BDTT

Week-6

Stream Processing

2025

1. Introduction

Imagine listening to your favourite live radio broadcast where the music, news, and stories flow continuously, and you're catching every beat as it happens; this is the essence of stream processing. Rather than waiting for a complete album or a batch of songs to download, stream processing handles each data point as soon as it arrives, like savouring every live note at a concert. Whether you're tracking the latest tweets, monitoring sensor data in a smart city, or following stock market fluctuations, stream processing makes sure you're always tuned in and ready to react. Think of it as watching a river in constant motion, where every drop of water is a new piece of information. Instead of collecting water in a bucket to analyse later, stream processing lets you observe the river's flow in real time, noticing changes and patterns immediately. This approach is crucial in today's fast-paced world, where timely insights can drive rapid decision-making. By leveraging tools like Apache Kafka, Spark, or Amazon Kinesis, businesses can build systems that transform raw, flowing data into instant, actionable intelligence; ensuring that every moment counts, and no important signal goes unnoticed.

2. Data Analytics

A data analytics pipeline is a structured process that transforms raw data into actionable insights. Here's a breakdown of the key steps.

2.1. Data Ingestion

Imagine you're hosting a large dinner party and guests are arriving with different dishes, stories, and experiences. Receiving input data is much like this welcoming phase. It involves gathering raw data from various sources; be it website logs, social media feeds, sensors, or customer transactions. This is the initial step where the "ingredients" for your analysis start to come together. In practice, data ingestion can occur in batches or as a continuous stream, much like how guests might arrive steadily throughout an evening or all at once.

Once the data arrives, just like greeting your guests warmly and ensuring they feel welcomed, the system makes sure that every bit of information is properly received and acknowledged. Tools and processes are put in place to verify that the data is complete and accessible, setting the stage for all the work that follows. This step is crucial because, just as you'd want a well-organized party, having properly ingested data ensures that your subsequent analysis won't be thrown off by missing or corrupt information [QLIK.COM].

2.2. Storing in Historical Databases

After welcoming your guests, imagine taking a moment to store their stories in a scrapbook so you can remember them later. Storing data in historical databases works similarly. Once the data is collected, it's placed into data warehouses or data lakes, which serve as the long-term memory of your organization. These storage systems allow you to keep a record of everything that's happened over time, making it possible to spot trends and changes that occur across months or years. Just as each scrapbook page might be neatly labelled and organized for future reminiscing, historical databases are carefully managed to allow easy retrieval of data when needed. This organization is essential for historical analysis; it helps ensure that as you look back to understand what led to current trends, every detail is right

where it should be. The reliability of this storage means that no matter how much data accumulates, you can always go back, learn from it, and make informed decisions for the future [EN.WIKIPEDIA.ORG].

2.3. Modelling

Now, imagine you've gathered all your guests' stories and it's time to piece them together to understand the evening better. Modelling is the process where raw data is transformed into insights by cleaning, processing, and applying algorithms. Initially, this involves tidying up the data; removing any errors or inconsistencies, much like editing a story for clarity. Once the data is neat and organized, you apply statistical methods or machine learning techniques to uncover patterns that aren't immediately obvious. Think of modelling as the creative process of turning scattered anecdotes into a coherent narrative that explains your dinner party's flow. In this stage, you might build models that predict future trends or identify key factors that made the event memorable. Just as you might validate your retelling of the party by checking with your guests, these models are rigorously tested and validated using historical data to ensure their predictions and insights are accurate and useful [IBM.COM].

2.4. Extracting Insights

Finally, consider the moment when you share the highlights and lessons from your dinner party with friends and family. Extracting insights is the stage where the results of your modelling come to life as actionable knowledge. Here, the findings from your data analysis are interpreted and presented in a way that makes sense to everyone—often through charts, dashboards, or detailed reports. This is the bridge between raw numbers and real-world decisions, turning data into a story that informs strategy and action.

This step is all about communication and clarity. Just as you'd tell engaging stories about your party to inspire or entertain your listeners, the insights from your analysis need to be presented in a clear, accessible manner. This makes it easier for business leaders or decision-makers to understand the underlying patterns, spot opportunities, and address potential issues. Ultimately, extracting insights is about transforming complex data into a compelling narrative that drives better, more informed decisions [ZUAR.COM].

3. Stream Processing

Stream processing is all about handling data the moment it arrives, rather than waiting for an entire batch to be collected and processed later. Imagine a bustling city where street cameras and sensors constantly capture live activity; from traffic flow to weather conditions; and immediately send this information for analysis. In stream processing, every new piece of data (or "event") is treated like a car passing a checkpoint on a busy highway; it's immediately processed, analysed, and the insights are instantly fed back into the system. This real-time capability means that if something unusual happens; like a sudden traffic jam or a spike in online activity; it can be detected and acted upon immediately.

Let's look at a few concrete examples. In finance, stream processing is used for fraud detection: as transactions occur, systems analyse patterns in real time to flag suspicious activity before it escalates. In the world of IoT, smart home devices continuously monitor temperature, humidity, or security sensors, enabling automated adjustments and alerts

without delay. Even in social media, platforms analyse streams of posts and comments to detect trends or breaking news as it unfolds. Tools such as Amazon Kinesis exemplify this approach; launched by AWS in 2013, it enables developers to build applications that continuously ingest and analyse live data streams, providing instant insights that help drive immediate decision-making.

By processing data as it flows into the system, stream processing minimizes latency; ensuring that businesses and services can respond to events in near real-time. Whether it's a financial institution catching a fraudulent transaction as it happens or a retailer tracking live customer behaviour to adjust inventory on the fly, this approach makes it possible to not only monitor but also act on data immediately. In contrast to batch processing; which might collect hours or even days of data before processing; the immediacy of stream processing transforms how organizations respond to rapid changes, providing a significant competitive edge in environments where every second counts [DATABRICKS.COM]. In a nutshell a stream processing pipeline contains data ingestion, data processing, and data output.

3.1. Data Sources

Think of your data sources as a lively neighbourhood market, where different stalls (IoT sensors, social media platforms, transaction systems, etc.) are all constantly providing fresh goods. In stream processing, each stall delivers new "items" (data) in real time, and the moment those items appear, they're immediately processed, no waiting around in a backroom. Below are some common data sources that fuel these continuous data pipelines, along with everyday scenarios to illustrate why real-time data can be so powerful.

3.1.1. IoT Devices

Imagine you're wearing a smartwatch that measures your heart rate, step count, and even how well you slept last night. This device sends out new readings every few seconds, and if your heart rate suddenly spikes, the system can detect it instantly. That's stream processing in action: data is analysed the moment it arrives. So whether you're a runner trying to beat your best time or someone watching out for health issues, real-time insights mean you don't have to wait until tomorrow to find out something important happened today [DATABRICKS.COM].

3.1.2. Social Media Feeds

Picture your favourite restaurant chain wanting to know the instant someone tweets about their food; good or bad. With real-time stream processing, they can see a sudden wave of praise (or complaints) as it happens, then immediately respond or adjust their strategy. Maybe they'll jump in with a witty reply or roll out a quick promotion to capitalize on the buzz. By treating each social media post like a fresh "data event," they stay plugged in to the heartbeat of their audience without missing a beat [DATABRICKS.COM].

3.1.3. Transaction Logs

Think of a ride-hailing app that logs every ride the second you tap "Pay." The moment your payment is processed, the system checks if it's valid, updates your rewards, and notifies the driver that the transaction is complete. If the app notices an unusual pattern; maybe someone trying to tip $1,000 by mistake; it flags it right away. This real-time vigilance

prevents errors or fraud from slipping through the cracks, keeping both riders and drivers happy and secure [DATABRICKS.COM].

3.1.4. Clickstream Data

Envision yourself shopping online for a new pair of running shoes. Every click you make; viewing a product, adding it to your cart, or comparing prices; generates a tiny piece of data. With stream processing, a retailer can detect patterns in your clicks instantly. If they see you're hovering over a certain shoe brand for a while, they might pop up a limited-time coupon to nudge you toward completing the purchase right then and there, rather than waiting for a daily analytics report to roll in hours later [DATABRICKS.COM].

3.1.5. Sensor Data in Manufacturing

Imagine a busy chocolate factory where machines run 24/7. Each machine has sensors that measure temperature, humidity, and pressure in real time. If any of those measurements drift out of the ideal range, a stream processing system can alert the control room immediately. Instead of discovering spoiled chocolate hours later; when the entire batch might be ruined; the system catches issues as they happen, saving both time and resources while ensuring customers get perfectly crafted treats [DATABRICKS.COM].

3.2. Stream Processing Engines

Stream processing engines operate like well-coordinated teams in a fast-paced environment, ensuring that every data point is captured and acted upon as soon as it arrives. Google Cloud Dataflow, for instance, dynamically scales its resources to handle incoming data streams effortlessly, much like a full-service catering operation that adjusts its staff based on demand. Apache Spark Streaming takes a modular approach by breaking data into micro-batches, allowing organizations to leverage familiar Spark APIs for near real-time processing without losing the benefits of batch processing techniques. Apache Flink, with its focus on processing each event individually, offers extremely low latency and precise state management, making it ideal for applications that require immediate responsiveness. Meanwhile, Apache Kafka serves as a robust backbone by reliably ingesting and distributing streams of data to various processing engines. Together, these engines provide a diverse toolkit that organizations can choose from, depending on their specific needs, existing infrastructure, and expertise.

3.2.1. Google Cloud Dataflow

A fully managed service on Google Cloud that handles both batch and streaming data through a unified model. If you're running your workloads on GCP, Dataflow automatically scales resources up or down and simplifies pipeline development using the Apache Beam SDK. It's particularly useful for teams wanting minimal infrastructure overhead and seamless integration with other Google Cloud services [GOOGLE.COM].

3.2.2. Apache Spark Streaming

Built on top of the Apache Spark ecosystem, Spark Streaming uses a micro-batch model; processing data in small, continuous intervals. This approach is straightforward if you're already familiar with Spark's APIs for batch jobs. Spark Streaming can handle high

throughput and integrates well with the broader Spark stack for tasks like machine learning and SQL-based analytics [SPARK.APACHE.ORG].

3.2.3. Apache Flink

Known for its “true” stream processing model, Flink processes data on an event-by-event basis, offering advanced features like stateful streaming and exactly once semantics. It’s particularly powerful for complex event processing, where you need to maintain and update state information in real time (e.g., tracking a user’s session data across multiple streams) [NIGHTLIES.APACHE.ORG].

3.2.4. Apache Kafka

Often referred to as a distributed event streaming platform, Kafka is widely used for ingesting and distributing data streams at scale. While Kafka itself can handle some processing via Kafka Streams, it’s typically used as a central backbone for streaming data into systems like Spark, Flink, or Dataflow. This makes Kafka an essential building block for many real-time analytics and event-driven architectures [KAFKA.APACHE.ORG].

3.3. Data Storage

When real-time data has been successfully processed, deciding where to store it becomes the next big challenge; like figuring out the best place to keep groceries after a big shopping trip. Some items need the fridge (low-latency updates in a database), others do fine in a pantry (vast amounts of raw data in a data lake), and certain goods can live in the freezer for future use (cloud storage). Each choice comes with its own trade-offs in terms of cost, speed, and accessibility, and most organizations end up blending multiple storage solutions to cover their diverse data needs.

3.3.1. Databases

Databases are like the refrigerator in a well-organized kitchen; everything is neatly labelled, easily searchable, and updated quickly when something changes. They handle structured information, such as transaction records or user accounts, ensuring reliable and consistent access. This makes them perfect for workloads that need immediate, precise queries; like checking a user’s purchase history or updating inventory counts after every sale.

3.3.2. Data Lakes

Data lakes function more like a big pantry with open shelves, where items can be dropped off without needing a specific arrangement. They’re great for large volumes of raw or semi-structured data: sensor logs, clickstream records, or social media feeds. Because data lakes don’t require a fixed schema upfront, they’re ideal for experiments and advanced analytics; teams can revisit older data, reprocess it, and discover new insights without being constrained by a rigid format.

3.3.3. Cloud Storage

Cloud storage offers a “freeze it for later” kind of approach; scalable, cost-effective, and accessible from almost anywhere. Whether you need to store occasional backups, offload infrequently accessed data, or make large files available globally, cloud storage provides near-infinite capacity. It’s a handy complement to databases and data lakes, particularly for

archiving older datasets or sharing data across multiple regions, all without worrying about running out of space on your own hardware [AWS].

3.4. Data Sinks

After streaming data has been processed and stored, the final step is making it genuinely useful; this is where data sinks come into play. A data sink is any endpoint or service where processed data is delivered so that real people (or other systems) can act on it. Think of it like the final leg of a relay race: you've handled the baton of data through ingestion, transformation, and storage, and now you pass it to its destination so that valuable insights can power dashboards, automated alerts, or new features in an application.

3.4.1. Data Visualization

Data visualization acts as the storefront window where your newly polished data is put on display for everyone to see. Tools like Power BI, Tableau, or custom web dashboards translate raw metrics into easy-to-digest graphs and charts. Instead of wading through tables of numbers, teams can instantly spot trends, outliers, and correlations. It's the difference between reading a dense technical manual and browsing a picture book; both may contain the same facts, but one is far more accessible and engaging.

3.4.2. Alerting Systems

Alerting systems function like a security guard, ready to sound the alarm the moment something goes awry. Whether it's detecting unusual spikes in e-commerce transactions or noticing a drop in sensor readings from a wind turbine, these systems push out immediate notifications; often via email, SMS, or internal messaging platforms. By focusing on key thresholds or events, alerting systems help organizations respond faster to potential issues, reducing downtime and preventing minor hiccups from spiralling into major crises.

3.4.3. Application Integration

Application integration is all about weaving processed data into everyday workflows. Consider an e-commerce website that personalizes product recommendations in real time based on a user's browsing history, or a logistics app that automatically updates shipment statuses as they happen. By embedding data directly into applications, organizations can automate decisions, streamline operations, and deliver more personalized user experiences. This approach makes data less of a static report and more of a dynamic resource that powers new features and drives innovation [SPARK.APACHE.ORG].

3.5. Monitoring and Management

Monitoring and management form the backbone of any robust stream processing pipeline. Think of it like running a busy transit system: you need real-time signals to ensure trains stay on schedule (monitoring) and a central control room to adjust routes or add more cars when crowds surge (management). Monitoring tools keep a constant watch on performance metrics such as throughput, latency, or error rates, alerting you at the first sign of trouble. Meanwhile, management tools handle tasks like scaling up resources or rolling out updates, ensuring the pipeline continues to run smoothly even as demands evolve.

3.5.1. Monitoring Tools

Are all about keeping an eye on the pipeline's real-time performance. These tools gather metrics (such as throughput, latency, or error rates) and display them on dashboards or send alerts when something goes wrong; much like a car's dashboard lights warning you if the engine is overheating. By watching these metrics closely, teams can spot bottlenecks, detect anomalies, and quickly resolve issues before they escalate.

3.5.2. Management Tools

Focus on day-to-day administration and control. They let you configure the pipeline's resources, manage version upgrades, and apply changes to different components without disrupting ongoing data flows. Think of it as the operational layer: scaling up compute when data volume spikes, scheduling updates, or rolling back to a previous version if new changes introduce errors.

4. Key Features

Stream processing isn't just about speed; it's about reacting to data as soon as it appears, like a lively conversation where everyone is constantly exchanging new information. Instead of storing data for later, stream processing treats each incoming event as an opportunity to update insights, trigger actions, or feed analytics dashboards on the spot. Below are some of the most important capabilities that make stream processing so powerful.

4.1. Real-Time Processing

Stream processing pipelines continuously ingest and transform data, so there's no waiting around for batches to accumulate. This setup is ideal for scenarios like fraud detection or customer engagement, where insights lose value if they're even a few minutes late.

4.2. Low Latency

In a streaming world, each new event is processed almost instantly; much like reading a text message the moment it's received. By reducing latency, organizations can respond faster to customer needs, operational issues, or market changes.

4.3. Data Streams

Instead of handling data in large, static sets, stream processing deals with a constant flow of records. Think of a never-ending queue of messages; each one arrives and gets processed in the blink of an eye, fuelling real-time dashboards or triggering alerts.

4.4. Stateful Processing

Sometimes, you need to remember past events to make sense of new ones; for instance, tracking a user's session over time. Stateful processing lets the pipeline hold onto important details, so it can update aggregates, compare new events to historical trends, or maintain user context.

4.5. Event-Driven

Every piece of data that arrives can trigger an immediate action, like sending a notification or updating an application's interface. This event-driven architecture reduces lag between data generation and data usage, making the entire system more responsive.

4.6. Windowing

Many real-time analyses revolve around time intervals; like counting how many transactions occur in a five-minute window. Windowing functions group events by specific periods or sessions, enabling metrics like rolling averages or sums without manually tracking each individual record.

5. Key Challenges

While stream processing opens the door to real-time insights and rapid decision-making, it also brings its share of complications. Managing continuous data flows can be like juggling while riding a unicycle: you need to keep everything balanced, especially when data arrives out of order, spikes in volume, or demands strict quality standards. Below are the main hurdles that can trip up even the most well-planned streaming pipelines.

5.1. Complexity of Handling Late Data

Not all data arrives on schedule; some events might trickle in minutes or hours late due to network issues or system delays. This creates challenges for time-based aggregations or windowing logic, as the pipeline must decide how to handle "straggler" records that show up after a window is technically closed.

5.2. State Management

Keeping track of past events can be a double-edged sword. While it enables richer analysis and context, it also introduces complexity. Memory usage and the risk of data corruption grow as you store more state, requiring careful design and robust mechanisms to ensure accuracy over time.

5.3. Resource & Performance Tuning

Continuous processing means continuous consumption of compute and memory resources. When data spikes or patterns change, the pipeline must adapt without crashing or becoming a bottleneck. This dynamic environment makes capacity planning and performance tuning an ongoing challenge.

5.4. Increased System Complexity

Real-time systems often integrate multiple components; messaging queues, databases, alerting tools, and more. Each new piece adds potential points of failure and complicates deployment, monitoring, and maintenance, making the entire ecosystem trickier to manage.

5.5. Data Quality

In a stream processing setup, there's little time to clean or re-check data. If a faulty record slips in, it can instantly affect analytics or trigger incorrect alerts. Ensuring data quality in near real-time often requires more sophisticated validation rules and fallback mechanisms.

6. Stream vs Batch

When it comes to handling data, stream processing and batch processing are like two distinct cooking styles; one whips up dishes on the fly as ingredients arrive, and the other gathers everything first and prepares meals in bulk. Choosing between these two often depends on whether you need lightning-fast reactions or you're comfortable waiting a bit for a more comprehensive analysis. Below are some everyday scenarios that illustrate these differences across key aspects.

6.1. Latency

Stream Processing: Designed for low latency, often delivering insights in milliseconds to seconds. Ideal if you need to react quickly; like detecting fraudulent credit card transactions as they happen. Consider a mobile banking app that must instantly detect unusual spending patterns. The moment a suspicious transaction occurs, the system flags it, potentially freezing the card to prevent further fraud.

Batch Processing: Generally, operates on a schedule, taking minutes or hours to produce results. This is fine for tasks like generating daily sales reports, where real-time speed isn't critical. A retail chain might gather all its sales data overnight and generate a daily sales report each morning. This delay is acceptable because the store manager just needs to see the previous day's totals and trends.

6.2. Data Nature

Stream Processing: Handles continuous, unbounded data that arrives in a steady flow, such as social media feeds or sensor outputs. There's no "end" to the dataset; the pipeline just keeps running. Think of a social media platform where tweets or posts pour in 24/7. There's no finite end to this data; it just keeps flowing. The platform's analytics system can't wait for the stream to "stop," so it processes posts in real time to track trending topics.

Batch Processing: Works best with discrete, bounded sets of data; like a collection of logs from the previous day; where you know the start and end points before processing begins. An online bookstore might collect a week's worth of order logs and store them in a file. After the week is up, they process the entire file to reconcile inventory and plan restocks.

6.2.1. Bounded Data

Bounded data refers to data that is restricted to a specific range with clear minimum and maximum limits.

Example: Consider test scores that range from 0 to 100. No score can be below 0 or above 100, so the data is bounded.

6.2.2. Unbounded Data

Unbounded data, on the other hand, isn't confined to a fixed interval. It can, in theory, extend infinitely in one or both directions.

Example: Think about annual income. While practical factors might limit it, there's no inherent theoretical maximum, so annual income is considered unbounded.

6.3. Processing Method

Stream Processing: Often uses micro-batches or event-by-event handling, letting you update dashboards and trigger alerts almost instantly. It's like cooking each dish as soon as the ingredients arrive. A ride-hailing service might update its surge pricing every few seconds based on the latest driver and passenger data. Each event (e.g., a new ride request) is processed individually or in micro-batches to adjust fares dynamically.

Batch Processing: Processes the entire dataset at once, which can be more efficient for large-scale transformations or complex queries. Think of it like preparing a big family dinner after all the groceries have been gathered. A biotech lab could store data from multiple experiments and run a big computational model over the entire dataset at the end of each month. The result is a thorough, in-depth analysis of all completed trials, albeit delivered more slowly.

6.4. Use Cases

Stream Processing: Ideal for real-time monitoring, IoT device tracking, or fraud detection; any scenario where delayed information loses value. Quick response times can avert crises or seize fleeting opportunities.

Batch Processing: Perfect for historical analysis, scheduled reports, or machine learning training tasks that don't require instant updates. Once the data is processed, it's easy to produce in-depth insights or combine it with older data for trend analysis.

7. Structured Streaming in Spark

Every second, our world produces vast amounts of data; from sensors and IoT devices to social media feeds and online transactions. This continuous flow means that if we want to respond to changes as they happen, we need a system that processes data in real time. In this tutorial, we explore Structured Streaming in Apache Spark; a model designed for handling live, ever-growing data streams.

Structured Streaming treats an incoming data stream like a table that's constantly being updated with new rows. Even though you write queries just as you would for a static, batch-processed table, Spark runs them incrementally on this unbounded input. The result is a seamless blend of batch-like ease with the immediacy of real-time processing, ensuring you always have the freshest insights at your fingertips.

Check here for further reading.

8. References

https://www.qlik.com/us/data-ingestion

https://en.wikipedia.org/wiki/Data_warehouse

https://www.ibm.com/think/topics/data-modeling

https://www.zuar.com/blog/what-are-data-insights/

https://docs.databricks.com/en/structured-streaming/index.html

https://cloud.google.com/dataflow/docs

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

https://nightlies.apache.org/flink/flink-docs-release-1.20/docs/learn-flink/overview/

https://kafka.apache.org/documentation/

https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html