NÁSKOK
DÍKY
ZNALOSTEM

PROFINIT

B0M33BDT
Technologie pro velká data

Storage

Milan Kratochvíl

25.10.2017

Motivace

- Jak efektivně ukládat data v Hadoop ekosystému?
  - formát ukládání dat a jejich komprese
  - možnost paralelně zpracovávat na mnoha nodech
- Jak efektivně přistupovat k datům v Hadoop ekosystému?
  - nástroje na dotazování/ukládání dat
- Jak efektivně spravovat data v Hadoop ekosystému?
  - „data lifecycle management“

PROFINIT

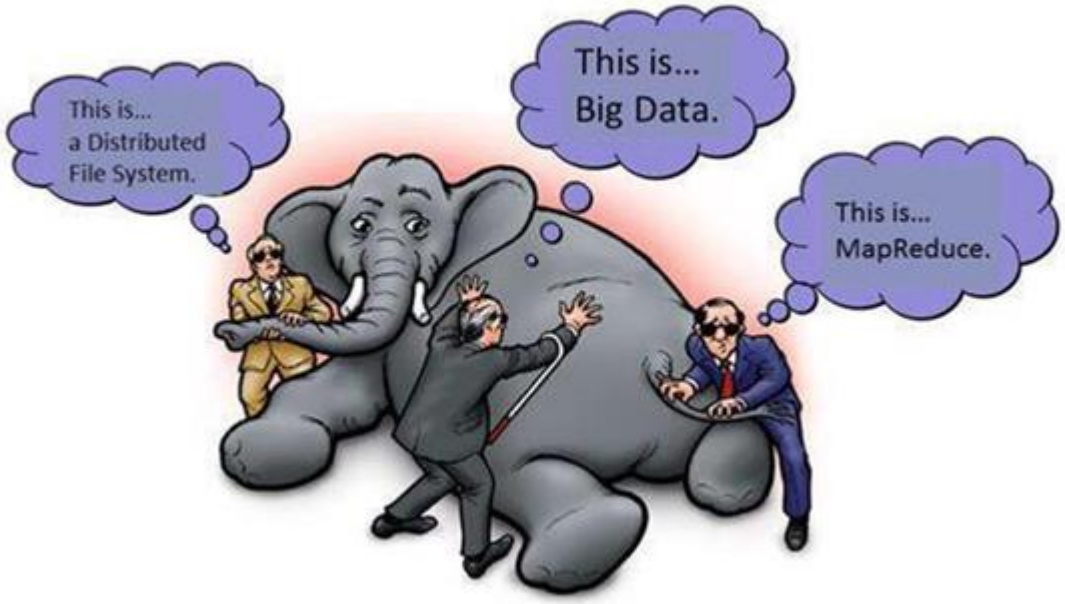
Osnova přednášky

- HDFS – možnosti ukládání dat
- Formáty ukládání dat v Hadoop
- Komprese
- Hive/Impala
- HBase
- Shrnutí

PROFINIT

HDFS

What is this Hadoop thing?



PROFINIT

HDFS – opakování

- Distribuovaný file system
- Namenode vs Datanode
- Replikace (3 by default)
- Velké soubory fyzicky rozloženy do bloků (nemusí být na stejném nodu!)

- Co musíme řešit:
  - Jak pracovat s velkými soubory?
  - Jak pracovat s mnoha malými soubory?
  - Jak modifikovat soubory?

HDFS – API

- Java API
  - Soubory jsou z pohledu API streamy
- Podporované operace
  - create (zápis) `[hadoop fs -put]`
  - open `[hadoop fs -get]`
  - delete `[hadoop fs -rm]`
  - append `[hadoop fs -appendToFile]`
  - seek (je možno v otevřeném souboru „skákat“) `[N/A]`

HDFS – ukládání souborů

- Typy souborů – vstupní data
  - Binární
    - exporty zdrojových systémů (CDRs - telco)
    - videa, obrázky, zvukové záznamy
  - Textové
    - prostý text – CSV (comma-separated values), TSV (tab-separated values) apod.
    - XML
    - JSON
  - Komprese
    - žádná
    - zip, gzip

PROFINIT

HDFS – ukládání souborů

- Typy souborů – interní formáty ukládání dat
  - Binární – beze změny
  - Textové
    - prostý text – CSV (comma-separated values), TSV (tab-separated values) apod.
  - „Tabulární“ řádkově orientované úložiště
    - **CSV/TSV** – nenese informaci o schématu
    - **Avro** (https://avro.apache.org/) – ve skutečnosti nástroj na serializaci, obsahuje schéma
  - „Tabulární“ sloupcově orientované úložiště – v metadatech obsaženo schéma
    - **RCfile**
    - **ORC** (https://orc.apache.org/)
    - **Parquet** (https://parquet.apache.org/)
  - Ostatní
    - **SequenceFile**

Řádkově a sloupcově orientované úložiště

- CSV, tradiční relační databáze – **řádkově orientované** (většinou)
- Příklad (https://en.wikipedia.org/wiki/Column-oriented_DBMS)

RowId	EmpId	Lastname	Firstname	Salary
001	10	Smith	Joe	40000
002	12	Jones	Mary	50000
003	11	Johnson	Cathy	44000
004	22	Jones	Bob	55000

- Reprezentace úložiště

řádkově

```
001:10,Smith,Joe,40000;
002:12,Jones,Mary,50000;
003:11,Johnson,Cathy,44000;
004:22,Jones,Bob,55000;
```

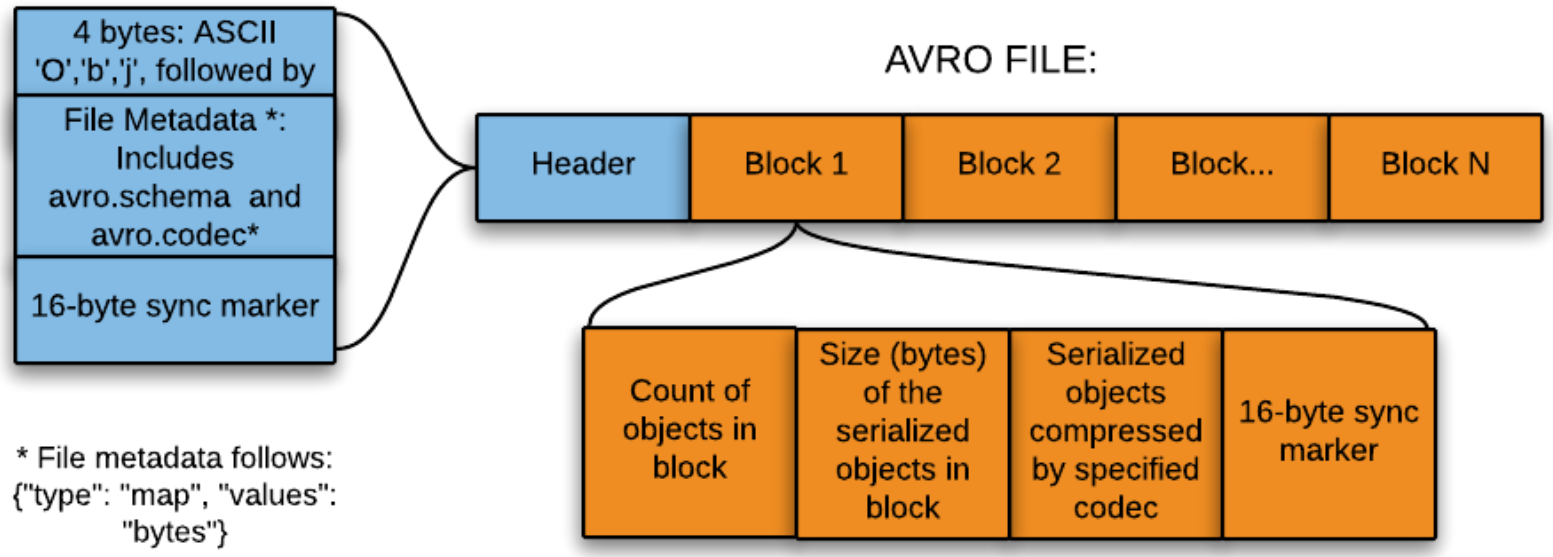
sloupcově

```
10:001,12:002,11:003,22:004;
Smith:001,Jones:002,Johnson:003,Jones:004;
Joe:001,Mary:002,Cathy:003,Bob:004;
40000:001,50000:002,44000:003,55000:004;
```

PROFINIT

Avro

- Řádkově orientované úložiště/formát pro serializaci
- Nese schéma, umožňuje „appendovat“ data (stejné schéma)



http://www.svds.com/dataformats/

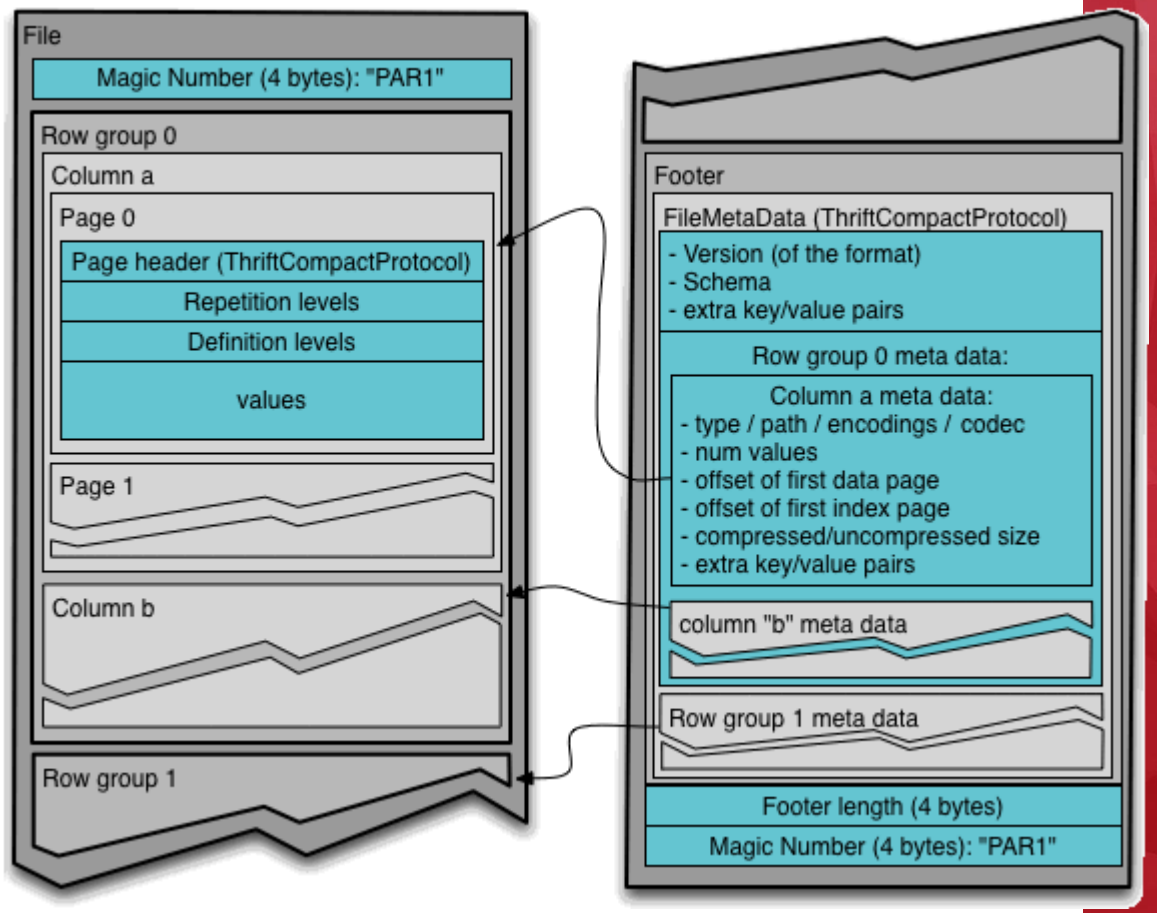
PROFINIT

Sloupcově orientované úložiště

- Výhody
  - Možnost **rychle** číst **jen sloupce,** které potřebuji (omezení IO operací disků)
  - Efektivnější komprese
- Nevýhody
  - Práce s jedním záznamem a potřeba čtení celého záznamu je pomalá
  - Nelze snadno modifikovat – ale to v Hadoop stejně prakticky není možné (myšleno napřímo – tedy nejedná se o nevýhodu!)
  - Náročný pro zápis
    - Vstupní data je třeba rozdělit do „bloku řádků“, až ty se ukládají sloupcově
    - Velikost každého „bloku řádků“ se musí vejít do bloku HDFS
- Shrnutí
  - Výhodné použít pro OLAP
  - Velmi nevhodné pro OLTP

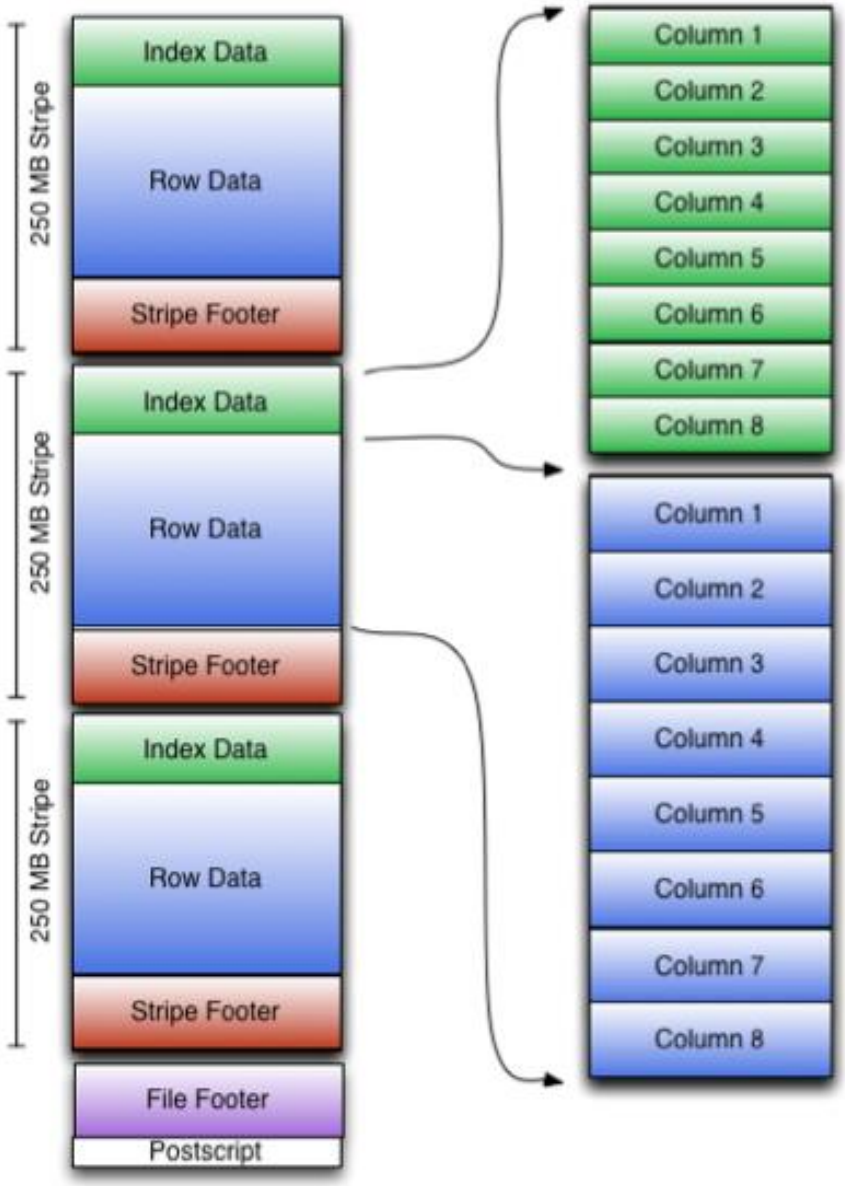
Parquet

```
4-byte magic number "PAR1"
<Column 1 Chunk 1 + Column Metadata>
<Column 2 Chunk 1 + Column Metadata>
...
<Column N Chunk 1 + Column Metadata>
<Column 1 Chunk 2 + Column Metadata>
<Column 2 Chunk 2 + Column Metadata>
...
<Column N Chunk 2 + Column Metadata>
...
<Column 1 Chunk M + Column Metadata>
<Column 2 Chunk M + Column Metadata>
...
<Column N Chunk M + Column Metadata>
File Metadata
4-byte length in bytes of file metadata
4-byte magic number "PAR1"
```



PROFINIT

ORC



PROFINIT

Sloupcově orientovaná úložiště

- RCFile – jen historické aplikace
- ORC – Optimized Row Columnar file format
  - vše, co RCFile + něco navíc
  - ukládá metadata/statistiky (average, max, count...)
  - může obsahovat i indexy
  - načítá jen potřebná data – optimalizuje dotaz
- Parquet – velmi podobné, jednoduší/lightweight

- Metadata na konci souboru, resp. bloku
  - zapisují se, až když je soubor známý
  - na jaké konkrétní místo souboru skočit

Odbočka – malé soubory

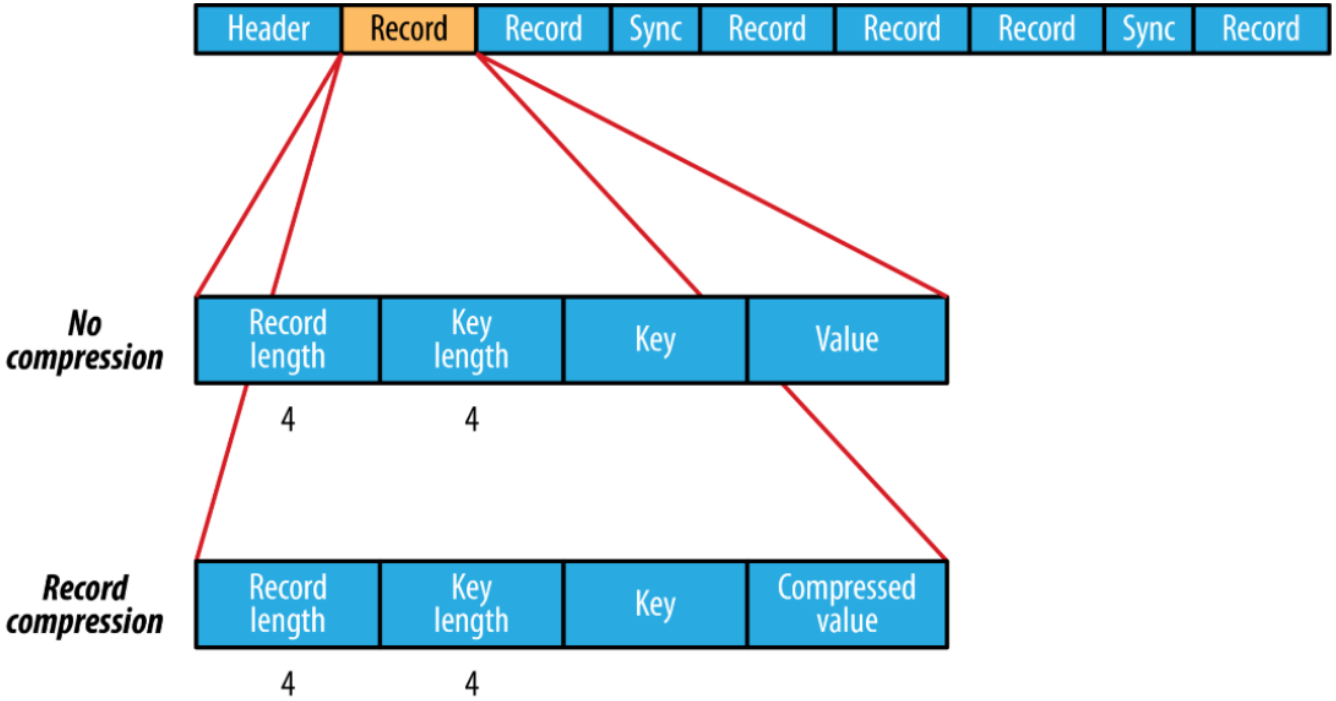
- Mnoho malých souborů je problém
  - malý soubor typicky stovky bajtů až stovky kilobajtů
- Typická situace – když už potřebuji zpracovávat malé soubory, pak jich je skutečně mnoho
- Kolik je mnoho?
  - Více než desítky milionů

- 1 záznam o bloku na HDFS cca 200bajtů v RAM na NameNode
- Příklad 1 soubor 10kB
  - 10E6 souborů ➔ 100 GB dat ➔ cca 2 GB RAM na NameNode
  - 1E9 souborů ➔ 10 TB dat ➔ cca 200 GB RAM

- Typické řešení na následujícím slajdu

SequenceFile

- Umožňuje ukládat data ve formátu key/value
  - key – název souboru
  - value obsah souboru
- Umožňuje blokové rozdělení/blokovou kompresi
- Umožňuje data přidávat [append]
- Vhodné pro full scan



Komprese

- Výrazně zrychlí přístup k datům
  - opakování – nejpomalejší jsou IO operace
- Používané algoritmy
  - **Gzip** – základní formát, často vstupní soubory
    - v Hadoop přímo se moc nepoužívá, relativně pomalý, ale účinný
  - Bzip2
  - LZO
  - **Zlib** – typicky používaný ve spojení s ORC; stejný algoritmus jako Gzip
  - **Snappy**
    - typicky používaný ve spojení s Parquet (ale nejen to)
    - nižší účinnost, ale nejrychlejší

Kompresní algoritmy - srovnání

Algoritmus	Rychlost	Účinnost	„Splittable“
GZIP/ZLib		✓	
BZip2		✓	✓
LZO	✓		✓
Snappy	✓		

- „Splitovatelnost“
  - kompresní algoritmus vytváří bloky, které lze samostatně dekomprimovat
  - nutnost pro paralelní zpracování
- Kompatibilita
  - **Ne každý formát a každý nástroj podporuje libovolnou kompresi!** (Např. v Impala nelze použít ORC)

PROFINIT

Kdy použít jaký kompresní algoritmus?

- Častý přístup k datům
  - „hot data“
  - minimalizace doby přístupu k datům
  - LZO, Snappy
- Archivace, řídký přístup k datům
  - „cold data“
  - minimalizace požadovaného diskového prostoru
  - GZIP, BZip2
- Kdy potřebuji „splittable“ algoritmus?
  - Možná nikdy... Obrovské CSV?
  - Inteligentní souborové formáty obsahují bloky, které se komprimují; nekomprimuje se celý soubor

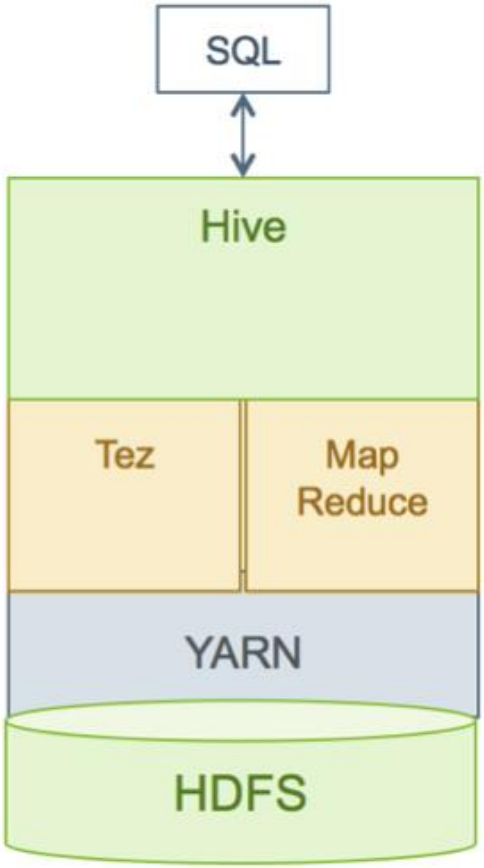
Hive a Impala

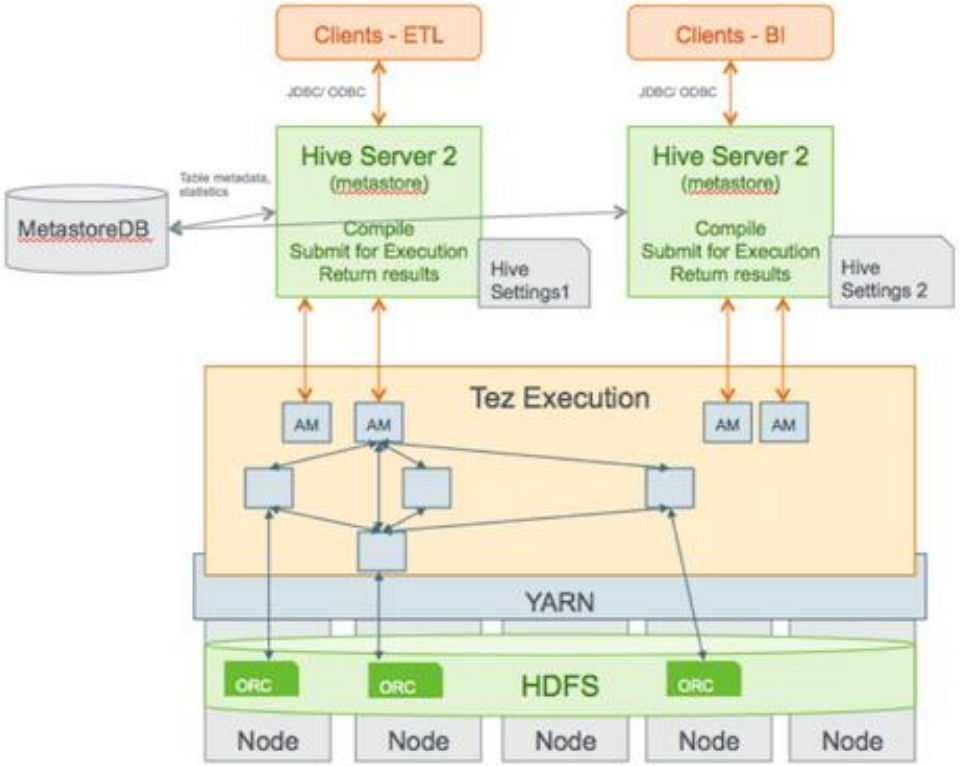
PROFINIT

Hive

- https://hive.apache.org/
- Snaha přivést SQL do světa Hadoop
- Nástroj pro dotazování a manipulaci s daty
- Vlastní jazyk HQL (variace na SQL)
- Exekuce dotazů probíhá prostřednictvím klasických technologií Hadoop
  - silná i slabá stránka zároveň
- Poměrně vyspělý a mocný nástroj

Hive – architektura





› https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.3.4/bk_performance_tuning/content/ch_hive_architectural_overview.html

PROFINIT

Hive – ukládání dat

- Přístup k datům je prostřednictvím „klasických DB“ tabulek
- Data jsou ukládána v HDFS
  - externí tabulky
  - managed tabulky
- Jak je tabulka uložena?
  - Tabulka je celý adresář
  - Obsahuje více souborů, nebo i dalších podadresářů
- Data jsou ukládána ve vhodném formátu
  - Parquet, Avro, CSV, ORC...
- Metadata jsou uložena v Metastore
  - klasická relační DB – MySQL, PostgreSQL
  - umístění souborů
  - statistiky
  - práva

Hive – HQL

- DDL (Data Definition Language)
  - `CREATE [EXTERNAL] TABLE`
  - `DROP TABLE`
  - `TRUNCATE TABLE`
  - `ALTER TABLE`
- DML (Data Manipulation Language)
  - `LOAD DATA`
  - `INSERT INTO TABLE, INSERT OVERWRITE TABLE`
- Query
  - `SELECT`

- Nelze obecně
  - `UPDATE`
  - `DELETE`
  - *(až na specifické případy tabulek s podporou ACID)*

PROFINIT

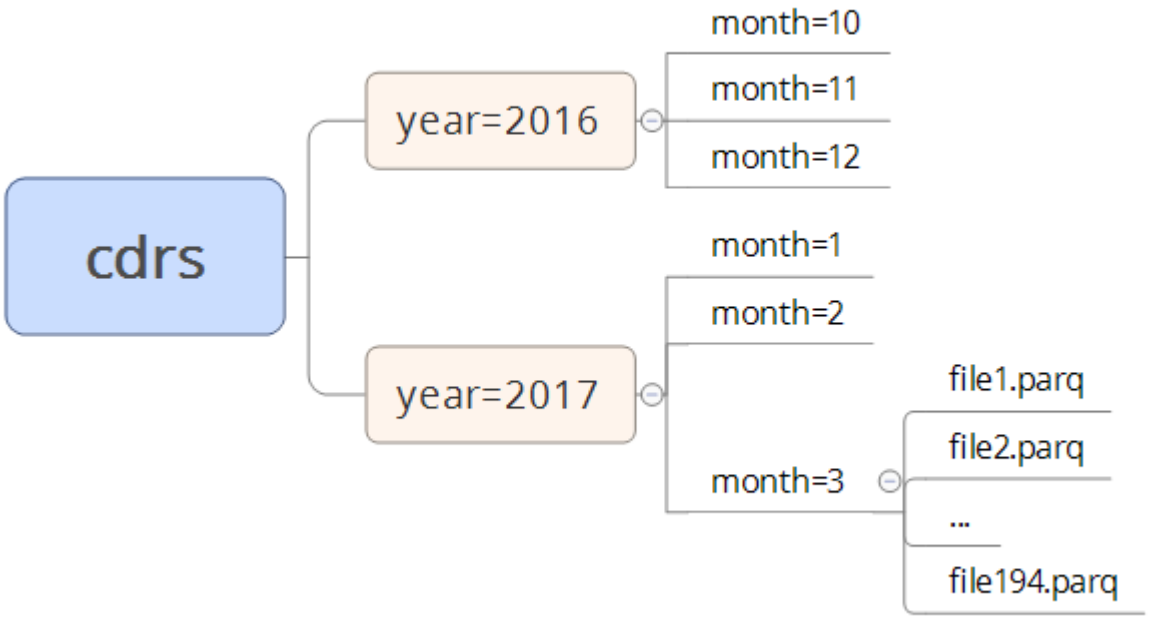
Loady a inserty dat

- Hive vytváří pro každou tabulku alespoň jeden soubor
  - pokud je dat hodně, vytvoří se více souborů (konfigurovatelné)
- Ale pozor – **při každém INSERT se vytvoří vždy aspoň jeden nový soubor!**
- Nedává tedy smysl INSERTovat záznam po záznamu jako v RDBMS
- Vždy INSERT z tabulky (např. pomocí externí tabulky)

- Cíl: můžeme mít mnoho souborů, ale měly by mít ideálně velikost cca 1 HDFS bloku (případně násobku) [typicky 128MB]

Partitioning (velmi důležité)

- Příklad:
  - Data chodí s časovým razítkem a jejich velmi mnoho (např. tel. hovory)
  - Typicky mě ale zajímají jen údaje za konkrétní den/měsíc...
  - Jak se k těmto datům rychle dostat?
- Partitioning
  - Logické rozdělení struktury tabulky do podadresářů



Hive – další možnosti

- Partition
  - s každou partition lze pracovat samostatně, např. DROP, LOAD, INSERT
- Indexy
  - podpora pro indexy, možnost psát vlastní indexery
  - defaultní indexer indexuje záznam k souboru v tabulce
- Bucketing
  - rozděluje data do definovaného počtu „kyblíčků“ podle zvoleného sloupce
  - pro konkrétní hodnotu se
    - spočítá hash (snaha o rovnoměrné rozdělení)
    - a na ten se aplikuje modulo počtu kyblíků
  - při dotazu na zvolený sloupec Hive šáhne přímo do správného kyblíku

PROFINIT

Jak na partitioning a bucketing?

- Stále platí – chceme mít soubory velikosti cca HDFS bloku
- Příliš mnoho partitions a bucketů může znamenat velmi malé soubory
- Např. při partioningu podle času je třeba zvážit, zda potřebuji skutečně partition podle jednotlivého dne, anebo stačí na celý měsíc

- Při návrhu partitions/buckets je třeba být uvážlivý. Prakticky nelze změnit!
- Změna = kompletní reload dat

PROFINIT

Hive – Execution engine

- Původně Hive využíval pouze klasický MapReduce
  - pomalé
  - intenzivní zápisy na disk
  - ale paměťově nenáročné
- V současné době ale servery mají dostatek paměti – má smysl hledat i jiné cesty
- Hive umožňuje nastavit „execution engine“
  - MapReduce (Hive on MapReduce)
  - Tez (Hive on Tez)
  - Spark (Hive on Spark)
- Ve výsledku ale vždy trvá určitou dobu (cca desítky sekund) než se celý stroj rozběhne...

Impala

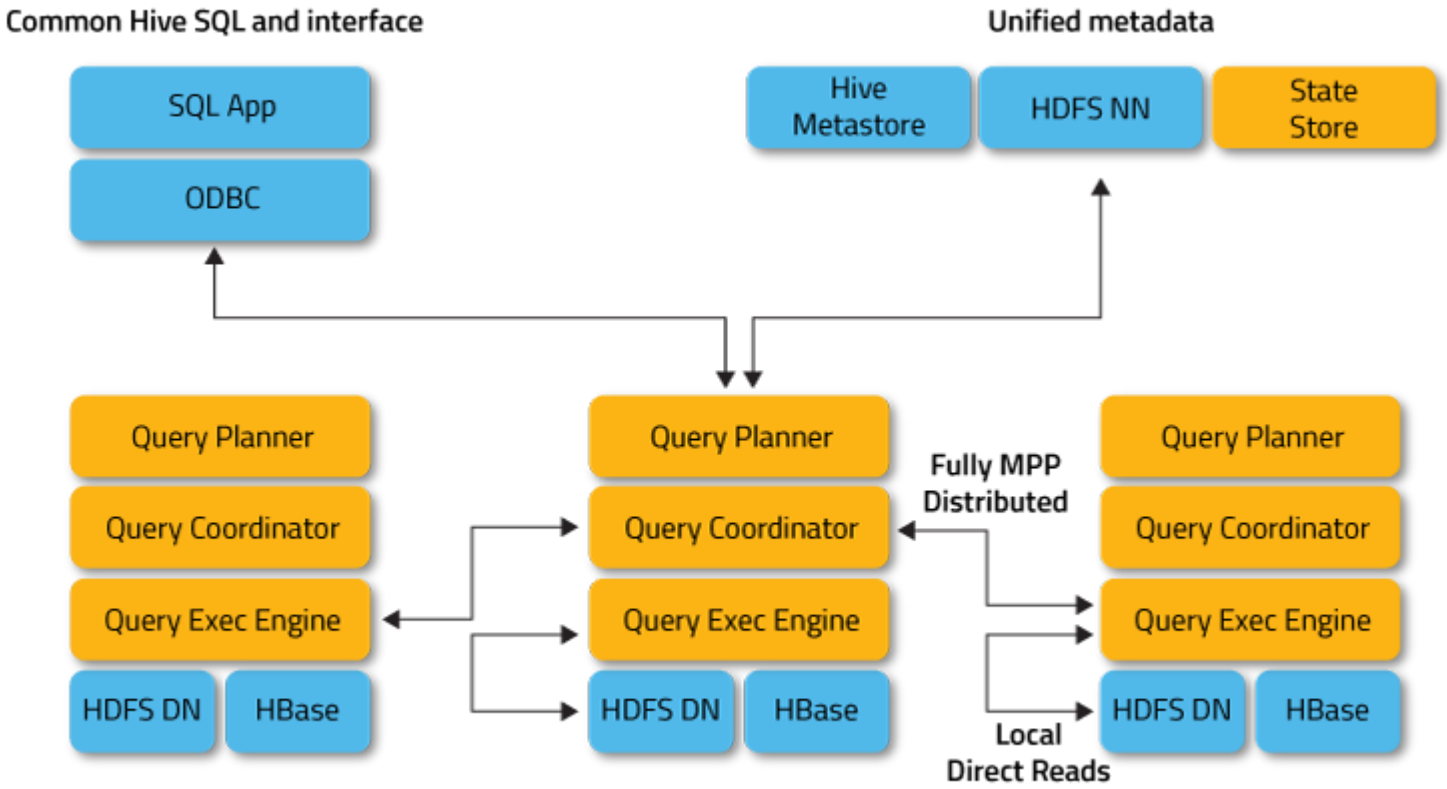
- Nástroj podobný Hive
- Velmi kompatibilní query language
- Sdílí Metastore, tj.
  - tabulky vytvoření v Impala jsou viditelné v Hive a (téměř) vice versa
- Impala je ale rezidentní
  - nemusí spouštět nové procesy, ale nepodporuje YARN
  - na (některé) dotazy umí dát odpověď téměř ihned (velmi záleží na dotazu a na formátu souboru)
  - ideální pro analytické dotazy (?)
- Omezenější v pestrosti
  - řadu formátů umí jen číst, některé ani nečte
  - nepodporuje bucketing, ani indexy

PROFINIT

Hive vs Impala

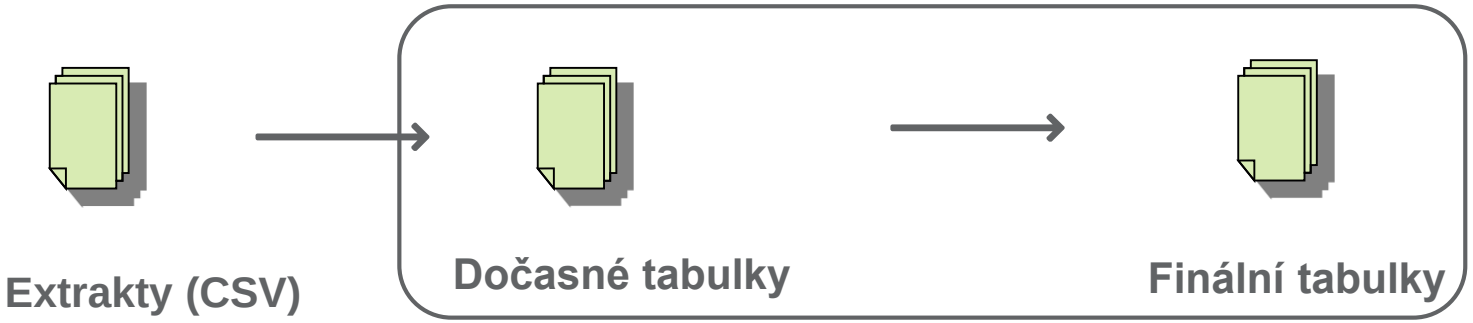
- Hive
  - je univerzálnější
  - má bohatší sadu funkcí
  - nahrávání dat je poměrně efektivní
  - při práci s petabajty dat asi jediná varianta
  - nové execution engines jsou výrazně rychlejší než MapReduce
  - „tlačí“ Hortonworks
- Impala
  - „uživatelsky“ přívětivější, rychlejší
  - orientace spíše na výkon, než na „feature set“
    - tj. můžeš si vybrat jakýkoli formát, pokud je to Parquet+Snappy
  - „tlačí“ Cloudera
- Jak se rozhodnout?
  - na loady dat nejspíše Hive
  - na analytiku podle použité distribuce...

Impala - architektura



PROFINIT

Hive – příklad



PROFINIT

Hive – příklad

- Vytvoříme externí tabulku vázanou na zdrojové soubory

```
CREATE EXTERNAL TABLE IF NOT EXISTS ap_temp (
    ACC_KEY BIGINT,
    BUS_PROD_TP_ID VARCHAR(255),
    START_DATE TIMESTAMP)
ROW FORMAT
    DELIMITED FIELDS TERMINATED BY '~'
    LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/data/input/acc';
```

Hive – příklad

- Vytvoříme prázdnou optimalizovanou tabulku

```
CREATE TABLE IF NOT EXISTS ap (
    ACC_KEY BIGINT,
    BUS_PROD_TP_ID VARCHAR(255),
    START_DATE TIMESTAMP)
PARTITIONED BY (BUS_PROD_TP_DESCR VARCHAR(255))
CLUSTERED BY (ACC_KEY) INTO 32 BUCKETS
STORED AS ORC tblproperties
    ("orc.compress"="ZLIB");
```

Hive – příklad

- Nahrajeme data do optimalizované tabulky

```
INSERT OVERWRITE TABLE ap
PARTITION (BUS_PROD_TP_DESCR)
SELECT
 ACC_KEY,
 BUS_PROD_TP_ID,
 START_DATE,
 BUS_PROD_TP_DESCR
FROM ap_temp;


DROP TABLE ap_temp;
```

HBase

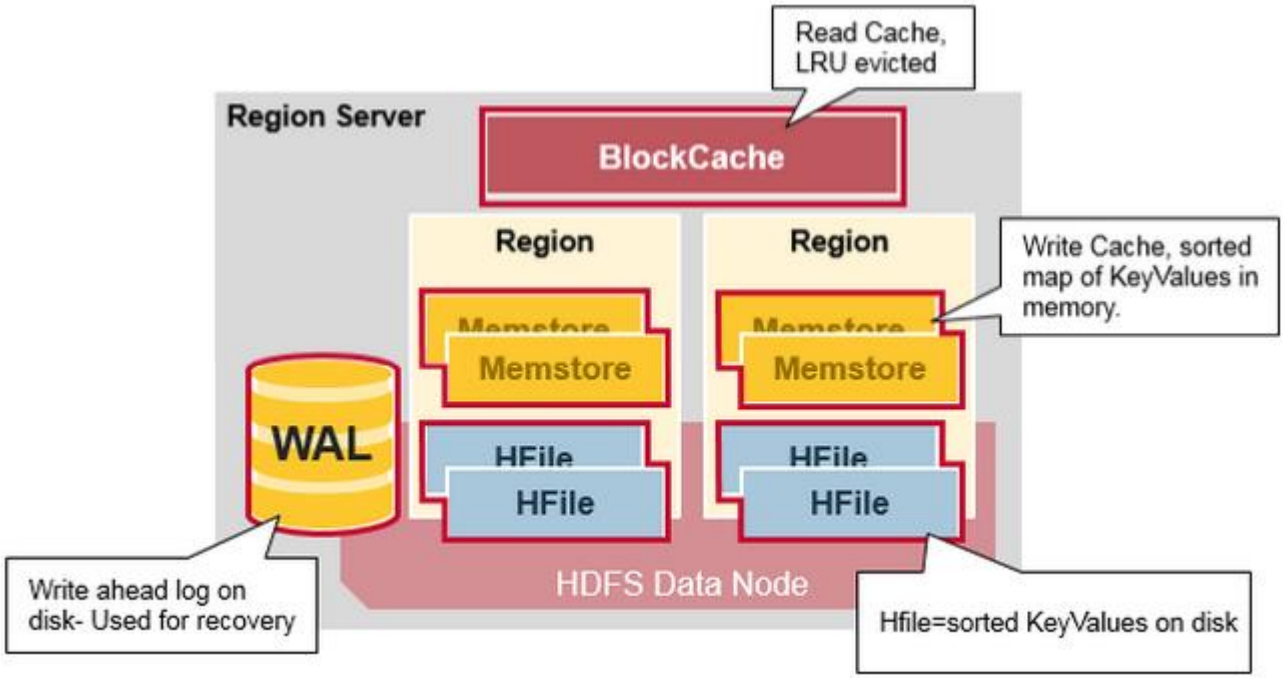

I WAS HOPING FOR A SLIGHTLY MORE DETAILED EXPLANATION OF HOW CLOUD COMPUTING WORKS THAN – "IT'S MAGIC"!
IT
© D.Fletcher for CloudTweaks.com

PROFINIT

HBase

- https://hbase.apache.org/
- NoSQL „databáze“
  - dotazování de facto programaticky (Java), žádný vhodný „SQL“ jazyk nemá
- Key/value storage
  - vhodná konstrukce klíče je základ pro použití HBase!
- Ukládá data na HDFS
- Velmi rychlý přístup k datům podle klíče
  - na rozdíl Impala a Hive random access!
- Velmi pomalý full scan
- Velmi dobrá horizontální škálovatelnost
  - cca 4000-5000 dotazů za sekundu per node
- Je třeba detailní znalost, při velkém množství dat je třeba správně nakonfigurovat
- Zajímavost: Využívá Facebook pro messaging

HBase – architektura



› https://www.mapr.com/blog/in-depth-look-hbase-architecture

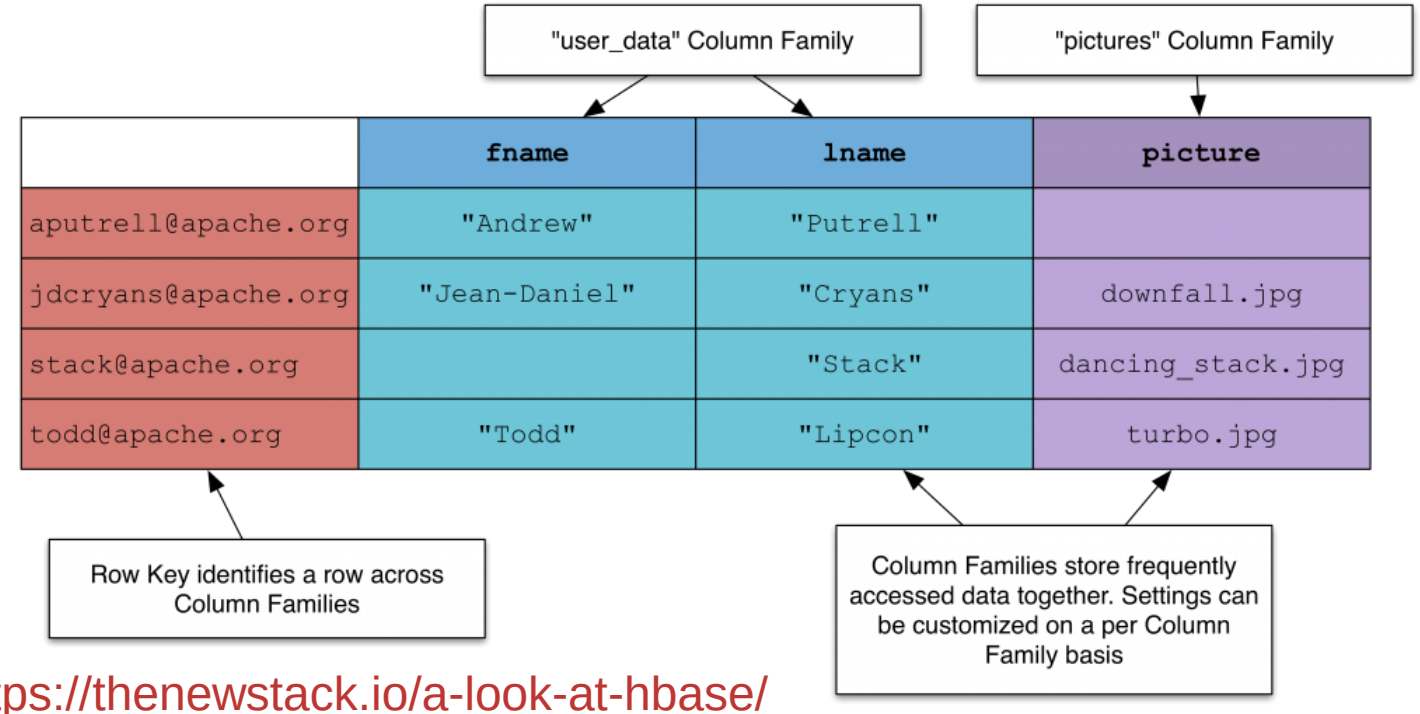
PROFINIT

HBase – použití

- Vhodné použití
  - jednoduché dotazy, hodně jednoduchých dotazů
  - více se čte, než zapisuje
  - odpověď je třeba velmi rychle (stovky ms)
  - k datům se přistupuje jen podle klíče, příp. počáteční části klíče
  - není třeba načítat velké množství dat sekvenčně
- Nevhodné
  - analytické dotazy
  - průchod daty/scan
  - pouze zápisy (nebo nepoměrně mnoho zápisů vůči čtení)

- Základem je **konstrukce klíče**, podle kterého se dotazuje
- Lze použít tam, kde je jasně definovaný use-case
  - na HBase nelze snadno stavět obecná/flexibilní řešení

HBase – příklad



https://thenewstack.io/a-look-at-hbase/

- Column family
  - skupina sloupců, které spolu souvisí – často se načítají společně
  - zajištěno, že v HDFS jsou uloženy společně

PROFINIT

HBase – příklad

```
public class RetriveData{
   public static void main(String[] args) throws IOException,
Exception{
      // Instantiating Configuration class
      Configuration config = HBaseConfiguration.create();
      // Instantiating HTable class
      HTable table = new HTable(config, "cli");
      // Instantiating Get class
      Get g = new Get(Bytes.toBytes("stack@apache.org"));
      // Reading the data
      Result result = table.get(g);
      // Reading values from Result class object
      byte [] value =
result.getValue(Bytes.toBytes("user_data"),Bytes.toBytes("lname"));
   }
}
```

PROFINIT

Shrnutí
+ Co se jinam nevešlo

Schema evolution

- Změna schématu souboru v průběhu života
- Typicky chceme
  - přidat sloupec
- Někdy je i možné
  - přejmenovat sloupec
  - odstranit sloupec (zpravidla jen v metadatech)
- Podpora schema evolution záleží na formátu souboru i použitém nástroji – vždy je třeba nastudovat
- Podporované formáty alespoň pro přidávání sloupců
  - Avro
  - ORC
  - Parquet

Transakce

- Obecně v Hadoopu téměř není
- **Otázka k diskusi: Je ale vůbec třeba při zpracování velkých dat?**

- Velmi limitovaná podpora:
  - Lze řešit přegenerováním celé tabulky nebo partition
  - ORC + Hive
    - https://orc.apache.org/docs/acid.html
    - https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions
  - HBase
    - ACID na úrovni řádku
    - https://hbase.apache.org/acid-semantics.html

Shrnutí

- Vždy se rozmýšlet, co je třeba, podle požadovaného použití
  - každý nástroj nabízí jinou míru flexibility!
- Snažit se maximálně těžit ze sloupcově orientovaných úložišť
  - např. analytické „SQL-like“ úlohy
- Použít řádkově orientované úložiště tam, kde je nutný full scan
- Soubory v databázi Hive je třeba mít rozumně velké – ideálně velikost HDFS bloku (po kompresi)
- Partitioning je základ optimalizace v Hive/Impala
  - ale nepřehánět – je dobré mít zhruba odhad velikosti partition
- Hive i Impala jsou dobré nástroje
  - na loady dat Hive
  - na analytiku se rozhodnu podle distribuce (Impala v případě Cloudery)
- HBase používat jen ve velmi specifických případech
  - vyžaduje znalosti, zkušenosti a testování

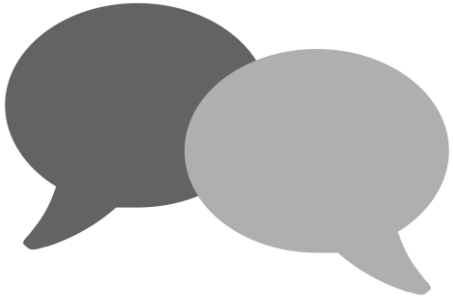
Praxe

- Souborové formáty
  - primárně Parquet, méně často pak Avro
  - na vstupu CSV
- Komprese
  - Parquet – účinnost kolem 50%
  - GZIP – vstupní data
- Hive
  - nahrávání dat, ETL
- Impala
  - analytické dotazy

- Pracujeme s tabulkami obsahujícími cca 2-10 mld záznamů/jednotky TB
  - složitý analytický dotaz běží cca 2min.
  - na relační DB stejný dotaz trval 40min.

PROFINIT

Diskuze



milan.kratochvil@profinit.eu

PROFINIT

Díky
za pozornost

PROFINIT
NÁSKOK DÍKY ZNALOSTEM

Profinit EU, s.r.o.
Tychonova 2, 160 00 Praha 6 | Telefon + 420 224 316 016

Web
www.profinit.eu

LinkedIn
linkedin.com/company/profinit

Twitter
twitter.com/Profinit_EU

Facebook
facebook.com/Profinit.EU

Youtube
Profinit EU