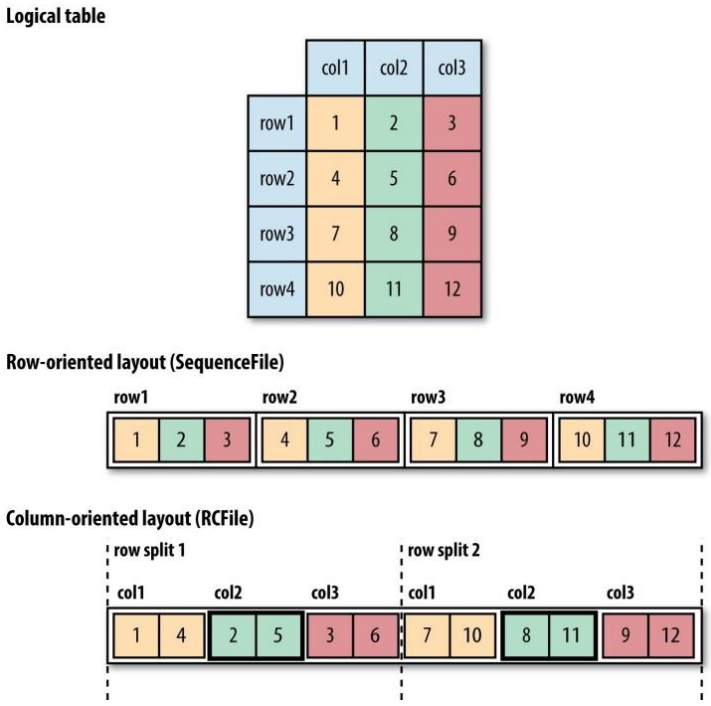


Hogy tároljuk adatainkat?

- Textfile, CSV, JSON, XML?
- Szempontok:
  - Tördelhetőség
  - Blokk alapú tömörítés
  - Mire szerenénk optimalizálni?
    - Write: file kiírás
    - Partial read: egy-egy oszlopot kiolvasása
    - Full read: minden adat kiolvasása egy fájlból

Sor vs oszlop alapú tárolás

- **Sor orientált:** sorfolytonos tárolás
- **Oszlop orientált:** a sorokat eltördeljük, majd minden “split-ben” oszlopfolytonos tárolás

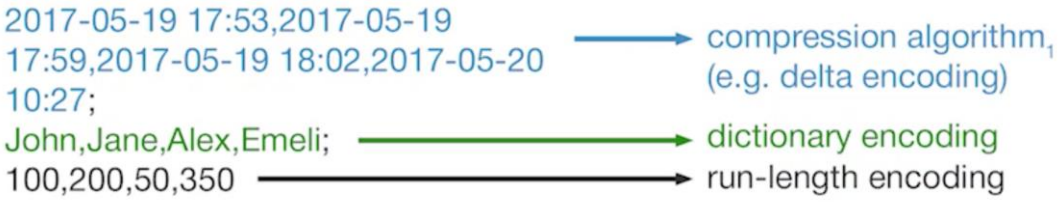


Oszlop orientált

- **Partial-read**: ha csak pár oszlopot szeretnénk, nem kell az egész sort betölteni memóriába
- Jobb tömörítés: Kihasználhatjuk hogy egymás mellett ugyanolyan típusú értékek vannak (pl timestamp esetén delta-encoding)
- Hátrány: lassabb írás egész rekordok esetén

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



XML, JSON

- Nem tördelhető
- Nem ajánlott a használata

Text, CSV

- Legelterjedtebb
- Tördelhető
- CSV nem támogatja a blokk alapú tömörítést => olvasási teljesítménynél rosszabb
- Nem tárolunk metaadatokat CSV-ben

- Séma evolució (Schema evolution)
  - A fájlformátum függ a mezők sorrendjétől =>
    - Új mező felvétele: csak a rekordok végéhez fűzve
    - Meglévő mező törlése: nem tudjuk megoldani

Avro

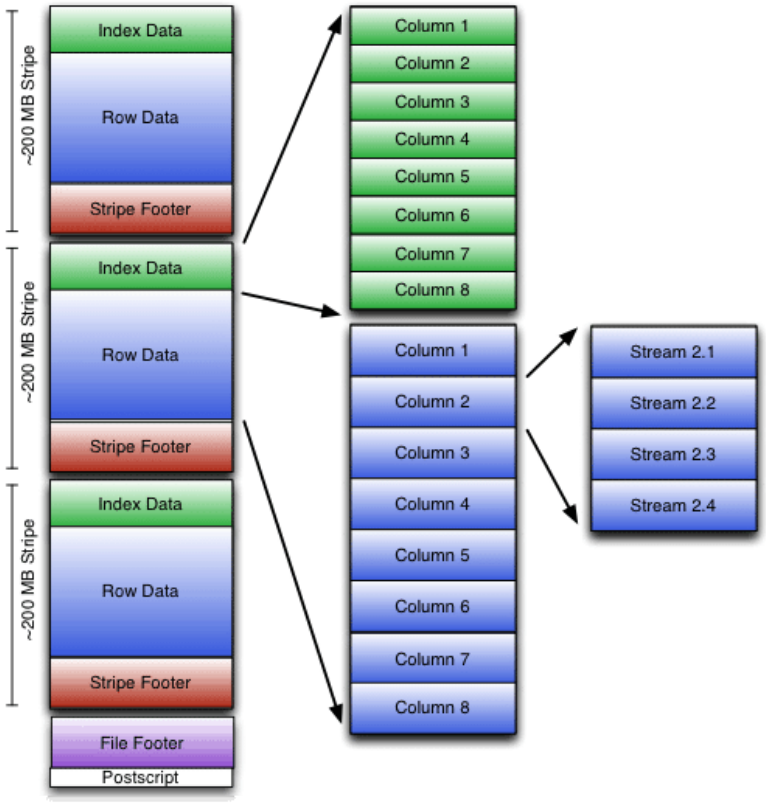
- Sor orientált
- Fájlokban tárolja a metaadatokat is => séma evolúció
- Támogatja a blokk alapú tömörítést

RCFile

- Record Columnar File
- Oszlop orientált
- Oszlop orientált tárolás előnyeit hordozza
- Nem támogatja a séma evolúciót => dinamikus workload esetén nem ajánlott a használata
- Helyette: ORC vagy parquet

ORC

- Optimized Row Columnar (file format): Hortonworks és Facebook
- Sémainformációk a footer-ben
- Indexeket tartalmaz
  - Max, min érték egy “stripe”-on belül => ezt felhasználva egy egész stripe-ot kihagyhatunk (nem kell felolvasni, dekódolni)
  - Tartalmaz sum, count értékeket => aggregate lekérdezések nagyban gyorsulnak
- Sok más trükk



ORC tábla létrehozása Hive-ban

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")
…;
```

Parquet

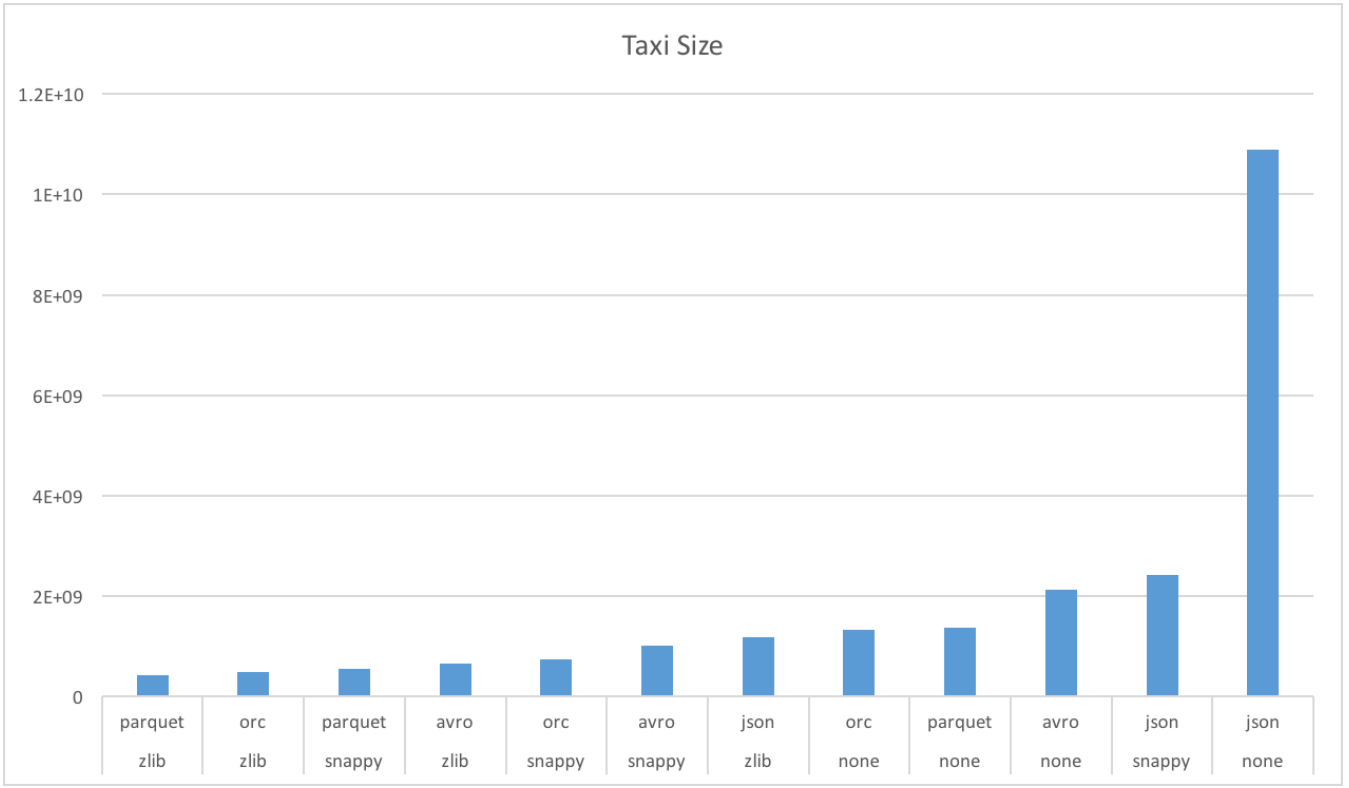
- Cloudera és Twitter
- Nagyvonalakban ugyanaz, mint ORC
- Legnagyobb különbség: Cloudera által támogatott

Melyiket használjam?

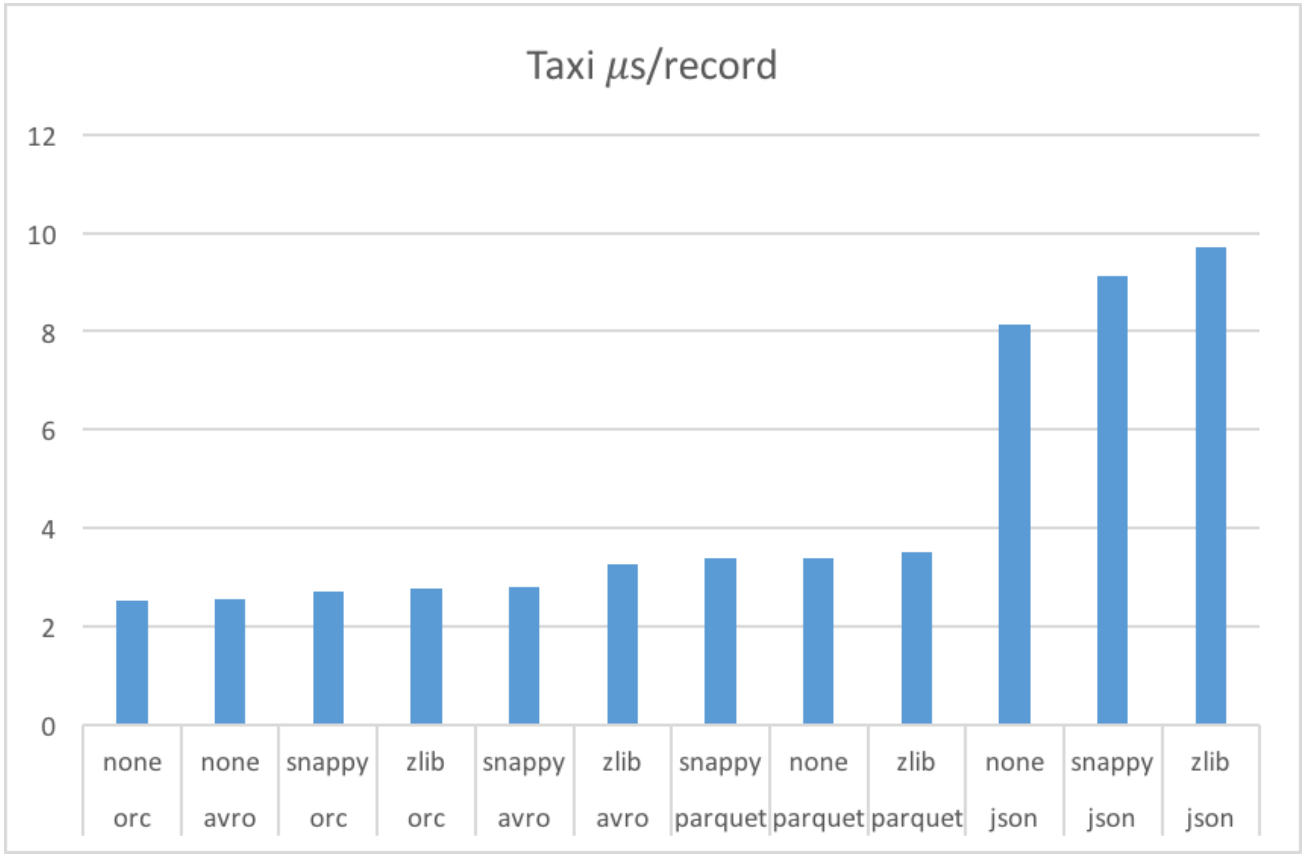
- Szempontok:
  - Hadoop disztribúció típusa (már nem annyira releváns):
    - Cloudera - Parquet
    - Hortonworks - ORC
  - Workload típus
    - Read heavy: oszlop orientált (ORC, Parquet)
    - Write heavy: sor orientált (Avro, CSV)
  - Tömöríthetőség: Ha fontos akkor oszlop orientáltak jobbak
  - Hadoopról adatátvitel közönséges adatbázisba?
    - CSV-vel a legkönnyebb
    - ACID: ha szükséges akkor ORC
    - Nem ritka, hogy több formátumban is eltárolják egyszerre, hogy támogassák a változó követelményeket

Benchmarks - tömörítés

- New York Taxi adatok: 23 millió sor



Benchmark - Full table scan



Amit még a Hive tud

- Több “Execution Engine” támogatása
  - Hadoop MapReduce
  - Apache Spark
  - Apache Tez
- ACID: fejlesztés alatt, limitált
- CBO (Cost Based Optimization)
- Materialized views
- Caching

Más megoldások

- Apache Impala: BI/analytic lekérdezések, Cloudera
- Presto: Facebook
- BigQuery: Google
- Miért Hive?
  - Legrégebbi, legkiforrottabb
  - Több ezer konfigurációs lehetőség
  - Stabil
  - Open source

“Takeaway”

- **HDFS**: elosztott, hibatűrő fájlrendszert biztosít
- **MapReduce**: HDFS-t felhasználva elosztott adatfeldolgozás
- **Hive**: SQL adattárház eszköz Hadoop fölött
- **Fájlformátum** választása a workload-hoz