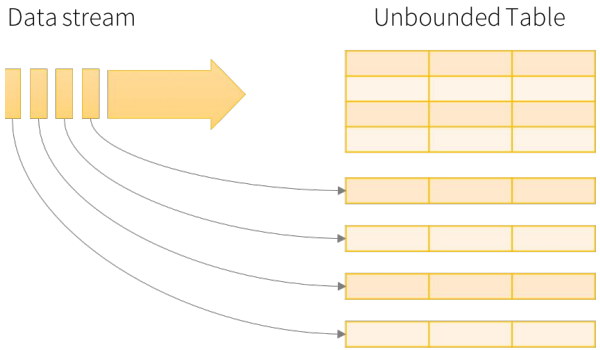
Apache
Spark Structured Streaming

Spark Structured Streaming - pregled

- *Spark Structured Streaming* je nadogradnja nad *Spark SQL engine*-om koja omogućava procesiranje tokova podataka uz skalabilnost, veliki protok (*high-throughput*) i otpornost na otkaze (*fault-tolerance*).
  - Nadogradnja aktuelna od *Spark* verzije 2.x;
  - Do verzije 2.3, bilo je moguće koristiti i *Spark Streaming* nadogradnju nad *Spark API*-jem
    - *Spark Streaming* pružao nivo apstrakcije koji se naziva diskretizovani tok (discretized stream) - DStream, koji predstavlja kontinuirani tok podataka. Interno, DStream je bio predstavljen pomoću sekvence RDD-eva
- Omogućava izražavanje željenih izračunavanja nad tokovima podataka na način kao da se radi paketna obrada
  - SQL engine je zadužen za inkrementalno i kontinualno pokretanje izračunavanja i za ažuriranje rezultata s pristizanjem novih podataka
- Pošto je zasnovan na SQL engine-u, možemo se osloniti na obradu tokova podataka uz oslonac na koncepte poput *DataFrame*-a i *DataSet*-a

Spark Structured Streaming - pregled

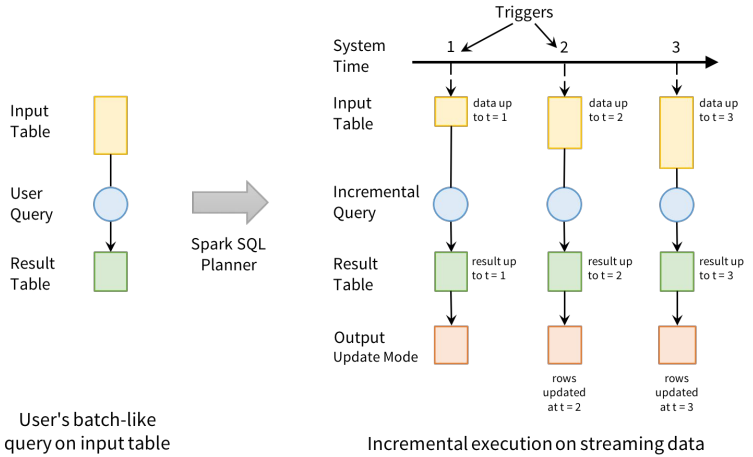
- Structured Streaming tok podataka posmatra kao neograničenu ***ulaznu tabelu***
  - svaki novi podatak posmatra se kao novi red ulazne tabele
- Vrši se **Micro-Batching** ulaznih podataka
  - u zadatim vremenskim intervalima, vrši se obuhvat novopristiglih podataka, i potom započinje proces obrade
  - Vremenski intervali se zadaju korišćenjem tzv. ***trigera***
    - Definišu fiksne vremenske intervale obuhvata
    - Ukoliko se triger ne specificira eksplicitno, novi *micro-batch* će biti automatski generisan odmah nakon završetka obrade nad prethodnim *micro-batch*-om
    - Od verzije 2.3 uveden eksperimentalni režim kontinualnog procesiranja - još uvek nije spreman za produkciju



Izvor: *Databricks*

Spark Structured Streaming - pregled

- Model obrade tokova podataka
  - Podaci pristižu u ***ulaznu tabelu***
  - Nad ulaznom tabelom se potom definišu ***upiti***, kao da je u pitanju statička tabela
    - *Spark SQL Planner* automatski određuje plan izvršavanja upita
      - Ovaj proces naziva se ***inkrementalizacija*** - potrebno je odrediti koji rezultati se moraju održavati kako bi se stanje moglo ažurirati nakon pristizanja novog podatka
  - Rezultati upita se smeštaju u ***tabelu rezultata***
  - Redovi tabele rezultata se prosleđuju na ***izlazni slivnik*** u skladu sa zadatim ***izlaznim režimom***



Structured Streaming Processing Model

Users express queries using a batch API; Spark incrementalizes them to run on streams

Izvor: *Databricks*

Spark Structured Streaming - primer prebrojavanja reči

- Prebrojavanje reči sa definisanog *Reddit Subreddit*-a
- Povezanje na *spark-master*
  - `sudo docker exec -it spark-master bash`
- Pokretanje primera
  - `./spark/bin/spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1 ./spark/primeri/kafka_wordcount.py`

Spark Structured Streaming - primer prebrojavanja reči

- Prebrojavanje reči sa definisanog *Socket*-a
- Pokretanje *NetCat* aplikacije:
  - `nc -vvl -p 5858`
- Pokretanje primera
  - `sudo docker exec -it spark-master bash`
  - `./spark/primeri/network_wordcount.py`
- ToDo:
  - postaviti izlazni režim na:
    - *update*
    - *append*

Spark Structured Streaming - pregled

- Izvori podataka
  - Nativno podržani Kafka, Fajlovi, Socket-i
  - Za custom:
    - https://stackoverflow.com/questions/47604184/how-to-create-a-custom-streaming-data-source

Spark Structured Streaming - pregled

- Izlazni režimi
  - Redovi tabele rezultata se prosleđuju na izlazni slivnik u skladu sa zadatim izlaznim režimom:
    - Režim ***append*** (podrazumevani režim): na slivnik će biti prosleđeni samo novopristigli redovi tabele rezultata
      - Može se primeniti samo na upite kod kojih se sadržaj tabele rezultata ne može promeniti. Svaki red se na izlazu može naći samo jednom.
    - Režim ***complete***: kompletna tabela rezultata će biti prosleđena na slivnik
      - Podržava upite koji sadrže agregacije
    - Režim ***update***: na slivnik će biti prosleđeni samo redovi tabele rezultata koji su ažurirani nakon prethodne obrade
      - Može se primeniti samo ukoliko izlazni slivnik dozvoljava ažuriranje sadržaja (poput relacione SQL tabele)
  - Zbog malo kompleksnije situacije sa podrškom različitih režima za različite primenjene operacije, proučiti dokumentaciju pri radu

Spark Structured Streaming - pregled

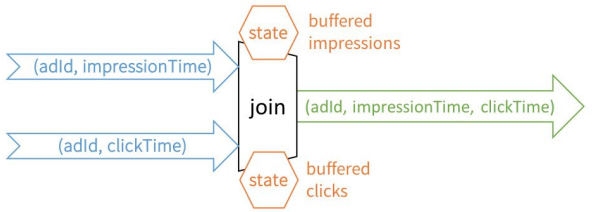
- Izlazni slivnici
  - Postoji nekoliko nativno podržanih izlaznih slivnika:
    - ***File sink*** - podaci se slivaju u zadati izlazni fajl
      - može se specificirati format fajla - Parquet, ORC, JSON, CSV, itd.
    - ***Kafka sink*** - podaci se slivaju na jedan ili više definisanih *Kafka topic*-a
    - ***Console sink*** - podaci se slivaju na standardni izlaz nakon svakog okinutog trigera
    - ***Memory sing*** - podaci se slivaju u tabelu u radnoj memoriji
  - Postoji mogućnost da se rezultati obrade slivaju i na skladišta podataka koja nisu nativno podržana
    - Ili uz korišćenje različitih biblioteka
    - Ili uz oslonac na korišćenje ***foreach*** i ***foreachBatch*** slivnika
      - Podržana primena proizvoljne logike za slanje podataka na izlazni slivnik
  - Za detalje oko podržanih izlaznih režima, opcija dostupne pri upotrebi izlaznog slivnika, kao i za detalje oko garantovane podrške za otkaze, pogledati zvaničnu dokumentaciju

Spark Structured Streaming - upiti i podržane operacije

- Podržane skoro sve standardne operacije koje se mogu izvršavati nad *Spark DataFrame/Dataset*-ovima
  - Podržane **standardne operacije** - selekcije, projekcije, spajanja, deduplikacije, agregacije, …
    - Omogućeno i kreiranje privremenih pogleda (*TempView*) za podršku korišćenju standarnih SQL naredbi
  - Podržane i operacije nad **kliznim vremenskim okvirima** (*Window* operacije)
  - Postoje i neki izuzeci - operacije koje nisu u potpunosti podržane

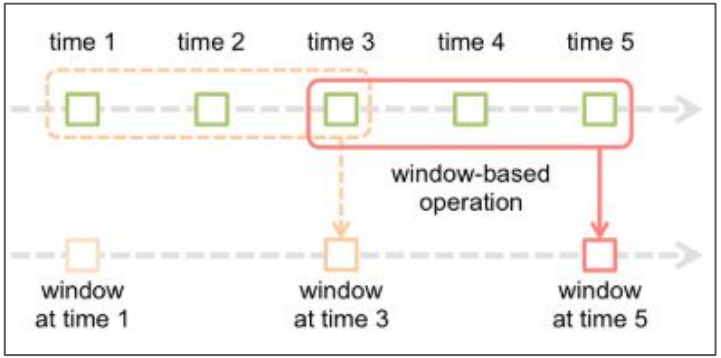
Spark Structured Streaming - upiti i podržane operacije

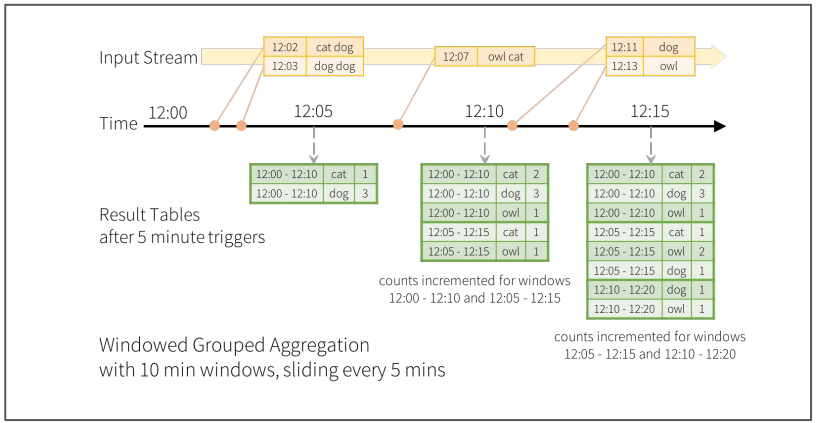
- Operacije spajanja podataka
  - Podržane operacije spajanja tokova podataka i sa statičkim podacima, i sa drugim tokovima podataka
  - Prilikom spajanja dva toka podataka mora se konstantno održavati stanje tokova
    - Ovo može imati za posledicu nekontrolisani porast potrebne memorije za čuvanje stanja
    - Dobra praksa - uvesti vremensko ograničenje prilikom spajanja, kako bi se deo memorije mogao osloboditi kada više ne ispunjava zadato ograničenje
- Operacije deduplikacije
  - Moguća je primena standardne deduplikacije podataka
    - Po zadatom obeležju
  - Podržano rukovanje zakasnelim podacima



Spark Structured Streaming - upiti i podržane operacije

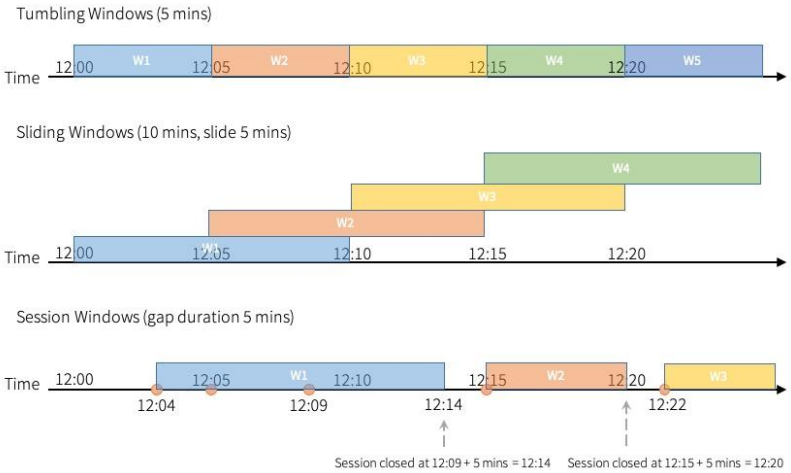
- Operacije nad **vremenskim okvirima** (*Window* operacije)
  - Omogućavaju da se prilikom kalkulacija u obzir uzmu samo podaci pristigli u definisanom vremenskom periodu
  - Za prozorske operacije bitna su dva parametra:
    - dužina prozora (*Window length*) - koliko prozor traje
    - Interval klizanja (*Sliding interval*) - interval na koji se primenjuje prozorska operacija





Spark Structured Streaming - upiti i podržane operacije

- Operacije nad **vremenskim okvirima** (*Window* operacije)
  - Podržane tri vrste vremenskih okvira: **kotrljajući** okviri, **klizni** okviri i okviri na nivou **sesije**
  - Podržane napredne opcije za rad sa zakasnelim podacima - automatsko ažuriranje i tzv. *watermarks*



Spark Structured Streaming - primeri

- Upotreba privremenih pogleda i *SQL*-a
  - `./spark/bin/spark-submit ./spark/primeri/sql_network_wordcount.py`
- Upotreba kliznih okvira
  - Povezati se na kafka brokera i pokrenuti *producer*-a u interaktivnom režimu rada
    - `docker exec -it kafka1 bash`
    - `usr/bin/kafka-console-producer --broker-list localhost:19092 --topic local-sentences --property "parse.key=true" --property "key.separator=:"`
  - Povezati se na spark-master kontejner i pokrenuti
    - `./spark/bin/spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1 ./spark/primeri/windowed_kafka_wordcount.py`
  - Slati poruke sa producer-a i posmatrati kako se odvija grupisanje po kliznim okvirima

Spark Structured Streaming - primeri

- Spajanje dva toka - *join*
  - Potrebna dva toka podataka - dva kafka topic-a
    - local-sentences
    - local-keywords
  - **`./spark/bin/spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1 ./spark/primeri/join_example.py`**

Spark Structured Streaming - Zadaci

- Preduslov: napraviti svoj *Reddit API Key*
- Napraviti novog producera, po uzoru na postojećeg, koji će skupljati komentare sa *subreddit*-a po izboru, i slati ih na *subreddit-{topic_name}* topik
- Napisati *Spark Structured Streaming* program koji će istovremeno prikupljati podatke sa oba postojeća topika i vršiti prebrajanje ulaznih reči
  - Prebrojane reči slati na novi Kafka topik
- Napisati *Spark Structured Streaming* program koji će skupljati podatke sa izlaznog kafka topika iz prethodne stavke, primeniti neku operaciju po izboru, i potom podatke smeštati na *HDFS*