100 XP

Introduction

4 minutes

In this module, we will look at message queues and stream processing. With the rise of internet services and the availability of continuous streams of real-time data, the challenge is to process these data streams in near real time.

Streams should be viewed as an infinite sequence of small messages that arrive continuously with no breaks. The data is not at rest, and the stream processing systems that are responsible for handling these streams should be able to continuously consume and process the data.

We will begin by looking at the main ideas behind message queue systems. We will also look at the abstractions that message queue systems provide, which are beneficial for distributed stream processing. Apache's Kafka is an example of a distributed message queue system that has been gaining a lot of popularity of late.

Next, we look at stream processing systems and understand the motivations of these systems in order to perform real-time processing of data streams. The primary issue in stream processing is that of managing state. Certain stream processing workloads are inherently **stateless**, as they can be performed on individual stream messages and can thus be scaled out quite easily. There is no coordination of data or state among the processors working in parallel.

Stateful stream processing workloads require the management of state that is updated by the messages. Stateful stream processes are more challenging to manage and scale out. This is especially true when the frequency of messages arriving in a stream increases or the number of simultaneous streams that need to be processed increases. Methods such as stream **windowing** allow for processing of multiple stream messages in batches.

Finally, we will put it all together by looking at a few big-data processing archetypes that have been discussed in the industry in the last few years. Proponents of the **Lambda architecture** propose a multi-layered approach with a batch layer to update the state of the data using a high-throughput, high-latency, and high-accuracy batch processing system. The very latest data is processed by a low-latency stream processing system that does not have high accuracy. An alternative is the **Kappa architecture**, which does away with batch processing entirely and relies on stream processing systems to produce the required results.

Learning objectives

In this module, you will:

- Define a message queue and recall a basic architecture
- Recall the characteristics, and present the advantages and disadvantages, of a message queue
- Explain the basic architecture of Apache Kafka
- Discuss the roles of topics and partitions, as well as how scalability and fault tolerance are achieved
- Discuss general requirements of stream processing systems
- Recall the evolution of stream processing
- Explain the basic components of Apache Samza
- Discuss how Apache Samza achieves stateful stream processing
- Discuss the differences between the Lambda and Kappa architectures
- Discuss the motivation for the adoption of message queues and stream processing in the LinkedIn use case

Prerequisites

- Understand what cloud computing is, including cloud service models and common cloud providers
- Know the technologies that enable cloud computing
- Understand how cloud service providers pay for and bill for the cloud
- Know what datacenters are and why they exist
- Know how datacenters are set up, powered, and provisioned
- Understand how cloud resources are provisioned and metered
- Be familiar with the concept of virtualization
- Know the different types of virtualization
- Understand CPU virtualization
- Understand memory virtualization
- Understand I/O virtualization
- Know about the different types of data and how they're stored
- Be familiar with distributed file systems and how they work
- Be familiar with NoSQL databases and object storage, and how they work
- Know what distributed programming is and why it's useful for the cloud
- Understand MapReduce and how it enables big-data computing
- Understand Spark and how it differs from MapReduce
- Understand GraphLab and how it differs from MapReduce and Spark

---

Next unit: Message queues

Continue >