ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Том 110

ANNUAL OF SOFIA UNIVERSITY “ST. KLIMENT OHRIDSKI”
FACULTY OF MATHEMATICS AND INFORMATICS
Volume 110

REAL-TIME DATA INTEGRATION IN INFORMATION SYSTEMS USING STREAM PROCESSING FOR MEDICAL DATA

MARTIN KOSTOV AND KALINKA KALOYANOVA

Real-time data processing in medical information systems is becoming harder with the increase in data volume. Stream processing is a popular approach for real-time data processing, which can process large volumes of data including medical in a scalable manner. In some cases, medical data may not be available in real time because of privacy and security concerns. In this paper, we will explore the use of stream processing with static medical data using streaming platforms, Kafka, and Apache Spark. We will demonstrate how these platforms can be used to work with static data in streams and discuss the benefits and limitations of the approach. We also present a case study to illustrate the effectiveness and performance.

Keywords: stream processing, performance comparison, Apache Kafka, Apache Spark, data analysis, medical data

CCS Concepts:

• Information systems∼Data management systems∼Database management system engines∼Stream management

1. Introduction

Real-time data processing is very important for medical information systems (MIS), enabling healthcare providers to access and analyze medical data effectively and fast. Traditional batch processing approaches with relational databases are often inadequate for handling large volumes of medical data, especially in real time. Stream processing has emerged as an alternative approach for real-time data integration, which involves processing and analyzing data in real time as it is generated or ingested.

However, in some cases, medical data may not be available in real-time or may not be suitable for streaming due to privacy and security concerns. In such cases, an

DOI: 10.60063/GSU.FMI.110.101-110

alternative approach can be used – processing static data. Stream processing with static medical data involves processing and analyzing medical data that has been stored in a predefined format – a database, XML files, etc.

This paper examines the use of stream processing with static medical data using stream processing with Apache Kafka [13] and Apache Spark [2]. We discuss the challenges of batch processing in MIS, the advantages of streaming, the main components of stream processing architecture, and some of the characteristics of medical data. We also provide a case study to demonstrate the effectiveness of this approach.

2. Real-time data integration in medical information systems

Medical data is generated continuously. In order to monitor patients adequately, doctors obtain their medical records – diagnoses, medical histories, lab results, images, etc., from different systems. Even when these systems are integrated [3] processing all patient data from different MIS in real-time could be a challenge.

The batch-processing approaches with relational databases are in use most of the time. However, it involves processing data in large batches, which can cause delays in medical data analysis, making it unsuitable for real-time integration in MIS.

Stream processing emerged as a popular approach for real-time data integration in MIS. It enables doctors to monitor patient health in real-time and respond quickly. Stream processing can also provide health trends, enabling healthcare providers to identify patterns and improve patient outcomes, something that is not possible with traditional treatment without hospitalization.

3. Overview of streaming solutions

Stream processing architecture typically consists of several components, including data sources, stream processing frameworks, and output destinations for our tests [1]. We will use Kafka for both input and output. Data sources for static data can include archived medical records or data repositories. Stream processing frameworks, such as Apache Kafka and Apache Spark [4], provide the necessary tools for processing and analyzing static data. Outputs can include dashboards, alerts, and other visualizations that enable doctors to make real-time decisions based on the processed data.

The key benefit of stream processing with static medical data is its ability to handle large volumes of data efficiently with a small overhead. Medical data can be complex and can be from multiple sources, making it difficult to ensure that the data is accurate and consistent. Stream processing with static data enables doctors to monitor this data in real time. With stream processing it is possible to provide insights into patient health trends, enabling doctors to identify patterns and diagnose better.

There are also challenges associated with stream processing static medical data. One of the main challenges is ensuring the quality and consistency of the data. Despite these challenges, stream processing with static medical data can provide significant benefits, including faster response times, improved patient outcomes, and increased efficiency. In the next section, we will explore the characteristics of medical data and the challenges of processing medical data in real time, both with static data and with true streaming data.

4. Stream processing for medical data

Processing medical data in real-time can be difficult due to its specific characteristics - huge volume, different formats, lack of interoperability, etc. But stream processing framework can be utilized to overcome some of these obstacles to enable real-time processing of medical data. This approach can significantly enhance the speed and efficiency of data processing in MIS.

4.1. Medical data distinction

Working with medical data is a complex and diverse field that encompasses a wide range of data sources and data formats. Sometimes this data is organized in predetermined schemas – electronic health records (EHRs), patient summaries, etc. But a huge amount of this data – medical images, sonic representation, physician notes, patient-generated data, and social media is still not organized in predefined schemas.

Several characteristics of medical data make it challenging to process this data in real time. The generated volume is increasing at an unprecedented rate due to advances in medical technology and the adoption of EHRs. The velocity is also increasing, with real-time monitoring of patient data becoming more common [12]. Finally, medical data is characterized by variety, with data generated by a variety of sources, including medical devices, social media, and wearables.

The quality of the data is one of the main challenges to achieve the full processing potential [6]. A basic problem of medical data processing still is the need for standardization of data presentation [8].

4.2. Challenges of processing medical data in real-time

While stream processing can be an effective approach for the real-time processing of medical data, there are also challenges associated with processing static data. One of the biggest challenges is the volume of data, which can be vast and complex, particularly when dealing with large-scale datasets [11]. Processing static data also requires specialized hardware and software infrastructure to handle the volume, complexity, and heterogeneity of the data.

Ensuring data accuracy and consistency is another challenge associated with processing static data. It can be complex and heterogeneous, with data quality issues

such as missing data, incomplete data, and data inconsistency leading to inaccurate results and incorrect diagnoses. This makes it important to establish data validation processes and ensure that data is cleaned and normalized before it is processed.

Data privacy and security are critical concerns when processing medical data, particularly when dealing with static data that may be stored for long periods. Healthcare organizations must implement strict security measures to protect sensitive patient information and comply with local regulatory requirements.

Moreover, interoperability between different MIS can be a challenge when working with data [10]. Data may be stored in different formats and structures, making it difficult to exchange and integrate data between different systems [14]. To enable interoperability, a standardized data schema is needed that can accommodate the diverse sources of medical data [9].

To enable real-time processing of medical data, it is also important to have a standardized data schema that can accommodate diverse sources of medical data. A standardized data schema can also enable interoperability between different systems.

5. Problem definition

With more and more data it is becoming challenging for healthcare institutions and researchers to process and analyze all of it in an efficient manner. To address these challenges, there has been growing interest in using stream processing frameworks such as Apache Spark with distributed messaging systems like Apache Kafka to handle the medical data processing in real-time. However, there are still many open questions about how best to design and implement such systems, including issues related to data quality, scalability, fault tolerance, and security. In this paper, we explore the use of stream processing medical data processing and identify key challenges and best practices for designing and implementing such systems.

5.1. Use case dataset for the experiment

In this study, a dataset of 220,000 medical records in XML format is used to simulate a stream of data. The files present daily information about admitted and discharged patients, coming from a hospital – patient demographics, medical history, diagnosis codes, treatment information, etc. The medical records are grouped into files per day for around a period of two years and file size is between 0.5 MB and 3.5 MB depending on the number of patients for the current day.

Since we do not have real-time data, we will use this dataset to simulate a stream that is as close as possible to real-time processing. What we will do is read this dataset, transform it, and publish data to stream. We will measure the mean time it takes to do the reading from the XML file, the transformation, and the publishing to stream. We will also measure how many records we can process in one minute.

5.2. Overview of the experiment environment

The experiment described in the paper uses a setup consisting of a CPU Ryzen 5950 with 4×32 GB DDR4-3200 memory modules under the Ubuntu 22.04 operating system.

We use Apache Kafka – a distributed open-source streaming system where servers and clients communicate via a high-performance TCP network protocol.

```yaml
version: '3.4'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.3.2
    hostname: zookeeper
    environment:
      SERVICE_NAME: zookeeper
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - "2181:2181"

  kafka:
    image: confluentinc/cp-kafka:7.3.2
    hostname: kafka
    depends_on:
      - zookeeper
    environment:
      KAFKA_ZOOKEEPER_CONNECT: "zookeeper:2181"
      KAFKA_ADVERTISED_LISTENERS: "PLAINTEXT://kafka:9093,INTERNAL://localhost:9092"
      KAFKA_LISTENERS: "PLAINTEXT://:9093,INTERNAL://:9092"
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: "PLAINTEXT:PLAINTEXT,INTERNAL:PLAINTEXT"
      KAFKA_INTER_BROKER_LISTENER_NAME: "PLAINTEXT"
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_LOG_SEGMENT_MS: 300000
      KAFKA_LOG_SEGMENT_BYTES: 10485760
      KAFKA_PRODUCER_MAX_REQUEST_SIZE: 2147483648 # 2GB
      KAFKA_CONSUMER_MAX_PARTITION_FETCH_BYTES: 2147483648
      KAFKA_AUTO_CREATE_TOPICS: 1
      KAFKA_NUM_PARTITIONS: 5
      KAFKA_MESSAGE_MAX_BYTES: 1000000000
    ports:
      - "9092:9092"
      - "9101:9101"
```

Figure 1. Apache Kafka deployment configuration

```yaml
version: '3.7'

services:
  spark:
    image: apache/spark:v3.3.2
    container_name: spark
    ports:
      - "8080:8080"
      - "7077:7077"
  postgres:
    image: postgres:15.2
    container_name: postgres-db
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: medical_data
    ports:
      - "5432:5432"
```

Figure 2. Apache Spark and PostgreSQL deployment configuration

There are two main operations that clients can perform: publish – write to the stream and consume – read from the stream.

For the purpose of our test, we will use also PostgreSQL – a widespread, open-source object-relational database system, and Apache Spark – an open-source, distributed processing system used for big data workloads.

The two applications – Kafka and Postgre will be evaluated in terms of performance and efficiency when we are working with streams or based on the size of the dataset.

As a tool that defines and runs multiple docker containers, Docker Compose is used. Figure 1 and Figure 2 present the Docker Compose YAML files configuration used for the deployment of Kafka, PostgreSQL, and Spark.

The test environment includes the latest versions of all mentioned tools – Docker v4.18.0, Kafka v7.3.0, Postgres v15.2, and Spark v3.3.2.

5.3. Experiment results and discussion

Overall, the medical dataset used for the experiment contains 220762 records stored in 494 XML files [7]. Every file consists of medical records of daily admitted or discharged patients in a hospital.

We measure how much time it takes to process and publish such a record to stream and how many records we can publish per second. We also tested different batching strategies for the publish to the stream and we see that the publishing strategy is very important.

In the case of batching per file or batching all patient records per day, the results were slowest. If the batching is time-based, the results are better with more time between the batches. The best results were with 75 ms, bigger time between batches was not possible in the existing environment, because the memory of the machine was not enough.

The results of our study demonstrate that Kafka is more suitable for applications that require high throughput, such as real-time monitoring of patient records. By using stream processing with Kafka, healthcare organizations can process medical data in real time, enabling faster and more accurate diagnoses which can improve patient outcomes. Both can be used for Streaming Analytics (SA) and Complex Event Recognition (CER) [5].

Kafka's improved throughput is likely due to its efficient batch-processing capabilities, which allow it to process larger amounts of medical data at once. However, this comes at the cost of higher latency, as it takes longer to process larger batches of medical data. Moreover, if SA or CER is required the throughput will be reduced.

When we process the dataset file by file and publish it to Kafka the throughput is not optimal as seen in the first row in Table 1. On the other hand, if we use time-based batching the results are better – the second row.

Table 1. Publishing the medical records to Kafka stream measurements

Publish batching	Publish per second	Publish mean time (in ms)
Per XML file	4136	0.24
Time-based batching (75 ms)	13 872	0.072

Based on our results we present the following recommendations:

- When working with **small to medium** datasets **without streaming** medical data – Kafka or Spark are not needed as they could lead to more downsides than benefits.

- When working with **small to medium** datasets **with streaming** of medical data – Kafka can be used as an extension or relational database management system such as PostgreSQL.

- When working with **large datasets** – a system like Spark is required as it has the capabilities to process large datasets. Since Kafka does not provide data integrity reliable storage is still required.

In Table 2 we have summarized some of the main advantages and disadvantages of the different platforms we tested.

Healthcare organizations should carefully consider their specific needs and requirements when choosing a stream processing platform and evaluate the performance of different platforms using their medical data and specific use cases.

Table 2. Aggregation source advantages and disadvantages

	Spark C# with **Postgres** source	Spark C# with **Kafka** source
Processing time for aggregation	Faster	Slower due to additional I/O
Scalability	Limited by PostgreSQL capabilities	Easy to add more instances
Data integrity	Guaranteed with Postgres	Potential for data loss
Use case data size	Low to medium data volumes	High data volumes with scalability

6. Conclusions

In this study, we discuss the importance of choosing the right stream processing tools for specific cases of data usage, as well as the need for continued development and improvement in real-time data integration for medical information systems. We tested different options for fast processing medical data. We compared two different data sources – Kafka and PostgreSQL. Overall, stream processing tools such as Kafka and Spark are important for handling large amounts of data or when real-time processing with low latency is important. When real-time processing is not needed or the dataset is not very big, then those tools may not be needed, and the traditional approach is a good decision.

Future research in this area could focus on the development of new stream processing tools that are specifically designed for handling medical data in real time, as well as the integration of machine learning and artificial intelligence techniques for real-time data analysis and decision-making. Additionally, further testing and evaluation of existing stream processing tools for medical data could provide valuable insights into their performance and limitations.

Acknowledgements

This work is supported by the Fund for Scientific Research at Sofia University "St. Kliment Ohridski" under Grant 80-10-8/11.04.2023 and the project BG05M2OP 001-1.001-0004 (UNITe) funded by Operational Program Science and Education for Smart Growth co-funded by European Regional Development Fund.

References

[1] A. Akanbi and M. Masinde, A distributed stream processing middleware framework for real-time analysis of heterogeneous data on big data platform: Case of environmental monitoring, Sensors 20(11) (2020) 3166, https://doi.org/10.3390/s20113166.

[2] M. Armbrust, R. S. Xin, C. Lian et al., Spark SQL: Relational data processing in Spark, in: SIGMOD'15: Proc. 2015 ACM SIGMOD Int. Conf. on Management of Data (2015) 1383–1394, https://doi.org/10.1145/2723372.2742797.

[3] A. Bocevska, S. Savoska, I. Jolevski, N. Blazheska-Tabakovska and B. Ristevski, Implementation of innovative e-Health services and digital healthcare ecosystem – Cross4all Project summary, in: Proc. Information Systems & Grid Technologies (ISGT2022), CEUR Workshop Proceedings, 285–301, http://ceur-ws.org/Vol-3191/paper26.pdf.

[4] F. Hassan, M. E. Shaheen and R. Sahal, Real-time healthcare monitoring system using online machine learning and Spark streaming, Int. J. Adv. Comput. Sci. Appl. 11(9) (2020) 650–658, https://doi.org/10.14569/IJACSA.2020.0110977.

[5] S. Langhi, R. Tommasini and E. D. Valle, Extending Kafka streams for complex event recognition, in: 2020 IEEE Int. Conf. on Big Data (2020) 2190–2197, https://doi.org/10.1109/BigData50022.2020.9378217.

[6] K. Kaloyanova, I. Naydenova and Z. Kovacheva, Addressing data quality in healthcare, in: Proc. Information Systems and Grid Technologies (ISGT2021), CEUR Workshop Proceedings, 155–164, http://ceur-ws.org/Vol-2933/paper16.pdf.

[7] M. Kostov, StreamingMedicalData (2023) https://github.com/KostovMartin/StreamingMedicalData.

[8] P. Kovachev, E. Krastev, D. Tcharaktchiev, E. Markov and I. Evg. Ivanov, Conversion of Bulgarian observational data to OMOP common data model: Initial results, in: Proc. Information Systems & Grid Technologies (ISGT'2022), CEUR Workshop Proceedings, 113–125, http://ceur-ws.org/Vol-3191/paper10.pdf.

[9] E. Krastev, S. Abanos and D. Tcharaktchiev, Health data exchange based on archetypes of clinical concepts, in: Proc. Information Systems & Grid Technologies (ISGT2022), CEUR Workshop Proceedings, 98–112, http://ceur-ws.org/Vol-3191/paper09.pdf.

[10] M. Nisheva, H. Georgiev and P. Pavlov, Building a semantic repository for outpatient sheets, in: Proc. Information Systems and Grid Technologies (ISGT 2020), CEUR Workshop Proceedings, 14–29, http://ceur-ws.org//Vol-2656/paper3.pdf.

[11] I. Patias and V. Georgiev, The use of big data in medicine and public health policy-making: Opportunities and challenges, in: Proc. Information Systems & Grid Technologies (ISGT2020), CEUR Workshop Proceedings, 7–13, https://ceur-ws.org/Vol-2656/paper1.pdf.

[12] S. Savoska, A. Bocevska and H. Simonoski, Visualization of sensors' data in time series databases for health purposes, in: Proc. Information Systems & Grid Technologies (ISGT'2022), CEUR Workshop Proceedings, 317–334, http://ceur-ws.org/Vol-3191/paper28.pdf.

[13] M. J. Sax, G. Wang, M. Weidlich and J.-C. Freytag, Streams and tables: Two sides of the same coin, in: BIRTE'18: Proc. Int. Workshop on Real-Time Business Intelligence and Analytics (2018) 1–10, https://doi.org/10.1145/3242153.3242155.

[14] S. Velikov, K. Merdzhanov, N. Leventi and T. Kundurdzhiev, Application of openEHR platform for data exchange in ophthalmology, in: Proc. Information Systems & Grid Technologies (ISGT'2022), CEUR Workshop Proceedings, 126–134, http://ceur-ws.org/Vol-3191/paper11.pdf.

Received on March 31, 2023
Accepted on May 7, 2023

Martin Kostov

Faculty of Mathematics and Informatics
Sofia University "St. Kliment Ohridski"
5 James Bourchier Blvd.
1164 Sofia
BULGARIA

E-mail: martinkk@fmi.uni-sofia.bg

Kalinka Kaloyanova

Faculty of Mathematics and Informatics
Sofia University "St. Kliment Ohridski"
5 James Bourchier Blvd.
1164 Sofia
BULGARIA

Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia
BULGARIA

E-mail: kkaloyanova@fmi.uni-sofia.bg