Requirements for stream processing architecture

Abhishek Tiwari

Citation: *A. Tiwari*, "Requirements for stream processing architecture", Abhishek Tiwari, 2015. doi:10.59350/9b166-ek913

Published on: May 06, 2015

In 2005 Stonebraker et al. published a paper that outlined 8 key requirements for stream processing architecture. These key requirements can be easily translated into building blocks of stream processing architecture. Although, this article dates before systems such as Apache Kafka, Amazon Kinesis, Apache Spark, Apache Storm, etc. most of the requirements are still relevant.

Rule 1: Keep the data moving,

A stream processing architecture is like a data pipeline. Data moves from one stream to another, streams are branched, streams are merged but data keep moving. Storage operation are costly only performed when data is enriched and processed.

> process messages “in-stream”, without any requirement to store them to perform any operation or sequence of operations. Ideally the system should also use an active (i.e., non-polling) processing model.

Rule 2: Query using SQL on streams

A stream processing architecture should support a querying mechanism to filter out events of interest or compute real-time analytics.

> support a high-level “StreamSQL” language with built-in extensible stream oriented primitives and operators.

Rule 3: Handle stream imperfections

A stream processing architecture should handle delayed, missing and out-of-order data - i.e. straggling records. This can be typically achieved by user-defined watermarks and straggler-handlers.

> built-in mechanisms to provide resiliency against stream “imperfections”, including missing and out-of-order data, which are commonly present in real-world data streams.

Rule 4: Generate predictable outcomes

A stream processing architecture should yield the same outcome when an input stream is replayed and reprocessed. Due to the distributed nature of stream architecture, although ordering can not be guaranteed but exactly-once delivery certainly guaranteed.

> must guarantee predictable and repeatable outcomes

Rule 5: Integrate stored and streaming data

A stream processing architecture must also provide ability integrate the historical data (“batch layer”) with streaming data (“speed layer”) - i.e. something lambda architecture with .

> have the capability to efficiently store, access, and modify state information, and combine it with live streaming data. For seamless integration, the system should use a uniform language when dealing with either type of data.

Rule 6: Guarantee data safety and availability

A stream processing architecture should ensure high-availability (HA) using a fault tolerance mechanism. Typically, replication across multiple nodes enhances the durability and availability.

> ensure that the applications are up and available, and the integrity of the data maintained at all times, despite failures.

Rule 7: Partition and auto-scale

A stream processing architecture must scale horizontal by utilising shards and using partition keys to distribute load across shards. In addition, number of shards and shard capacity should automatically scale.

> have the capability to distribute processing across multiple processors and machines to achieve incremental scalability. Ideally, the distribution should be automatic and transparent.

Rule 8: Process and respond instantaneously

A stream processing architecture as a whole must support high-throughput with very low latency.

> have a highly-optimized, minimal-overhead execution engine to deliver real-time response for high-volume applications