Projekt Big Data: Kafka + Spark Structured Streaming

Kierunek: Big Data - inżynieria danych
Autor: Łukasz Bola

Spis treści

2. Wprowadzenie i cel projektu

Projekt prezentuje lokalny potok przetwarzania danych strumieniowych dla zdarzeń zakupowych w czasie zbliżonym do rzeczywistego. Rozwiązanie łączy Apache Kafka (warstwa przesyłania i buforowania zdarzeń), Spark Structured Streaming (warstwa przetwarzania) oraz analizę wyników i wizualizacje biznesowe.

Główny cel projektu:

- zaprojektowanie i uruchomienie kompletnego pipeline'u Big Data od generatora danych do raportu końcowego,
- przygotowanie danych i wizualizacji wspierających decyzje biznesowe.

Zakres projektu:

Zakres projektu odpowiada pełnemu przepływowi danych w zaprojektowanej architekturze:

- generowanie syntetycznych zdarzeń zakupowych JSON przez `producer.py` (oraz danych błędnych przez `producer_invalid.py`),
- publikacja i kolejkowanie zdarzeń w Apache Kafka (topic `orders`, konfiguracja dwubrokerowa),
- konsumpcja strumienia w Spark Structured Streaming, parsowanie JSON i zapis danych poprawnych do `data/orders_json`,
- równoległe wydzielanie i zapis zdarzeń niepoprawnych do osobnej ścieżki danych,
- utrwalanie stanu przetwarzania w checkpointach (`data/_checkpoints/...`) w celu wznawiania pracy strumienia,

- przetwarzanie wsadowe danych wynikowych: walidacja, czyszczenie i agregacje biznesowe,
- przygotowanie warstwy analitycznej i generowanie wizualizacji (`output/*.png`),
- niezależny monitoring strumienia przez `tracker.py` (opóźnienia end-to-end, przepustowość, commit offsetów).

3. Opis problemu i danych wejściowych

Problem badawczy:

- czy lokalna architektura oparta na Kafka + Spark Structured Streaming pozwala niezawodnie i wydajnie przetwarzać zdarzenia zakupowe, przy zachowaniu kontroli jakości danych oraz metryk opóźnień.

Dane wejściowe:

- syntetyczne zdarzenia zakupowe JSON generowane przez `producer.py`,
- dodatkowe zdarzenia celowo niepoprawne generowane przez `producer_invalid.py`.

Przykładowe pola zdarzenia:

- `order_id`,
- `event_time_ms`,
- `product`,
- `quantity`,
- `unit_price`,
- `sales_channel`,
- `payment_method`.

Wyzwania:

- obsługa danych błędnych i braków w polach,
- utrzymanie niskich opóźnień end-to-end,

4. Metody i narzędzia

Metody:

- przetwarzanie strumieniowe (Spark Structured Streaming),
- walidacja i filtracja danych wejściowych,
- agregacje biznesowe (produkt, dzień tygodnia, godzina, kanał, metoda płatności),
- analiza opisowa i wizualizacja wyników.

Narzędzia:

- Apache Kafka (2 brokerów, KRaft),
- Apache Spark / PySpark,
- Python (`confluent-kafka`, `pandas`, `seaborn`, `matplotlib`, `faker`),
- Jupyter Notebook (`kafka_spark_notebook.ipynb`),
- Docker Compose.

Artefakty projektu:

- dane wyjściowe: `data/orders_json`,
- zdarzenia niepoprawne: `data/invalid_events`,
- wykresy: `output/*.png`.

5. Wyniki i analiza

Wyniki obejmują:

- poprawne uruchomienie pełnego pipeline'u od producenta do zapisu danych i wizualizacji,
- oddzielenie zdarzeń poprawnych od niepoprawnych,
- wygenerowanie wykresów biznesowych (m.in. heatmapa przychodu, Pareto produktów, boxplot koszyka, wykres 3D).

Analiza techniczna:

- metryki producenta: `throughput_eps`, `avg_ack_ms`, `sent_ok`, `sent_error`,
- metryki konsumenta: `avg_end_to_end_latency_ms`, `processed`, `errors`,
- obserwacja zależności między tempem produkcji zdarzeń a opóźnieniami i stabilnością konsumpcji.

Analiza biznesowa:

- identyfikacja produktów o najwyższym udziale w przychodzie,
- wskazanie dni i godzin o najwyższej wartości sprzedaży,
- porównanie wartości koszyka między kanałami sprzedaży i metodami płatności.

6. Wnioski i rekomendacje

Wnioski:

- architektura Kafka + Spark jest skuteczna do lokalnej analizy strumieniowej i demonstracji procesów Big Data,
- rozdzielenie walidacji danych od warstwy analitycznej poprawia jakość wyników i interpretowalność raportu,
- wizualizacje ułatwiają przekład danych technicznych na decyzje biznesowe.

Rekomendacje:

- przenieść pipeline do środowiska klastrowego w celu testów pod większym obciążeniem,
- dodać automatyczne testy jakości danych i monitorowanie metryk (alerty),
- rozbudować projekt o model predykcyjny (np. prognoza popytu), wykorzystując przygotowane dane historyczne,
- wdrożyć dashboard online do ciągłego monitorowania KPI.

7. Repozytorium kodu

Kod źródłowy projektu: https://github.com/LukasBola/wsb-lbola-project-big-data

8. Bibliografia

1. Apache Kafka Documentation: https://kafka.apache.org/documentation/
2. Apache Spark Structured Streaming Guide: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
3. Confluent Kafka Python Client: https://docs.confluent.io/platform/current/clients/confluent-kafka-python/html/index.html
4. Pandas Documentation: https://pandas.pydata.org/docs/
5. Seaborn Documentation: https://seaborn.pydata.org/ 6. Matplotlib Documentation: https://matplotlib.org/stable/