(/learn)

Different Ways of Ingesting Data and the Challenges Involved

In this lesson, we will discuss the different ways in which we can ingest the data. We will also cover the challenges involved in this process.

We'll cover the following

Different ways to ingest data#

There are two primary ways to ingest data: in *real-time* and in *batches*, which run at regular intervals. Which of the two to pick depends entirely on the business requirements.

Data ingestion in real-time is typically preferred in systems reading medical data, like a heartbeat or blood pressure via wearable IoT sensors where the time is critical. It is also preferred in systems handling financial data like stock market events, etc. These are a few instances where time, lives, and money are closely linked, and we need information as soon as we can get it.

On the contrary, in systems that read trends over time, we can always ingest data in batches. For instance, consider estimating the popularity of a sport in a region over a period of time.

Let's talk about some of the challenges that developers have to face when ingesting massive amounts of data. I have added this lesson just to give you a deeper insight into the entire process. In the upcoming lesson, I also talk about the general use-cases of data streaming in the application development domain.

Challenges with data ingestion#

Slow process#

Data ingestion is a slow process. Why? We discussed this before. When the data is streamed from several different sources into the system, data coming from each and every source has a different format, different syntax, attached metadata, etc. The data as a whole is heterogeneous. It has to be transformed into a common format like *JSON* or something to be understood well by the analytics system.

The conversion of data is a tedious process. It takes a lot of computing resources and time. Flowing data has to be staged at several stages in the pipeline, processed, and then moved ahead.

Also, data has to be authenticated and verified at each and every stage to meet the organization's security standards. With the traditional data cleansing processes, it takes weeks if not months to get useful information on hand. Traditional data ingestion systems like *ETL* ain't that effective anymore.

Okay! But, you just said data can be ingested in real-time right? So, how is it slow?

I would like to bring up two things here. *First*, the modern data processing tech and frameworks are continually evolving to beat the limitations of the legacy, traditional data processing systems. Real-time data ingestion wasn't even possible with the traditional systems.

Second, analytics information obtained from real-time processing is not that accurate or holistic since the analytics continually runs on a limited set of data. This is because it streams as opposed to batch processing which takes into account the entire data set. So, basically the more time we spend studying the data the more accurate results we get.

You'll learn more about this when we go through the *Lambda* and the *Kappa* architectures of data processing.

Complex and Expensive#

The entire data flow process is resource-intensive. A lot of heavy lifting has to be done to prepare the data before being ingested into the system. Also, it isn't a side process. A dedicated team is required to pull off something like that.

Engineering teams often come across scenarios where the tools and frameworks available in the market fail to serve their needs. They have no option other than to write a custom solution from the bare bones.

Goblin is a data ingestion tool by LinkedIn. At one point in time, LinkedIn had fifteen data ingestion pipelines running which created several data management challenges. To tackle this problem, LinkedIn wrote Goblin in-house. (https://engineering.linkedin.com/data-ingestion/gobblin-big-data-ease)

It is a part of the Apache Software Foundation. This is a good read (https://engineering.linkedin.com/blog/2018/01/gobblin-enters-apache-incubation)

The semantics of the data coming from external sources changes sometimes because the data sources are not always under our control, which requires a change in the backend data processing code. Today the IoT machines in the industry are continually evolving at a rapid pace.

These are the factors we have to keep in mind when setting up a data processing and analytics system.

Moving data around is risky#

When data is moved around, it opens up the possibility of a breach. Moving data is vulnerable. It goes through several different staging areas, and the engineering teams have to put in additional effort and resources to ensure their system meets the security standards at all times.

These are some of the challenges developers face when working with streaming data.

← Back

Data Ingestion

Next →

Data Ingestion Use Cases

Mark as Completed

Report an Issue