Sign In START FREE TRIAL

The Path to Predictive Analytics and Machine Learning by Gary Orenstein, Ke…

Chapter 1. Building Real-Time Data Pipelines

Discussions of predictive analytics and machine learning often gloss over the details of a difficult but crucial component of success in business: implementation. The ability to use machine learning models in production is what separates revenue generation and cost savings from mere intellectual novelty. In addition to providing an overview of the theoretical foundations of machine learning, this book discusses pragmatic concerns related to building and deploying scalable, production-ready machine learning applications. There is a heavy focus on real-time uses cases including both *operational* applications, for which a machine learning model is used to automate a decision-making process, and *interactive* applications, for which machine learning informs a decision made by a human.

Given the focus of this book on implementing and deploying predictive analytics applications, it is important to establish context around the technologies and architectures that will be used in production. In addition to the theoretical advantages and limitations of particular techniques, business decision makers need an understanding of the systems in which machine learning applications will be deployed. The interactive tools used by data scientists to develop models, including domain-specific languages like R, in general do not suit low-latency production environments. Deploying models in production forces businesses to consider factors like model training latency, prediction (or “scoring”) latency, and whether particular algorithms can be made to run in distributed data processing environments.

Before discussing particular machine learning techniques, the first few chapters of this book will examine modern data processing architectures and the leading technologies available for data processing, analysis, and visualization. These topics are discussed in greater depth in a prior book (*Building Real-Time Data Pipelines: Unifying Applications and Analytics with In-Memory Architectures* [O’Reilly, 2015]); however, the overview provided in the following chapters offers sufficient background to understand the rest of

accommodate ultrafast data capture and processing. Real-time technologies share the following characteristics: 1) in-memory data storage for high-speed ingest, 2) distributed architecture for horizontal scalability, and 3) they are queryable for real-time, interactive data exploration. These characteristics are illustrated in Figure 1-1.

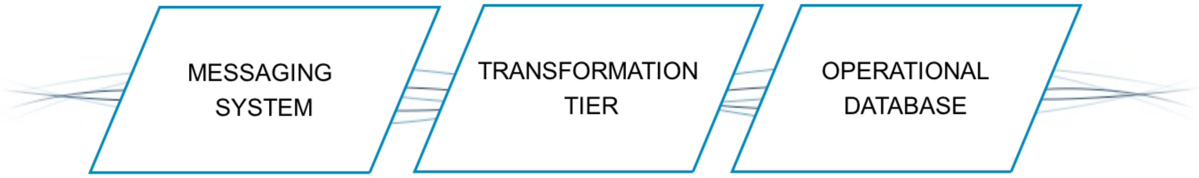

Figure 1-1. Characteristics of real-time technologies

High-Throughput Messaging Systems

Many real-time data pipelines begin with capturing data at its source and using a high-throughput messaging system to ensure that every data point is recorded in its right place. Data can come from a wide range of sources, including logging information, web events, sensor data, financial market streams, and mobile applications. From there it is written to file systems, object stores, and databases.

Apache Kafka is an example of a high-throughput, distributed messaging system and is widely used across many industries. According to the Apache Kafka website, “Kafka is a distributed, partitioned, replicated commit log service.” Kafka acts as a broker between producers (processes that publish their records to a topic) and consumers (processes that subscribe to one or more topics). Kafka can handle terabytes of messages without performance impact. This process is outlined in Figure 1-2.

Sign In START FREE TRIAL

The Path to Predictive Analytics and Machine Learning by Gary Orenstein, Ke…

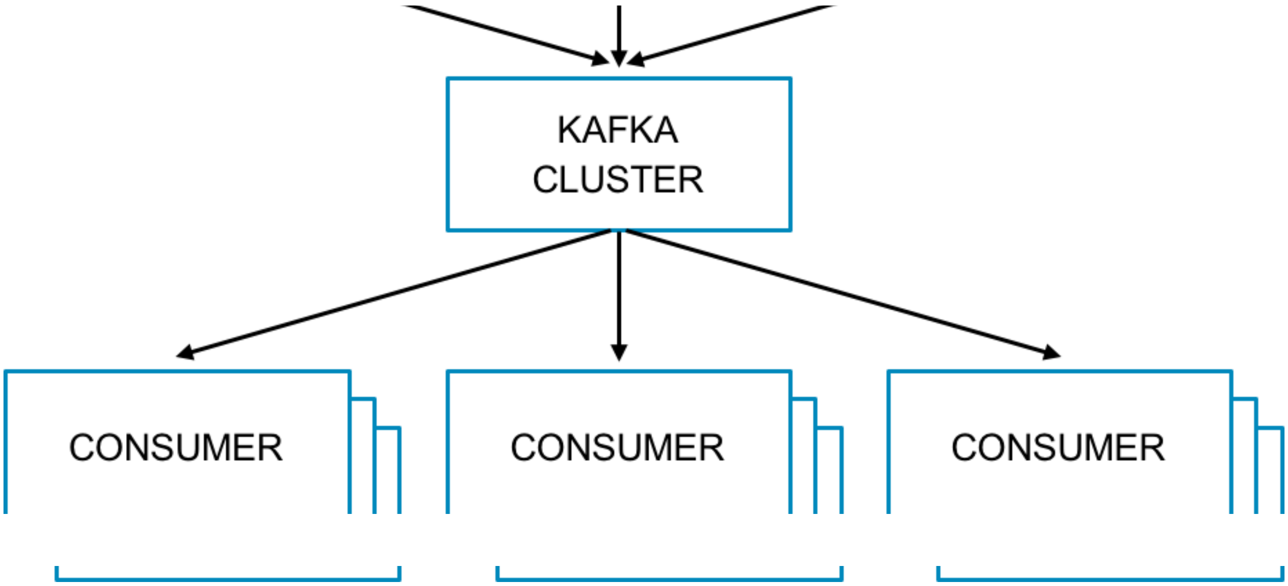

Figure 1-2. Kafka producers and consumers

Because of its distributed characteristics, Kafka is built to scale producers and consumers with ease by simply adding servers to the cluster. Kafka's effective use of memory, combined with a commit log on disk, provides ideal performance for real-time pipelines and durability in the event of server failure.

With our message queue in place, we can move to the next piece of data pipelines: the transformation tier.

Data Transformation

The data transformation tier takes raw data, processes it, and outputs the data in a format more conducive to analysis. Transformers serve a number of purposes including data enrichment, filtering, and aggregation.

Apache Spark is often used for data transformation (see Figure 1-3). Like Kafka, Spark is a distributed, memory-optimized system that is ideal for real-time use cases. Spark also includes a streaming library and a set of programming interfaces to make data processing and transformation easier.

Sign In START FREE TRIAL

Figure 1-3. Spark data processing framework

When building real-time data pipelines, Spark can be used to extract data from Kafka, filter down to a smaller dataset, run enrichment operations, augment data, and then push that refined dataset to a persistent datastore. Spark does not include a storage engine, which is where an operational database comes into play, and is our next step (see Figure 1-

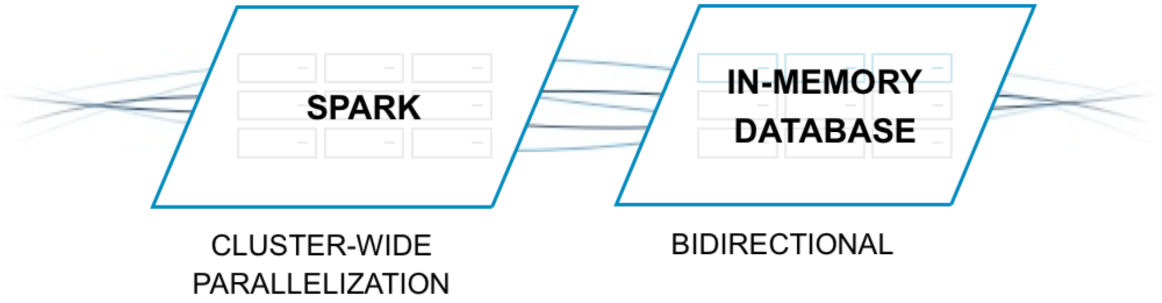

Figure 1-4. High-throughput connectivity between an in-memory database and Spark

Persistent Datastore

To analyze both real-time and historical data, it must be maintained beyond the streaming and transformations layers of our pipeline, and into a permanent datastore. Although unstructured systems like Hadoop Distributed File System (HDFS) or Amazon S3 can be used for historical data persistence, neither offer the performance required for real-time analytics.

On the other hand, a memory-optimized database can provide persistence for real-time and historical data as well as the ability to query both in a single system. By combining transactions and analytics in a memory-optimized system, data can be rapidly ingested from our transformation tier and held in a datastore. This allows applications to be built on top of an operational database that supplies the application with the most recent data information, building real-time applications at scale on legacy data processing systems is not possible. This is because traditional data architectures are siloed, using an Online Transaction Processing (OLTP)-optimized database for operational data processing and a separate Online Analytical Processing (OLAP)-optimized data warehouse for analytics.

The Enterprise Architecture Gap

In practice, OLTP and OLAP systems ingest data differently, and transferring data from one to the other requires Extract, Transform, and Load (ETL) functionality, as Figure 1-5 demonstrates.

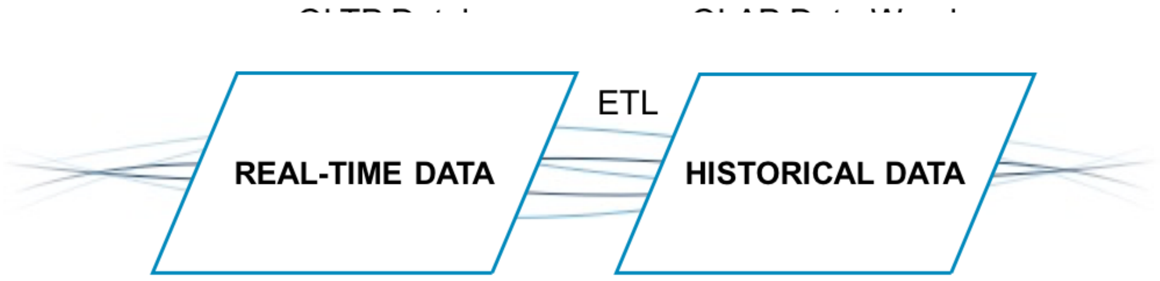

Figure 1-5. Legacy data processing model

OLAP silo

OLAP-optimized data warehouses cannot handle one-off inserts and updates. Instead, data must be organized and loaded all at once—as a large batch—which results in an offline operation that runs overnight or during off-hours. The tradeoff with this approach is that streaming data cannot be queried by the analytical database until a batch load runs. With such an architecture, standing up a real-time application or enabling analyst to query your freshest dataset cannot be achieved.

OLTP silo

On the other hand, an OLTP database typically can handle high-throughput transactions, but is not able to simultaneously run analytical queries. This is especially true for OLTP databases that use disk as a primary storage medium, because they cannot handle mixed OLTP/OLAP workloads at scale.

The fundamental flaw in a batch processing system can be illustrated through an example

Sign In START FREE TRIAL

Real-Time Pipelines and Converged Processing

Businesses implement real-time data pipelines in many ways, and each pipeline can look different depending on the type of data, workload, and processing architecture. However, all real-time pipelines follow these fundamental principles:

- Data must be processed and transformed on-the-fly so that it is immediately available for querying when it reaches a persistent datastore
- An operational datastore must be able to run analytics with low latency
- The system of record must be converged with the system of insight

One common example of a real-time pipeline configuration can be found using the technologies mentioned in the previous section—Kafka to Spark to a memory-optimized database. In this pipeline, Kafka is our message broker, and functions as a central location for Spark to read data streams. Spark acts as a transformation layer to process and enrich data into microbatches. Our memory-optimized database serves as a persistent datastore that ingests enriched data streams from Spark. Because data flows from one end of this pipeline to the other in under a second, an application or an analyst can query data upon its arrival.

With Safari, you learn the way you learn best. Get unlimited access to videos, live online training, learning paths, books, interactive tutorials, and more.

START FREE TRIAL

No credit card required

Sign In START FREE TRIAL

The Path to Predictive Analytics and Machine Learning by Gary Orenstein, Ke…

Explore

Tour

Pricing

Enterprise

Government

Education

Queue App

Learn

Blog

Contact

Careers

Press Resources

Support

Twitter

GitHub

Facebook

LinkedIn

Sign In START FREE TRIAL

The Path to Predictive Analytics and Machine Learning by Gary Orenstein, Ke…

Membership Agreement

Privacy Policy

Copyright © 2019 Safari Books Online.