NÁSKOK
DÍKY
ZNALOSTEM

PROFINIT

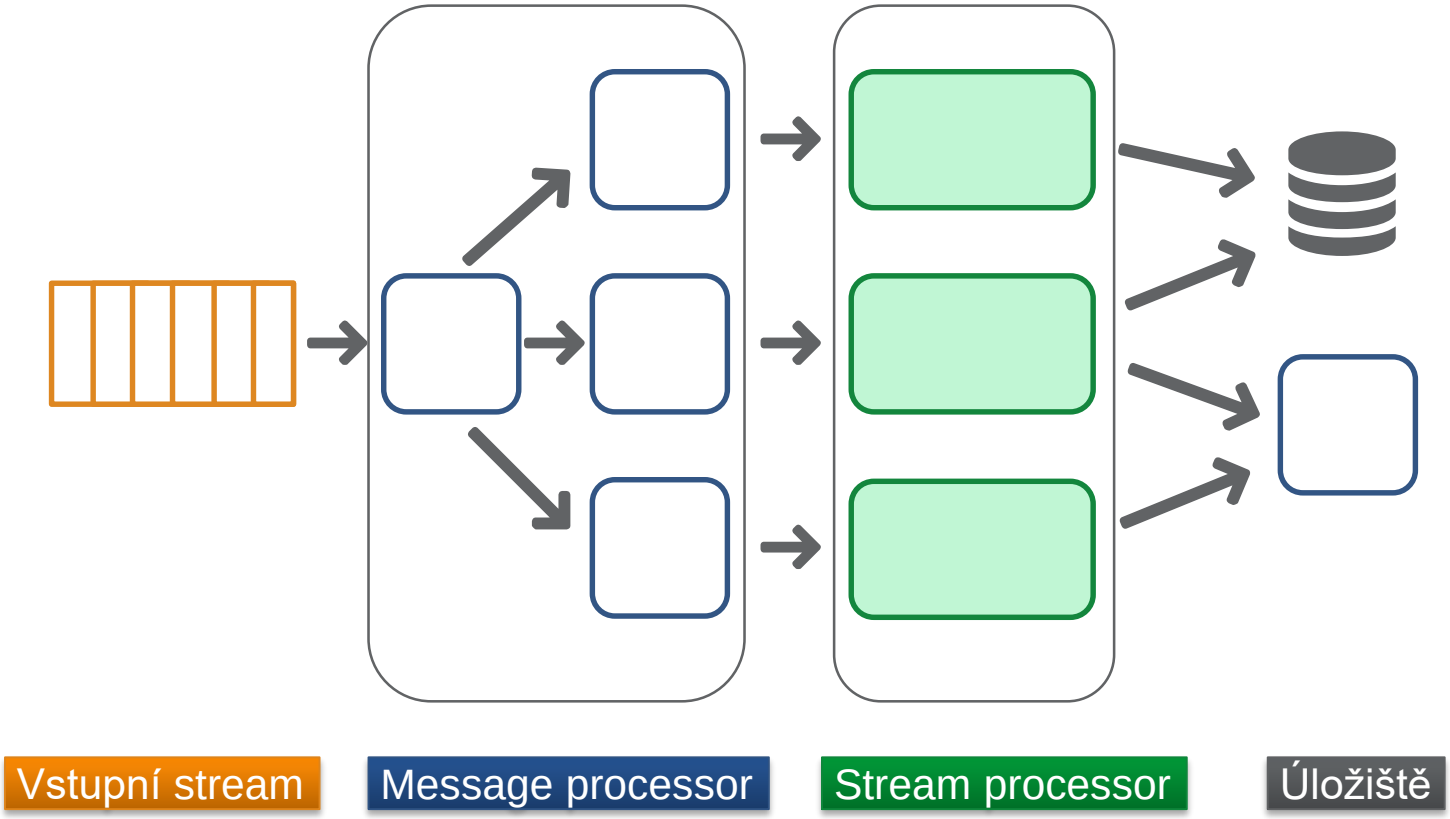
B0M33BDT
Stream processing

Milan Kratochvíl, Sergii Stamenov

9. prosinec 2020

Stream processing

- Průběžné zpracování trvalého toku zpráv



Stream processing

- Hlavní komponenty
  - Message processor – vstup dat do streamového zpracování
  - Stream processor – jádro zpracování údajů
  - Úložiště výstupů – persistentní úložitě zpracovaných dat

- Životní cyklus zpráv
  1. Příjem zpráv
  2. Rozdělení zpráv do partitions
  3. **Zpracování zpráv (logika aplikace)**
  4. Uložení, notifikace výsledků atd.

Stream processing vs batch processing

Batch processing

- Zpracování velkého množství dat najednou
- **Pořadí nutno vyčíst z dat**
- Zpracování s velkým zpožděním (denní, hodinové...)
- Výsledky z principu nelze poskytovat „online“
- Efektivní využití zdrojů (paměť, CPU)
- Zpracuje enormní množství libovolných dat (petabajty)
- Hadoop založen na batch zpracování

Stream processing

- Zpracování záznam za záznamem
- **Zaručené pořadí**
- Zpracování ihned po příchodu zprávy nebo s malým zpožděním
- Výsledky jsou často dostupné „online“
- Náročnější na zdroje (**paměť**, CPU)
- Zpracování složitých dotazů na velkém množství dat nemusí být efektivní
- Pro Hadoop relativně nové

PROFINIT

Messaging processor

Apache Kafka

- Distribuovaný systém pro zpracování datových streamů
- Typicky se používá jako messaging systém
- Transakční log

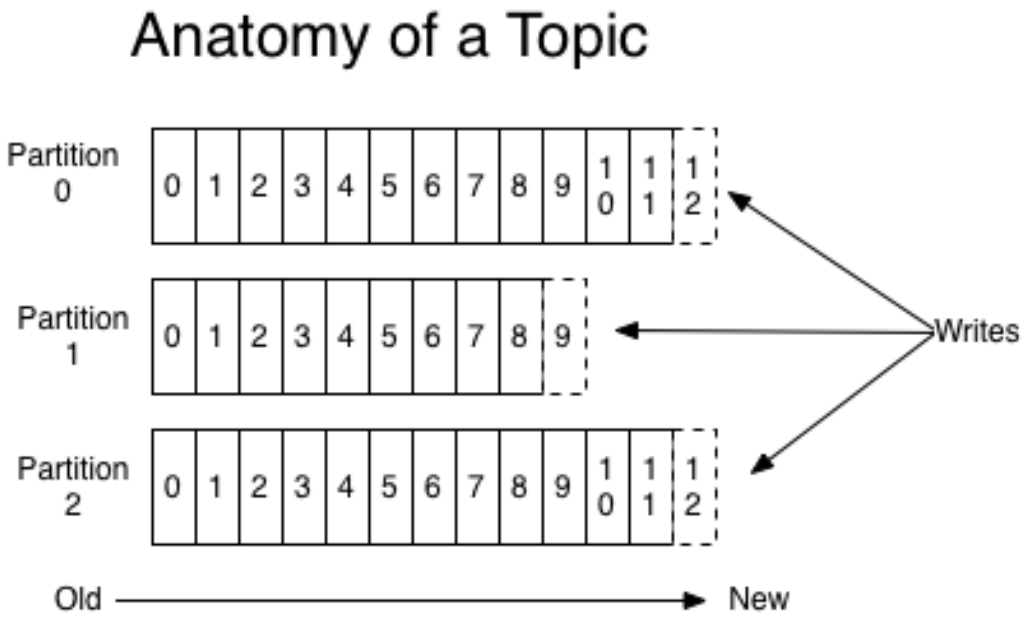
- Základní vlastnosti
  - vysoce výkonný (zpracuje i miliony zpráv za sekundu)
  - distribuovaný
  - zajišťuje replikaci dat
    - nevyžívá HDFS, má vlastní způsob ukládání dat
  - schopnost řešit výpadky v clusteru
  - škálovatelný
    - lze snadno přidat nový node pro zvýšení propustnosti

PROFINIT

Základní pojmy

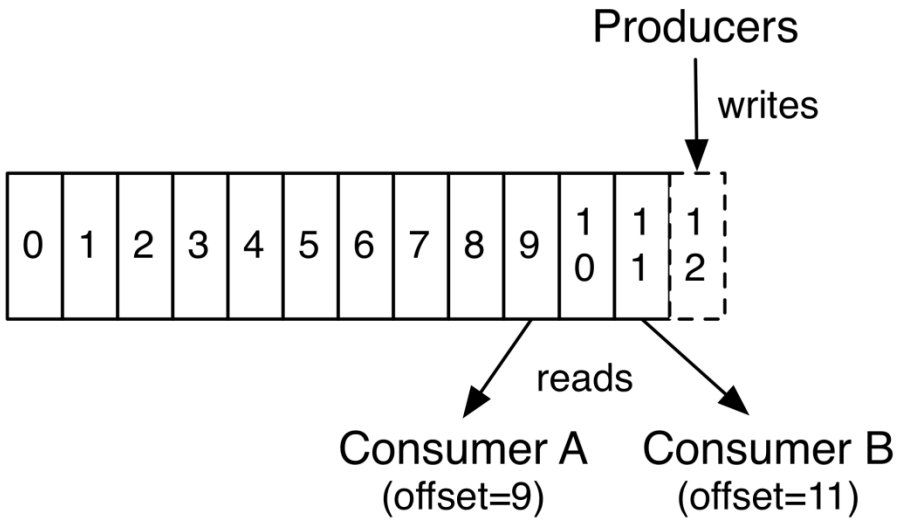
- Topic
  - pojmenovaná „fronta zpráv“
- Partition
  - dělení topiku na menší části
- Offset
  - aktuální pozice v topiku/paritition (orientace, kde se v topiku nacházíme)
- Consumer
  - odběratel dat
- Consumer Group
  - skupina odběratelů dat
- Producer
  - tvůrce dat

Základní pojmy



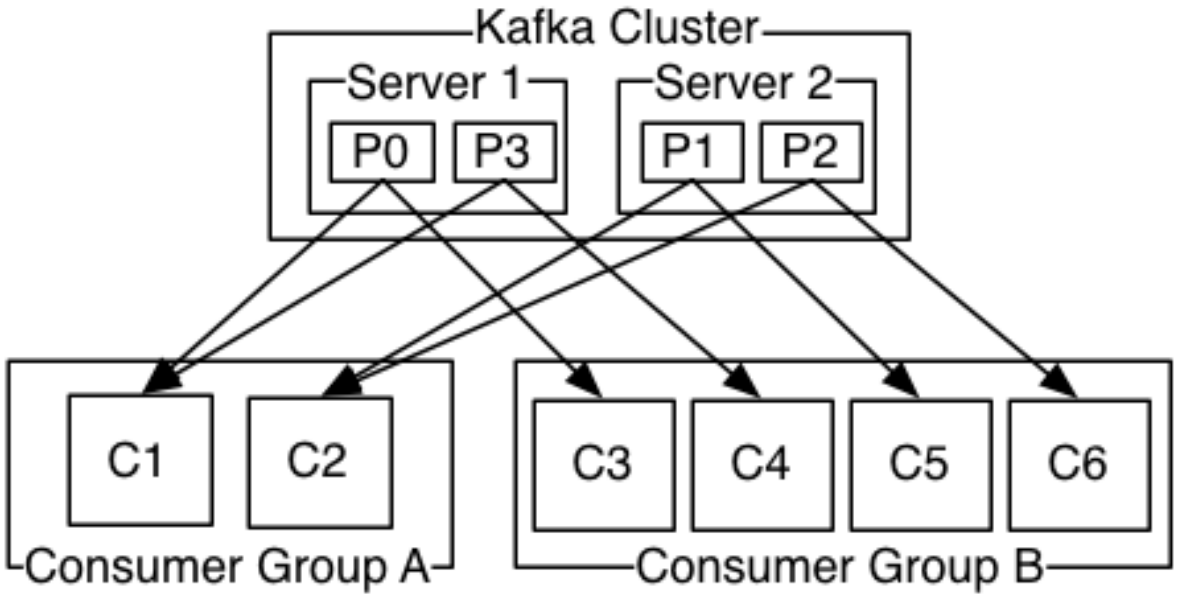
PROFINIT

Základní pojmy



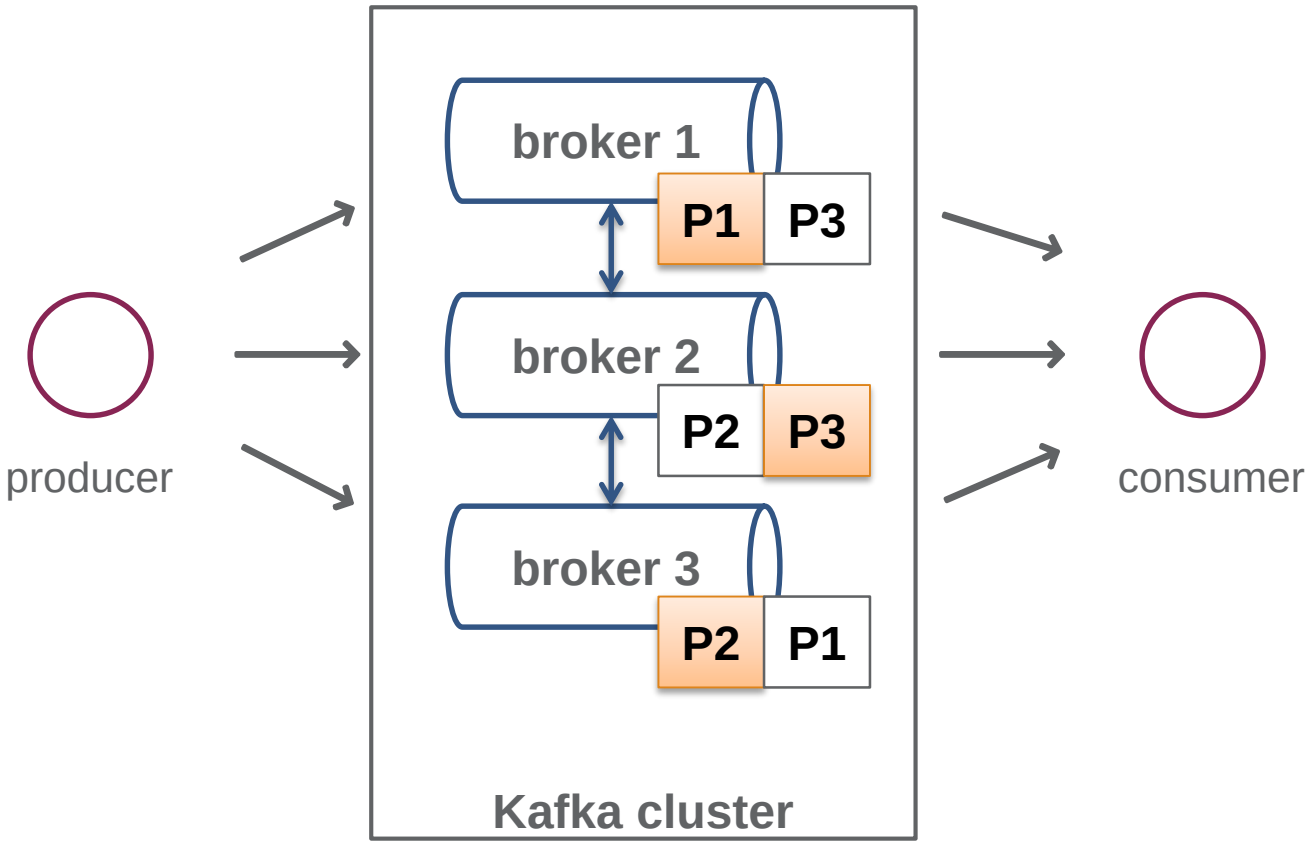
PROFINIT

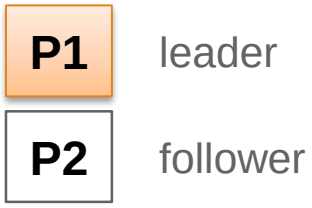
Základní pojmy



PROFINIT

Zápis a čtení





PROFINIT

Mathias Verraes
@mathiasverraes

Follow

There are only two hard problems in distributed systems: 2. Exactly-once delivery 1. Guaranteed order of messages 2. Exactly-once delivery

RETWEETS 6,775 LIKES 4,727

10:40 AM - 14 Aug 2015

69 6.8K 4.7K

PROFINIT

Pořadí zpráv v Kafce

- Pořadí zpráv
  - garance doručení zpráv v pořadí, v jakém byly zapsány **per partition**
  - ale ne v rámci všech zpráv daného topiku!
- Standardně se zprávy rozdělují rovnoměrně mezi jednotlivé partitions (náhodně)
- Typicky ale potřebuji zajistit pořadí zpráv jen např. pro klienta, zařízení... ➔ možnost definovat vlastní pravidla partitioningu

PROFINIT

Další možnosti Kafky

- Komprese zpráv
- Automatická retence zpráv
  - staré zprávy automaticky maže po uplynutí definovaného období
- Obsahuje vlastní systém pro zpracování streamů – Kafka Streams
- KSQL
  - SQL-like jazyk pro přístup k datům Kafky
- Základní transakční zpracování
  - consumer commituje poslední zpracovaný offset

Použití Kafky

- Všude tam, kde se komunikuje prostřednictvím zpráv, tj. skoro všude
  - senzorická data
  - finanční transakce
  - burzovní informace
  - logy
- Kappa architektura
- Lambda architektura
- Kafka nabízí velkou propustnost a robustnost, ale někdy za cenu vyšších latencí
  - obecně je třeba počítat s desítkami ms latencí jako minimum
  - lze optimalizovat na úkor propustnosti a bezpečnosti (konzistence) dat

Stream processor

Charakteristiky streamového zpracování

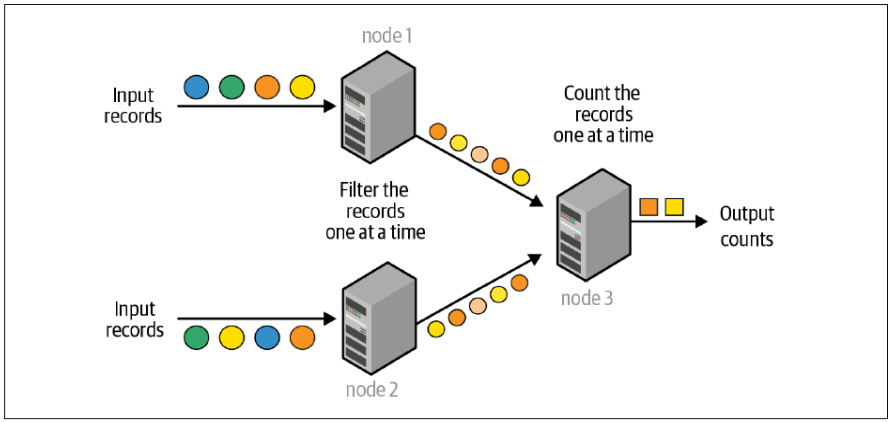
- Stream je analogie („nekonečné“) tabulky
- Streamy lze partitionovat
  - **paralelizace**
- Streamy je možné
  - číst
  - zapisovat
  - **joinovat**
- Často je potřeba udržovat stav (typicky agregace)
  - např. suma obratů na účtu, průměrná hodnota konkrétního senzoru...
- Práce s časovými okny
  - vyhodnocování úseků dat

Druhy streamového zpracování

- Podle doby zpracování
  - Real time
    - reakce na vstupní zprávu je typicky dokončena v řádu jednotek až stovek milisekund
  - Near-real time
    - reakce na vstupní zprávu je typicky dokončena v řádu jednotek až desítek sekund

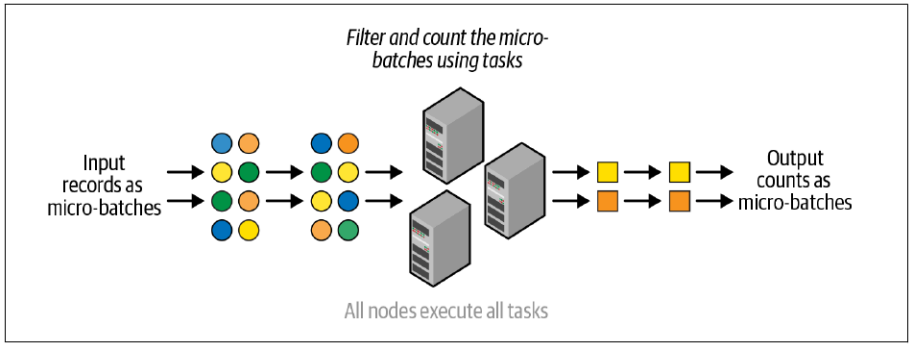
- Podle technologie zpracování zpráv
  - Real time streaming
    - skutečné zpracování jednotlivých zpráv, jak přicházejí jedna za druhou
  - Micro-batches
    - sekvenční spouštění malých dávek,
    - tj. nezpracovávají se zprávy ihned po příchodu, ale nejprve se nahromadí malá množina dat a ta se zpracuje jako celek

Real time streaming vs micro batches



Real time streaming

- každá zpráva je zpracována nezávisle



Micro batches

- zprávy jsou zpracovány v (malých) dávkách najednou

PROFINIT

Real time streaming vs micro batches

- Zpracování záznamu po záznamu
- Minimální latence
- Menší prostupnost (průměrný počet zpráv za sekundu)
  ➔ neustále se zlepšuje
- Vyžaduje pro realtime zpracování a batch zpracování samostatný kód
- Experimentalní podpora ve Spark Structured Streaming od verze 2.3

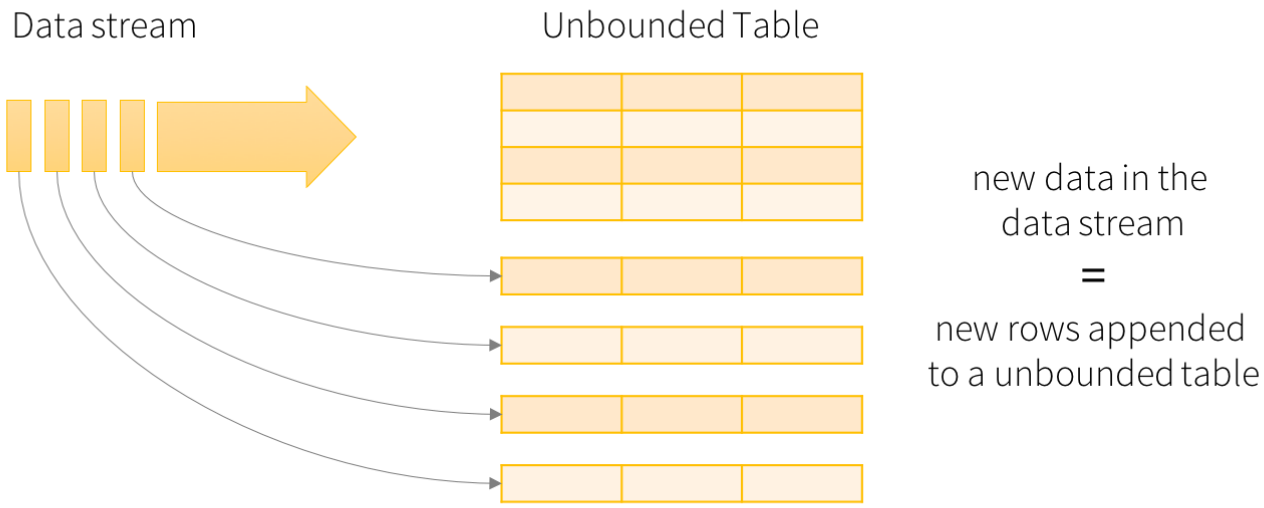
- Zpracování množství záznamů najednou
- Latence nejméně délka batch
- Typicky vyšší prostupnost
- Lze použít stejný kód pro streamové zpracování micro batches i pro „velké“ batches

PROFINIT

Streaming ve Sparku

- Spark má dvě implementace streaming
  - **DStreams** – RDD streaming API, dnes už se nepoužívá.
    - Low-level API, když to napíšete neoptimálně, spark vám nepomuže
    - Není jednoduché přepoužit batch kód
    - Chudá integrace s Kafka
    - Jenom processing-time okenka
  - **Spark Structured Streaming** – streaming pomoci Spark SQL API, je v aktivním rozvoji
    - Stejné API pro batch/streaming
    - Optimalizace exekučních planu stejně jako u Spark SQL
    - Podpora event-time oken

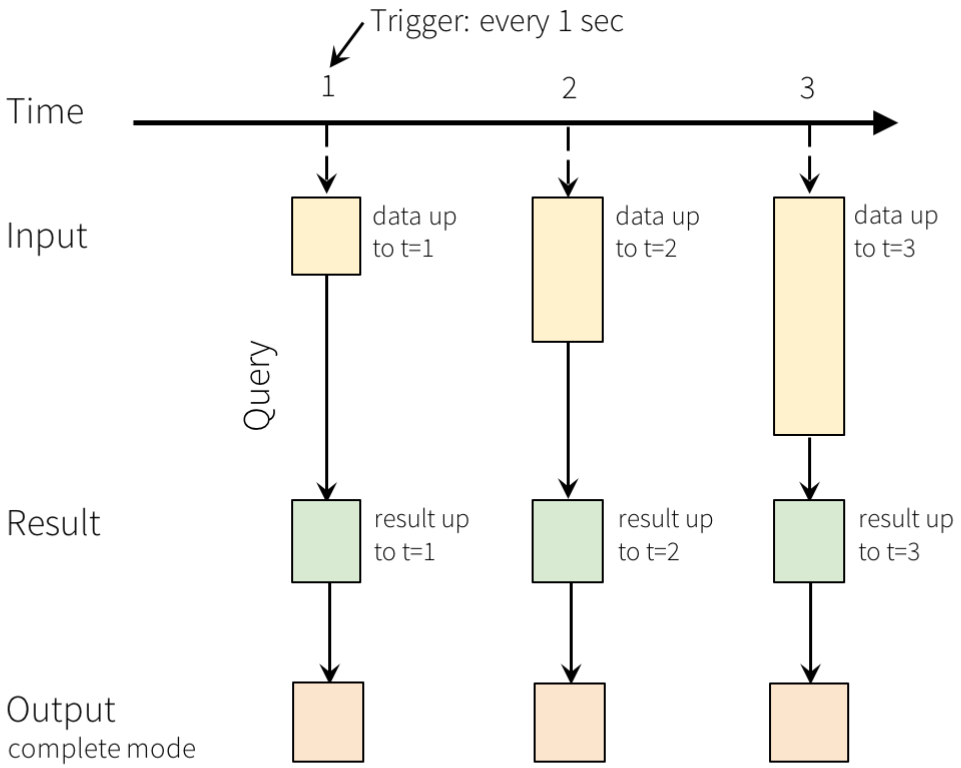
Stream jako nekonečná tabulka



Data stream as an unbounded table

PROFINIT

Diskretizace streamu



Programming Model for Structured Streaming

PROFINIT

Output modes

- Append mode – stream zapíše na výstup jenom nové řádky.
- Update mode – stream zapíše na výstup řádky co se změnily od posledního triggeru.
- Complete mode – stream zapíše na výstup všechny řádky. (Pouze agregační streaming dotazy)

Zdroje dat v Spark Structured Streaming (Source)

- File source

```
userSchema = StructType().add("name", "string").add("age", "integer")
csvDF = spark \
    .readStream \
    .option("sep", ";") \
    .schema(userSchema) \
    .csv("/path/to/directory")  # Equivalent to format("csv").load("/path/to/directory")
```

- Socket source

```
# Read text from socket
socketDF = spark \
    .readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", 9999) \
    .load()
```

- Kafka source
- Rate source

Výstupy (Sink)

- File sink (pouze append)

```
writeStream
    .format("parquet")        // can be "orc", "json", "csv", etc.
    .option("path", "path/to/destination/dir")
    .start()
```

- Kafka sink
- Console sink

```
writeStream
    .format("console")
    .start()
```

- Foreach sink

```
writeStream
    .foreach(...)
    .start()
```

PROFINIT

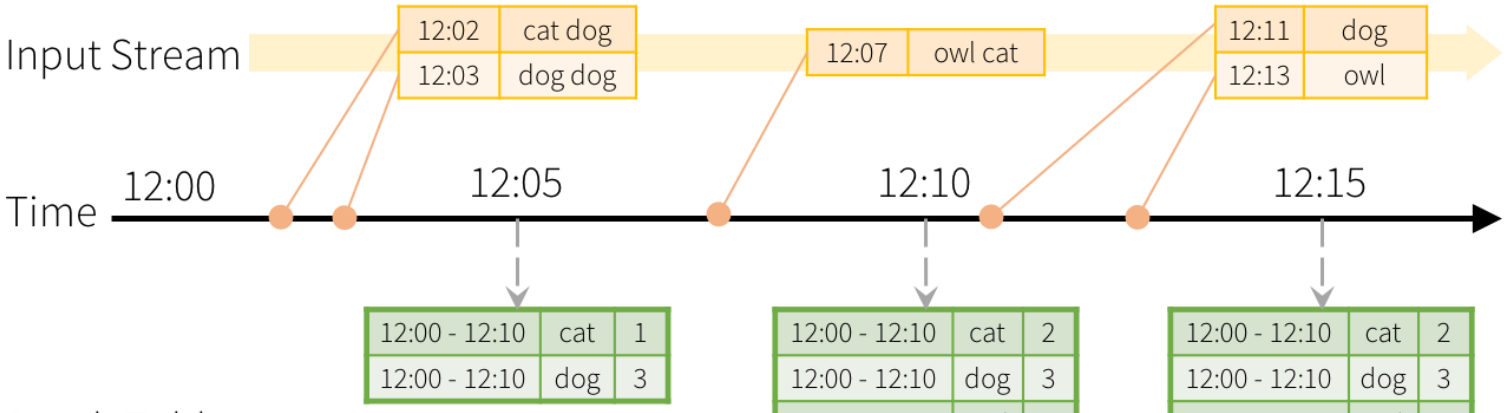
Operace nad streaming dataframe

- Většina operací funguje stejně jako u klasického dataframe
  - Select, where, groupBy, …
  - Dočasné tabulky + SQL
- Window operace nad Event Time
  - Lze zvolit délku okna a posuv nad sloupcem reprezentující čas (zde “timestamp”, groupuje se nad sloupcem “word”)

```
windowedCounts = words
    .groupBy(
        window(words.timestamp, "10 minutes", "5 minutes"),
        words.word
    )
    .count()
```

PROFINIT

Operace nad streaming dataframe - windows



12:00 - 12:10	cat	1
12:00 - 12:10	dog	3

Result Tables
after 5 minute triggers

12:00 - 12:10	cat	2
12:00 - 12:10	dog	3
12:00 - 12:10	owl	1
12:05 - 12:15	cat	1
12:05 - 12:15	owl	1

counts incremented for windows
12:00 - 12:10 and 12:05 - 12:15

12:00 - 12:10	cat	2
12:00 - 12:10	dog	3
12:00 - 12:10	owl	1
12:05 - 12:15	cat	1
12:05 - 12:15	owl	2
12:05 - 12:15	dog	1
12:10 - 12:20	dog	1
12:10 - 12:20	owl	1

counts incremented for windows
12:05 - 12:15 and 12:10 - 12:20

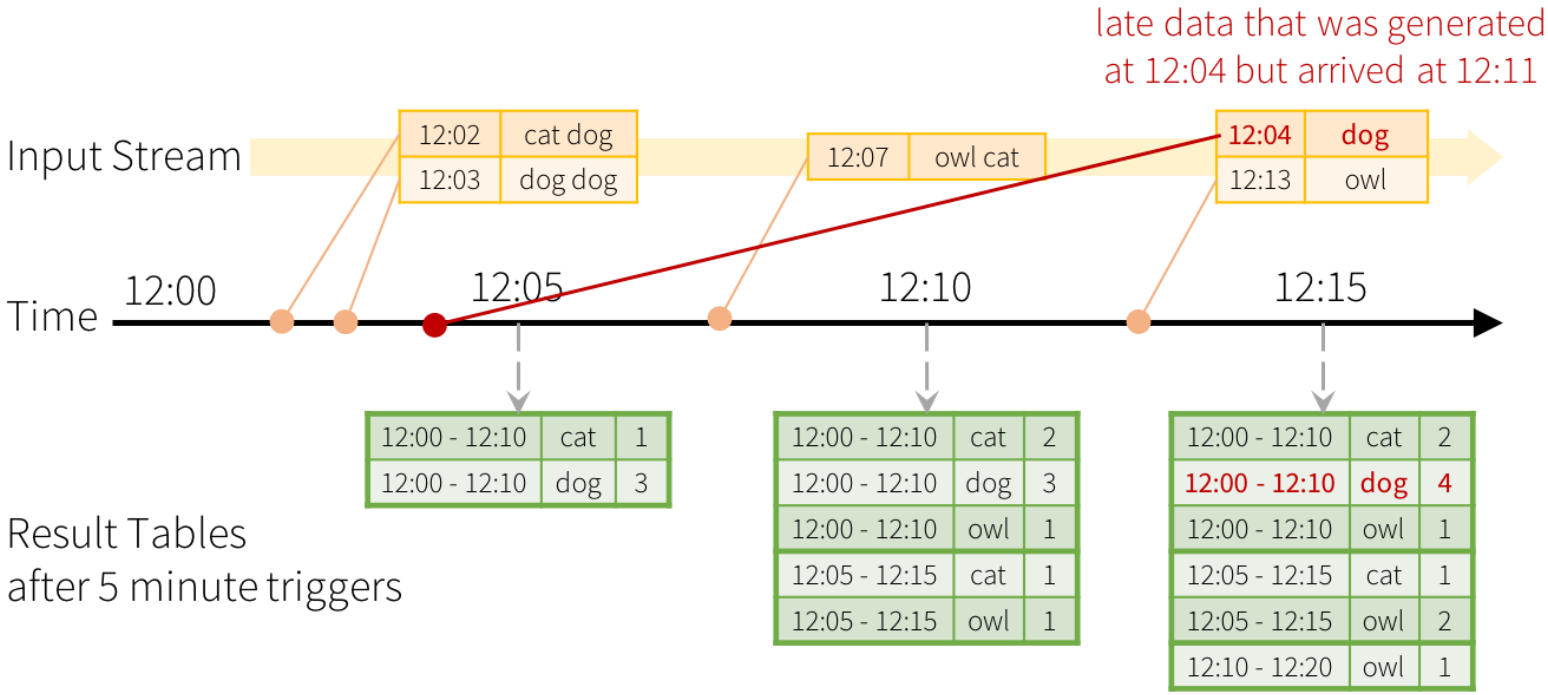
Windowed Grouped Aggregation
with 10 min windows, sliding every 5 mins

PROFINIT

Operace nad streaming dataframe - windows

- Pokud příjde zpráva mimo pořadí (mimo okno)
  - Spark správně provede agregaci
- Nicméně pro dlouhodobě spuštěné aplikace může být problém s pamětí
  - Drží všechen stav v paměti !

Operace nad streaming dataframe - windows



Late data handling in
Windowed Grouped Aggregation

PROFINIT

Operace nad streaming dataframe - windows

- Aby správně fungovalo čištění stavů u agregačních funkcí, zavádí se tzv. watermark
  - Maximální stáří, do kterého bude zpráva ještě zpracována (jinak se zahodí)
- Podmínky watermarkingu
  - Streaming musí běžet v **Update** nebo **Append** modu
  - Daná agregace musí mít buď sloupec s event-time nebo definované okno nad event-time sloupcem
  - Watermark musí mít specifikovaný stejný sloupec jako daná agregace
  - withWatermark musí být uveden před danou agregací

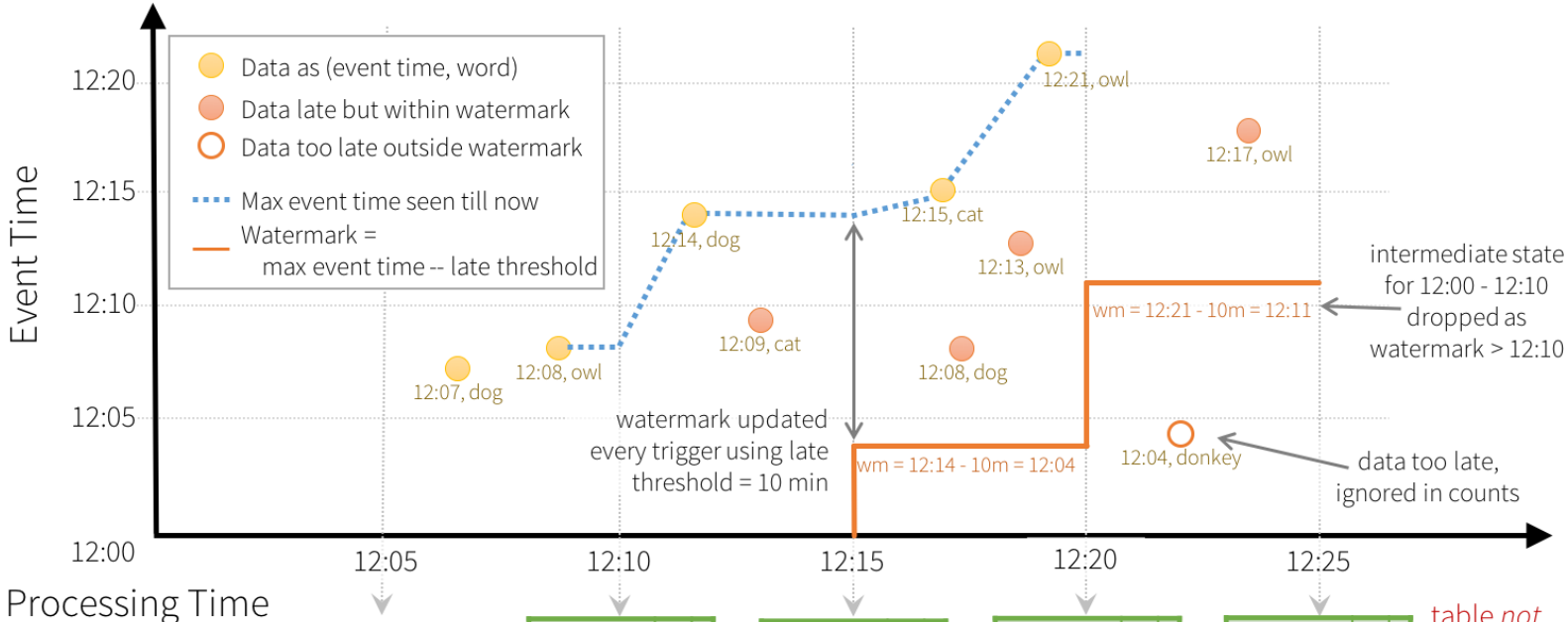
```
windowedCounts = words
    .withWatermark("timestamp", "10 minutes")
    .groupBy(
        window(words.timestamp, "10 minutes", "5 minutes"),
        words.word
    )
    .count()
```

PROFINIT

Operace nad streaming dataframe - windows

- Chování se liší v závislosti na výstupním módu
- Update Mode
  - Zapisuje na výstup nové / aktualizované agregace
- Append Mode
  - Zapisuje na výstup až výsledné agregace
  - Zápis až po uplynutí watermarku (čeká na eventuálě opožděné zprávy)

Operace nad streaming dataframe - windows



Result Tables after each trigger

12:00 - 12:10	owl	1
12:00 - 12:10	dog	1
12:05 - 12:15	owl	1
12:05 - 12:15	dog	1

12:00 - 12:10	owl	1
12:00 - 12:10	dog	1
12:00 - 12:10	cat	1
12:05 - 12:15	owl	1
12:05 - 12:15	dog	2
12:05 - 12:15	cat	1
12:10 - 12:20	dog	1

12:00 - 12:10	owl	1
12:00 - 12:10	dog	2
12:00 - 12:10	cat	1
12:05 - 12:15	owl	2
12:05 - 12:15	dog	3
12:05 - 12:15	cat	2
12:10 - 12:20	dog	1
12:10 - 12:20	cat	1
12:10 - 12:20	owl	1
...		

12:00 - 12:10	owl	1
12:00 - 12:10	dog	2
12:00 - 12:10	cat	1
12:05 - 12:15	owl	2
12:05 - 12:15	dog	3
12:05 - 12:15	cat	2
12:10 - 12:20	dog	1
12:10 - 12:20	cat	1
12:10 - 12:20	owl	2
...		

purple rows are updated rows that are written to the sink as output

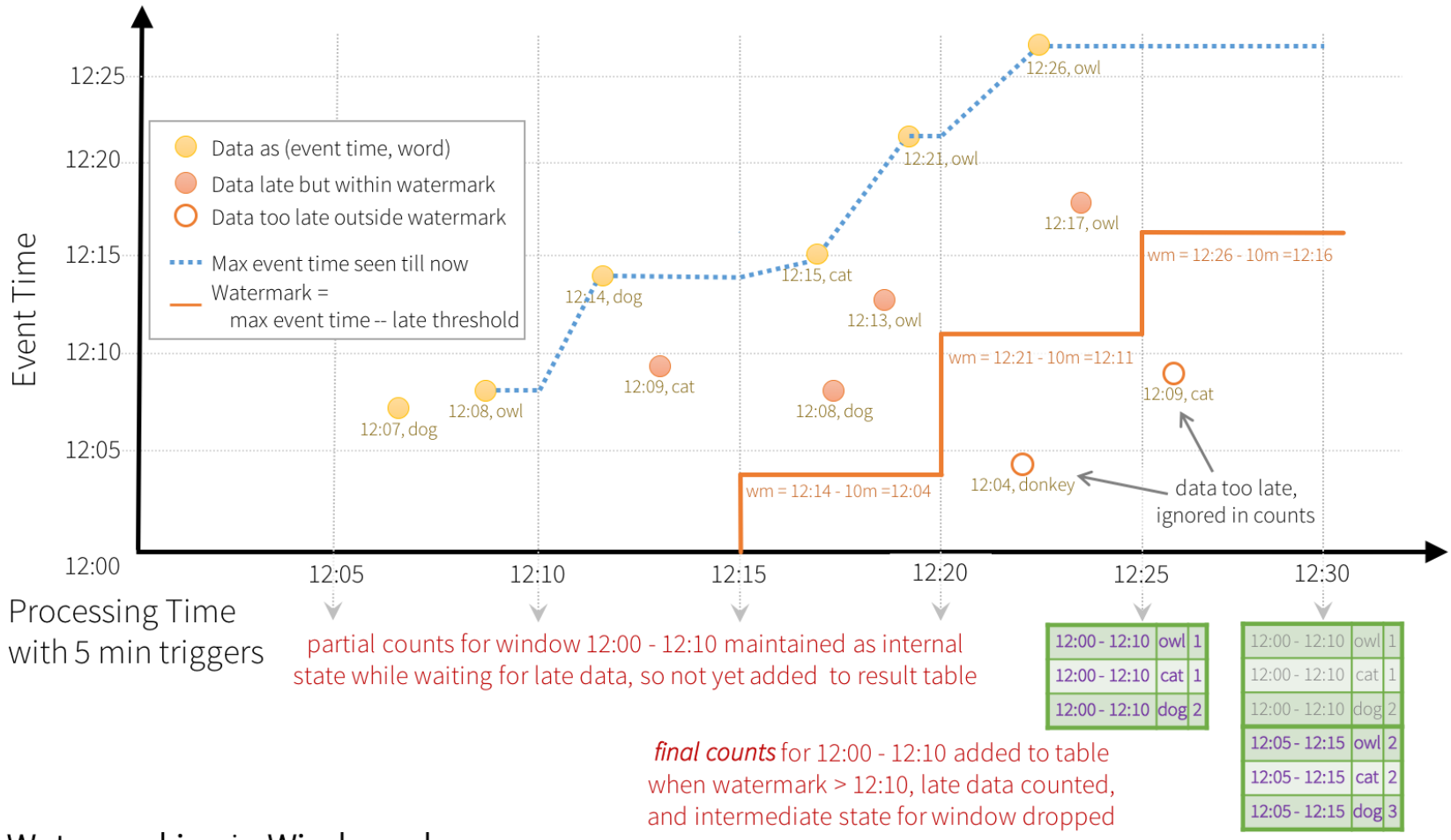
table *not* updated with too late data (12:04, donkey)

table updated with late data (12:17, owl)

Watermarking in **Windowed Grouped Aggregation** with **Update Mode**

PROFINIT

Operace nad streaming dataframe - windows



Watermarking in **Windowed Grouped Aggregation** with **Append Mode**

PROFINIT

Operace nad streaming dataframe - joins

- Joiny
  - Join se (statickým) dataframe (stream / static)
  - Join s dalším streaming dataframe (stream / stream)

- Stream / Stream join
  - Poměrně složité na implementaci
  - Potřeba držet stav obou streamů po definovanou dobu (často se uvádí watermark spolu s time range, popř. window)

- Ne všechny typy joinů jsou podporovány
  - Nutno nastudovat v dokumentaci

Nastavení triggeru

- Nastavení triggeru definuje způsob streamového zpracování
  - Micro-batch
  - Continous processing (**experimentální vlastnost**)
- Trigger lze nastavit
  - bez uvedení (micro-batch)
  - `processingTime` (definuje pevný interval micro-batche)
  - `once` (spustí jednorázově a zpracuje všechna data)
  - `continous` (continous processing)

PROFINIT

Sémantiky streamového zpracování

- Exactly once
  - za každých okolností zajistíme, že zpráva bude doručena
  - vyžaduje nějaký způsob checkpointování, aby se dalo zjistit, jaké zprávy byly zpracování, v případě, že dojde k havárii procesu
- At least once
  - zpráva se může doručit více než jednou
  - velmi častý kompromis
- At most once
  - zprávu nikdy nedoručíme opakovaně, může také nastat, že zpráva bude zcela ztracena
  - pouze pro nedůležitá data/data, která brzy ztrácejí cenu

Idempotence

- Nástroje pro stream zpracování obsahují možnosti, jak zajistit exactly once sémantiku v rámci jednoho procesu
- Ale je velmi složité zajistit toto v rámci více návazných procesů!

- **Idempotence je takové chování, kdy opětovné doručení totožné zprávy nezmění stav systému**

- Jak to řešit?
  - Např. pokud jediný výstup je databáze (HBase)
  - Doručení již existující zprávy způsobí uložení identických dat pod stejným klíčem (tj. systém není dotknut)

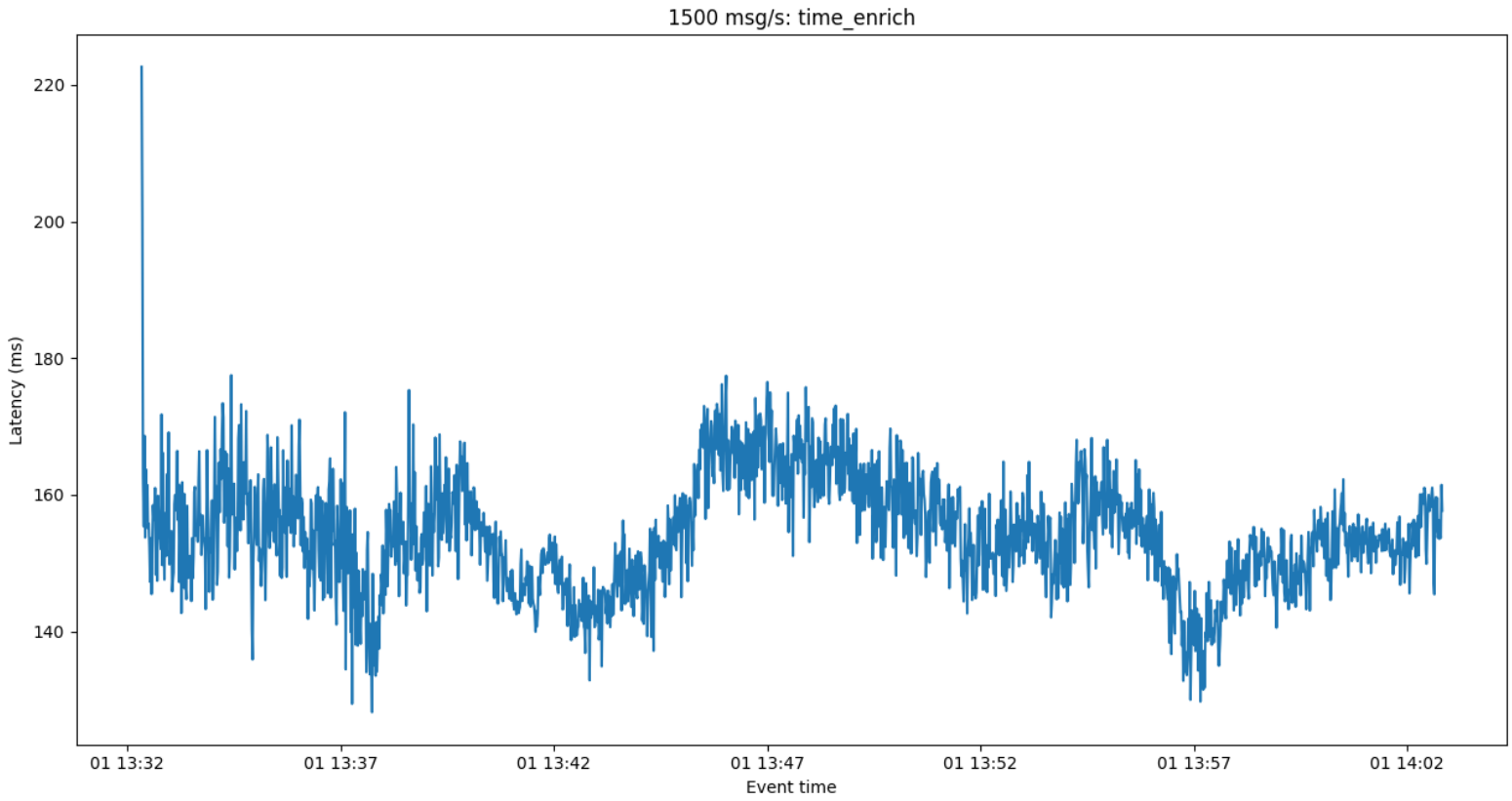
PROFINIT

Používané nástroj

- Apache Spark Streaming
  - pouze microbatche a velké batche
- Apache Flink
  - real time streaming,
  - podporuje i čistě batchové zpracování
- Apache Storm
  - real time streaming
- Kafka Streams

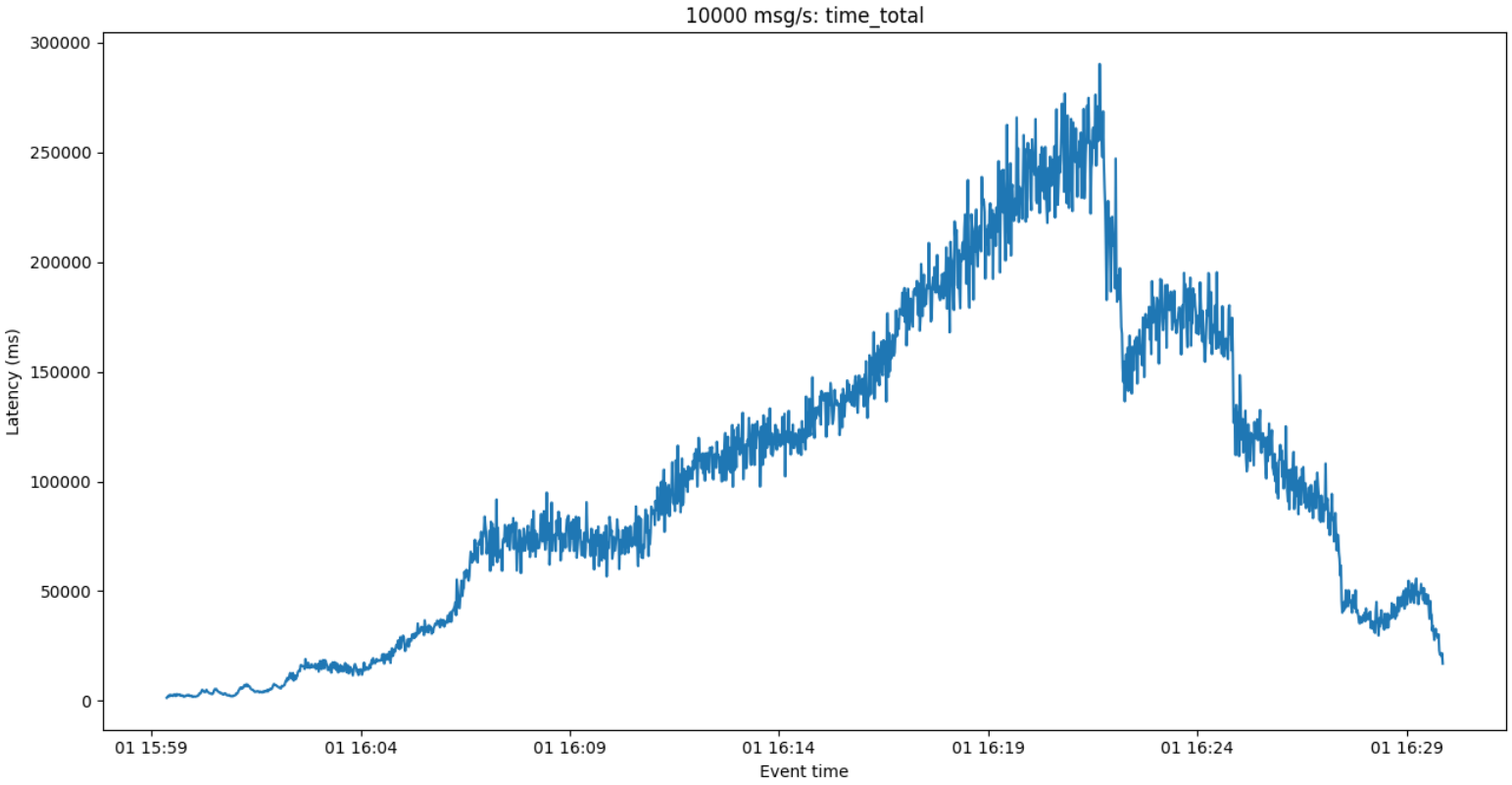
PROFINIT

Korektní průběh zpracování



PROFINIT

Kumulace zpoždění



PROFINIT

Úložiště - HBase

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 data model

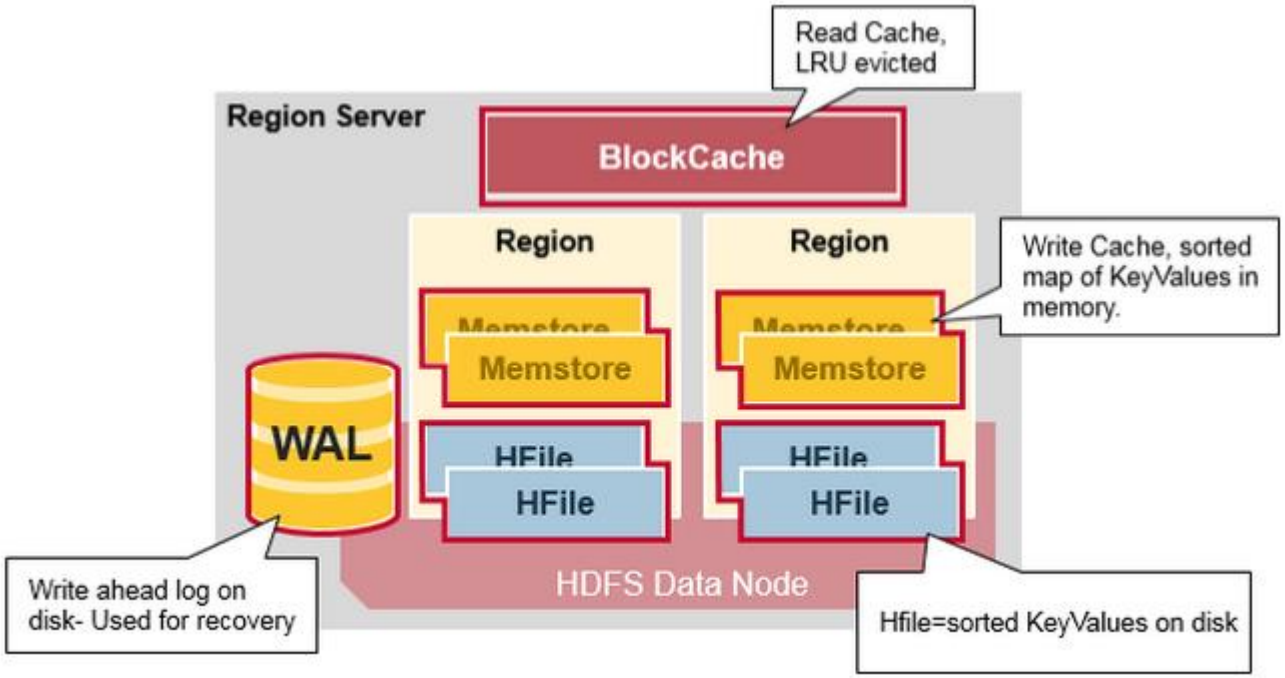
- Table
- Row
  - pro každý jednoznačný klíč
- Column Family
  - sloupce v jedné Column Family jsou vždy ukládány společně
  - naopak, velká data (např. obrázky), která se načítají zřídka, lze dát do jiné Column Family a omezit tím množství čtených dat
- Column
- Version

- HBase neobsahuje datové typy!

Operace

- Get
  - načtení záznamu podle klíče
- Put
  - uložení záznamu
- Scan
  - sekvenční načítání dat daného rozsahu klíčů nebo všech dat

HBase – architektura



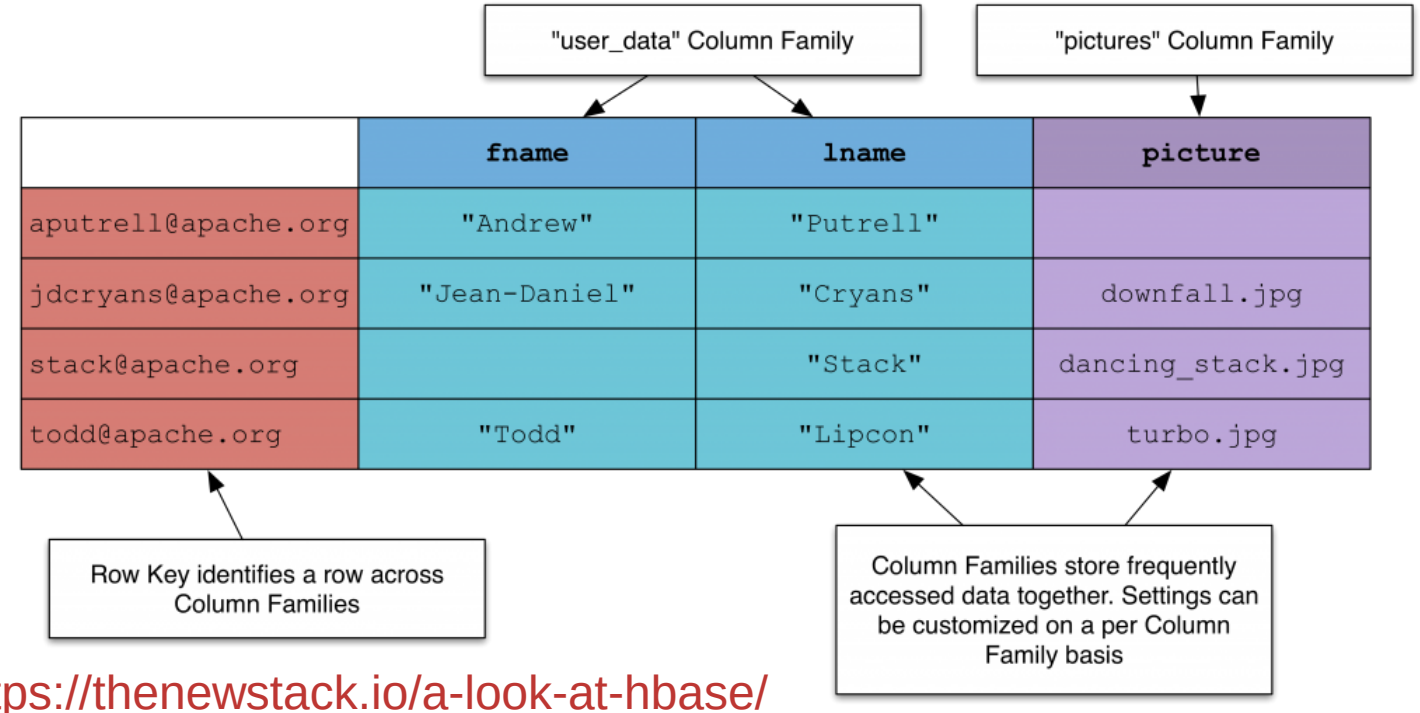
› https://www.mapr.com/blog/in-depth-look-hbase-architecture

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

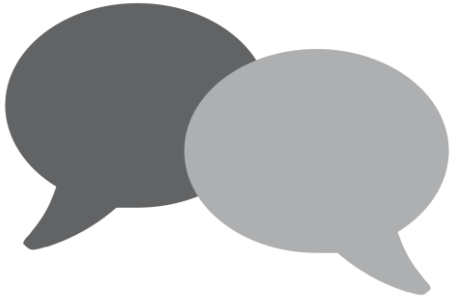
PROFINIT

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

Diskuze



sergii.stamenov@profinit.eu

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