УДК 004.415

ФЕДОРОВИЧ ІЛЛЯ АНДРІЙОВИЧ
ОЛІЙНИК ЮРІЙ ОЛЕКСАНДРОВИЧ

МОДЕЛІ ОБРОБКИ ПОТОКІВ ТЕКСТОВИХ ДАНИХ В РУШІЇ APACHE SPARK STRUCTURED STREAMING

Анотація. В даній статті розглядається рушій обробки потоків текстових даних Apache Spark Structured Streaming, описується його принцип роботи та складові. Також розглядаються моделі обробки потоків текстових даних, їх реалізації в рушії Spark Structured Streaming, порівнюються методи мікро-пакетної та потокової обробки, та описуються переваги й недоліки кожного з них.
КЛЮЧОВІ СЛОВА: ОБРОБКА ПОТОКІВ ТЕКСТОВИХ ДАНИХ, МІКРО-ПАКЕТНА ОБРОБКА, ПОТОКОВА ОБРОБКА, APACHE SPARK, APACHE SPARK STRUCTURED STREAMING.

Abstract. This paper discusses the Apache Spark Structured Streaming text data stream processing engine, describes its working principle and components. Models of text data stream processing, their implementation in the Spark Structured Streaming engine are also considered, micro-batch and stream processing methods are compared, and the advantages and disadvantages of each of them are described.
KEY WORDS: TEXT DATA STREAM PROCESSING, MICRO-BATCH PROCESSING, STREAM PROCESSING, APACHE SPARK, APACHE SPARK STRUCTURED STREAMING.

Вступ. У сучасному світі сфера обробки надвеликих масивів текстових даних в програмуванні є популярною і знаходиться у стані активного розвитку. Окремим видом обробки надвеликих масивів текстових даних є обробка потоків текстових даних, суть якої полягає в обробці не статичних, а динамічних даних, тобто таких, в яких текстові дані генеруються постійно (наприклад, новинні ресурси, соціальні мережі, месенджери). Так як зараз генерується велика кількість текстових даних, їх обробка вимагає високої швидкості обробки та масштабованості [1].

Основна частина. Apache Spark Structured Streaming – це рушій обробки надвеликих масивів даних, який забезпечує обробку структурованих даних за допомогою DataFrame API – прикладного програмного інтерфейсу, який дозволяє описувати операції над даними у вигляді SQL-подібних операцій, таких як фільтрація, перетворення та агрегація [2]. Операції над DataFrame не виконуються одразу, а будують ациклічний граф виконання, що дозволяє нам спочатку описати всі необхідні операції, які потрібно виконати над даними, а потім запустити побудований граф виконання. Рушій Spark SQL самостійно займається застосуванням операцій цього графу на нових даних, що прибувають [3]. Для забезпечення цього процесу в рушії реалізовано компонент Incremental Planner, який займається інкрементним плануванням завдань на обробку. Компонент Optimizer використовується для оптимізації графу виконань для прискорення процесу обробки даних. Рушій Spark Structured Streaming підтримує створення контрольних точок (checkpoints), тому протягом виконання процесу обробки рушій зберігає свій внутрішній стан для можливості відновлення у випадку збоїв, а також здійснює логування. Архітектура рушія Spark Structured Streaming зображена на рис. 1 [4].

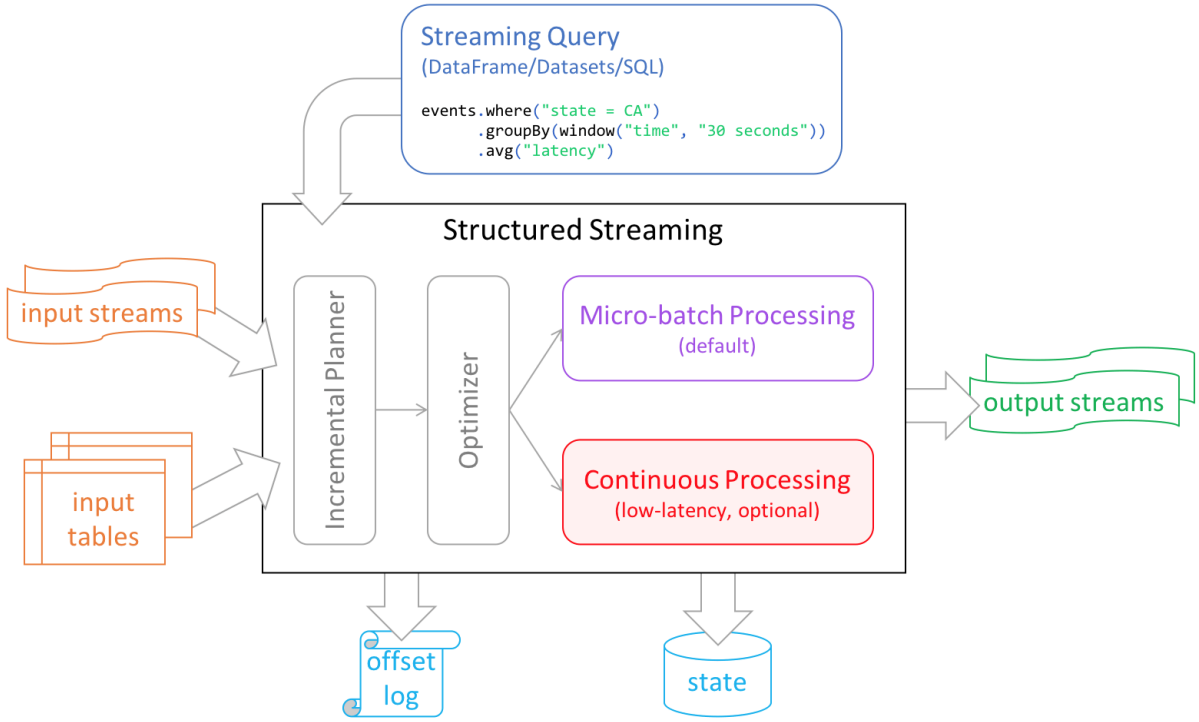

Рис. 1 – Архітектура рушія Apache Spark Structured Streaming

Мікро-пакетна модель обробки даних полягає у обробці потоку текстових даних, групуючи вхідні дані у блоки невеликого обсягу, які називаються мікро-пакетами. Критерієм завершення формування мікро-пакету може слугувати фіксований часовий відрізок (наприклад, 1 секунда) або певна подія чи тригер (наприклад, перевищення обсягу даних чи кількості записів). Враховуючи те, що обробка даних виконується лише на сформованих мікро-пакетах, а не на вхідному потоці, то така модель має властивість затримки в обробці даних, проте за рахунок одночасної обробки всіх записів у мікро-пакеті модель має кращу пропускну здатність. Мікро-пакетна модель обробки дозволяє інтерпретувати окремий мікро-пакет як статичний набір даних, що дозволяє виконувати агрегації.

Apache Spark Structured Streaming використовує оновлений підхід до обробки потокових даних, в яких потік розглядається як нескінченна таблиця, що дозволяє застосовувати ту ж логіку обробки, що і при пакетному режимі, проте рушій виконує запити інкрементно і постійно по мірі поступлення нових даних. Нескінченна таблиця з даними є уявним концептом, на практиці Spark Structured Streaming зберігає лише мінімально необхідний проміжний результат обробки даних, тобто вхідні дані, які були взяті в обробку з потоку, не зберігаються після завершення обробки. Такий підхід дозволяє гарантію обробки «рівно один раз» (exactly-once) і час затримки до 100 мс. Для моделі мікро-пакетної обробки, Apache Spark Structured Streaming також підтримує віконний режим у трьох видах: режим фіксованого, режим ковзного вікна та режим сесійного вікна. На рис. 2 [4] зображено схему мікро-пакетної моделі обробки даних. На рис. 3 [4] на часовій шкалі ілюстровано затримку обробки даних в мікро-пакетній моделі.

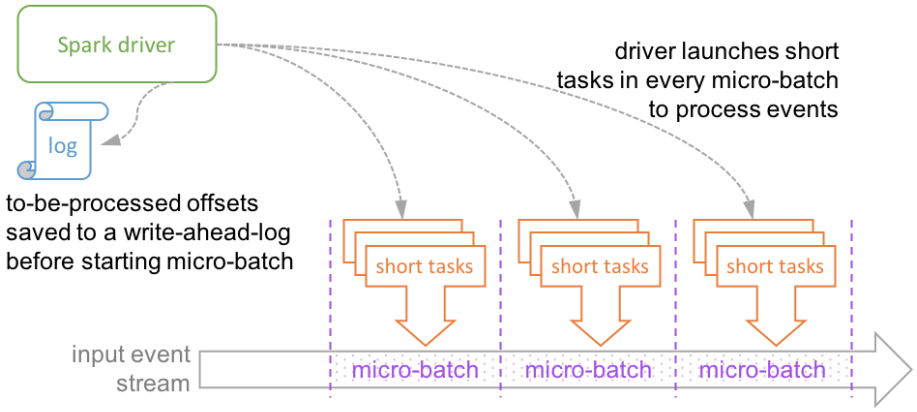

Рис. 2 – Мікро-пакетна модель обробки даних

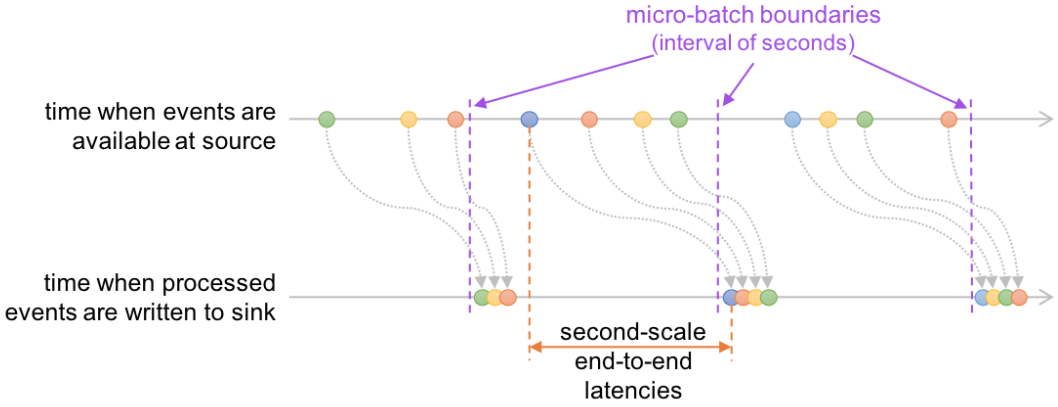

Рис. 3 – Ілюстрація затримки обробки даних в мікро-пакетній моделі

Потокова модель обробки даних полягає у обробці даних власне як потоку даних, тобто відсутнє формування будь-яких блоків, та дані, які надходять у систему, одразу ж починають оброблятись. За рахунок цього затримка обробки в такій моделі практично відсутня, проте через обробку кожного запису окремо знижується пропускна здатність.

Починаючи з версії Apache Spark 2.3, підтримується режим Continuous Processing, в якому обробка відбувається у потоковому режимі, та який дозволяє знизити час затримки до 1 мс та надає гарантію обробки «принаймні один раз» (at-least-once) [3]. Додатковою перевагою є те, що зміна режиму не потребує зміни коду для обробки даних. Spark Structured Streaming підтримує такі вхідні джерела, як файлова система (підтримуються формати CSV, JSON, Parquet, ORC), Apache Kafka, мережевий сокет (для тестування), а також два спеціальних джерела для виміру швидкодії (Rate Source та Rate Per Micro-Batch Source). Режим Continuous Processing поки що доступний в експериментальному режимі та станом на зараз підтримує як джерело даних лише Apache Kafka [3]. На рис. 4 [4] зображено схему потокової моделі обробки даних, представлену в Spark Structured Streaming режимом Continuous Processing. На рис. 5 [4] на часовій шкалі ілюстровано затримку обробки даних в потоковій моделі обробки даних. Порівняння описаних моделей представлено у табл. 1.

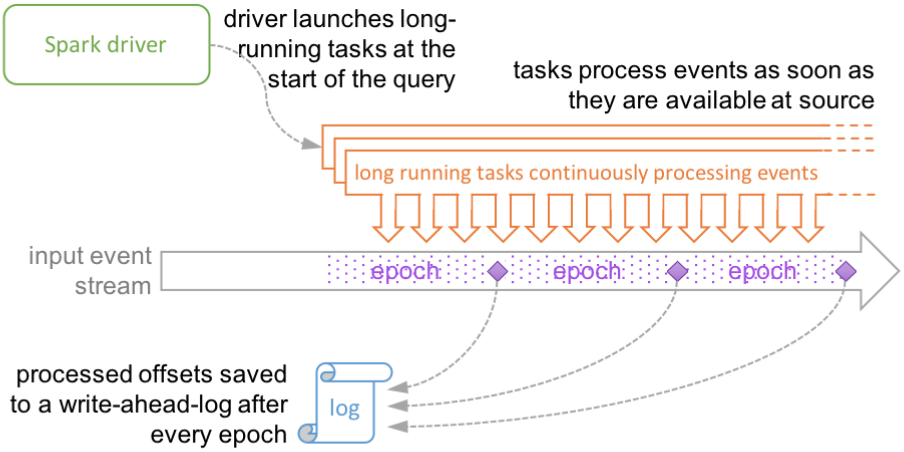

Рис. 4 – Потокова модель обробки у режимі Continuous Processing

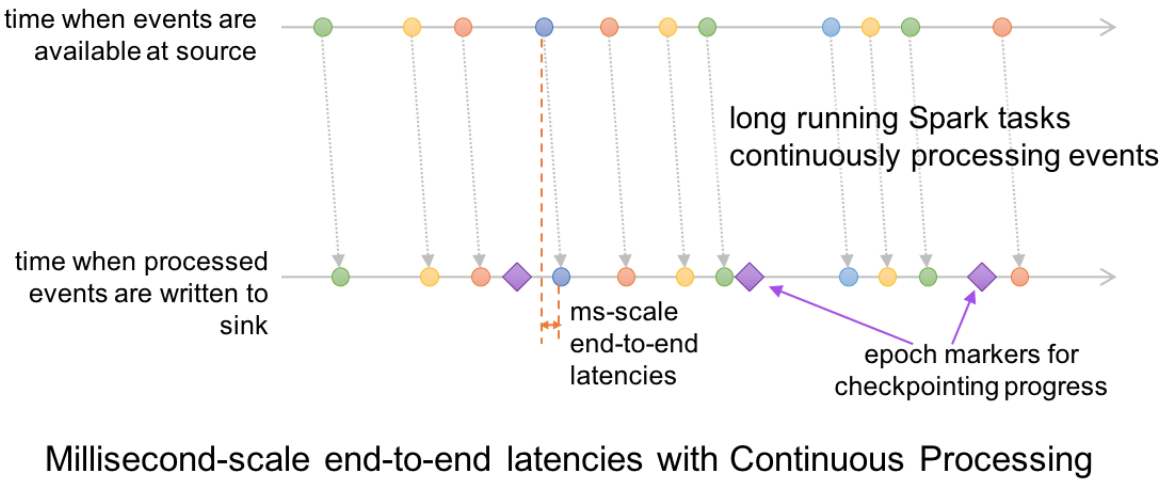

Рис. 5 – Ілюстрація затримки обробки даних в потоковій моделі, представлені режимом Continuous Processing

Табл. 1. Порівняння моделей Micro-batch Processing та Continuous Processing в рушії Apache Spark Structured Streaming

	Micro-batch Processing	**Continuous Processing**
Затримка	~ 100 мс	~ 1 мс
Пропускна здатність	Вища	Нижча
Гарантія обробки	Exactly-once	At-least-once
Вхідні джерела	Файлова система (CSV, JSON, Parquet, ORC), Apache Kafka	Apache Kafka
Вихідні джерела	Файлова система, Apache Kafka, програмний вивід (foreach та foreachBatch)	Apache Kafka
Підтримка функцій перетворення	Так	Так
Підтримка агрегацій	Частково	Ні

Висновки.

У даній роботі розглянуто мікро-пакетну та потокову модель обробки даних, реалізації цих моделей у рушії потокової обробки Spark Structured Streaming, порівняно їх між собою з описом переваг та недоліків кожної з них. Стандартна модель мікро-пакетної обробки працює з більшою затримкою та більшою пропускною здатністю та підходить до рішень, які потребують часткової агрегації оброблюваного потоку текстових даних. Модель Continuous Processing працює з меншою затримкою та меншою пропускною здатністю, та підходить рішенням, які мають велику потребу у мінімізації затримки обробки вхідних даних. Додатковим недоліком Continuous Processing є менша кількість підтримуваних джерел (лише Apache Kafka). Проте, модель Continuous Processing поки що зазначена як експериментальна, і, можливо, з часом кількість підтримуваних джерел буде зростати.

Список літератури

1. H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine and S. Khan, "A Survey of Distributed Data Stream Processing Frameworks," in IEEE Access, vol. 7, pp. 154300-154316, 2019, doi: 10.1109/ACCESS.2019.2946884.
2. Structured streaming: A declarative API for real-time applications in Apache Spark / [M. Armbrust, T. Das, J. Torres та ін.]. // Proc. Int. Conf. Manage. Data. – 2018. – С. 601–613.
3. Structured Streaming Programming Guide [Електронний ресурс] – Режим доступу до ресурсу: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html.
4. Introducing Low-latency Continuous Processing Mode in Structured Streaming in Apache Spark 2.3 [Електронний ресурс] / J.Torres, M. Armbrust, T. Das, S. Zhu – Режим доступу до ресурсу: https://www.databricks.com/blog/2018/03/20/low-latency-continuous-processing-mode-in-structured-streaming-in-apache-spark-2-3-0.html.