Use Cases

Apache Flink is an excellent choice to develop and run many different types of applications due to its extensive features set. Flink's features include support for stream and batch processing, sophisticated state management, event-time processing semantics, and exactly-once consistency guarantees for state. Moreover, Flink can be deployed on various resource providers such as YARN, Apache Mesos, and Kubernetes but also as stand-alone cluster on bare-metal hardware. Configured for high availability, Flink does not have a single point of failure. Flink has been proven to scale to thousands of cores and terabytes of application state, delivers high throughput and low latency, and powers some of the world's most demanding stream processing applications.

Below, we explore the most common types of applications that are powered by Flink and give pointers to real-world examples.

- Event-driven Applications
- Data Analytics Applications
- Data Pipeline Applications

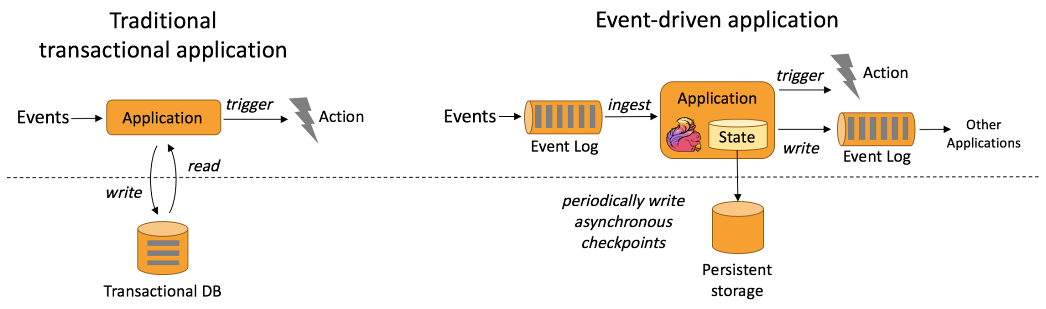
Event-driven Applications

What are event-driven applications?

An event-driven application is a stateful application that ingest events from one or more event streams and reacts to incoming events by triggering computations, state updates, or external actions.

Event-driven applications are an evolution of the traditional application design with separated compute and data storage tiers. In this architecture, applications read data from and persist data to a remote transactional database.

In contrast, event-driven applications are based on stateful stream processing applications. In this design, data and computation are co-located, which yields local (in-memory or disk) data access. Fault-tolerance is achieved by periodically writing checkpoints to a remote persistent storage. The figure below depicts the difference between the traditional application architecture and event-driven applications.



What are the advantages of event-driven applications?

Instead of querying a remote database, event-driven applications access their data locally which yields better performance, both in terms of throughput and latency. The periodic checkpoints to a remote persistent storage can be asynchronously and incrementally done. Hence, the impact of checkpointing on the regular event processing is very small. However, the event-driven application design provides more benefits than just local data access. In the tiered architecture, it is common that multiple applications share the same database. Hence, any change of the database, such as changing the data layout due to an application update or scaling the service, needs to be coordinated. Since each event-driven application is responsible for its own data, changes to the data representation or scaling the application requires less coordination.

How does Flink support event-driven applications?

The limits of event-driven applications are defined by how well a stream processor can handle time and state. Many of Flink’s outstanding features are centered around these concepts. Flink provides a rich set of state primitives that can manage very large data volumes (up to several terabytes) with exactly-once consistency guarantees. Moreover, Flink’s support for event-time, highly customizable window logic, and fine-grained control of time as provided by the

`ProcessFunction` enable the implementation of advanced business logic. Moreover, Flink features a library for Complex Event Processing (CEP) to detect patterns in data streams.

However, Flink’s outstanding feature for event-driven applications is savepoint. A savepoint is a consistent state image that can be used as a starting point for compatible applications. Given a savepoint, an application can be updated or adapt its scale, or multiple versions of an application can be started for A/B testing.

What are typical event-driven applications?

- Fraud detection (https://sf-2017.flink-forward.org/kb_sessions/streaming-models-how-ing-adds-models-at-runtime-to-catch-fraudsters/)
- Anomaly detection (https://sf-2017.flink-forward.org/kb_sessions/building-a-real-time-anomaly-detection-system-with-flink-mux/)
- Rule-based alerting (https://sf-2017.flink-forward.org/kb_sessions/dynamically-configured-stream-processing-using-flink-kafka/)
- Business process monitoring (https://jobs.zalando.com/tech/blog/complex-event-generation-for-business-process-monitoring-using-apache-flink/)
- Web application (social network) (https://berlin-2017.flink-forward.org/kb_sessions/drivetribes-kappa-architecture-with-apache-flink/)

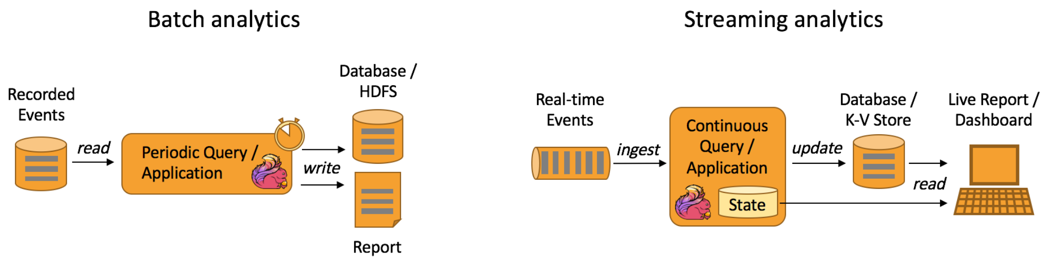
Data Analytics Applications

What are data analytics applications?

Analytical jobs extract information and insight from raw data. Traditionally, analytics are performed as batch queries or applications on bounded data sets of recorded events. In order to incorporate the latest data into the result of the analysis, it has to be added to the analyzed data set and the query or application is rerun. The results are written to a storage system or emitted as reports.

With a sophisticated stream processing engine, analytics can also be performed in a real-time fashion. Instead of reading finite data sets, streaming queries or applications ingest real-time event streams and continuously produce and update results as events are consumed. The results are either written to an external database or maintained as internal state. Dashboard application can read the latest results from the external database or directly query the internal state of the application.

Apache Flink supports streaming as well as batch analytical applications as shown in the figure below.



What are the advantages of streaming analytics applications?

The advantages of continuous streaming analytics compared to batch analytics are not limited to a much lower latency from events to insight due to elimination of periodic import and query execution. In contrast to batch queries, streaming queries do not have to deal with artificial boundaries in the input data which are caused by periodic imports and the bounded nature of the input.

Another aspect is a simpler application architecture. A batch analytics pipeline consist of several independent components to periodically schedule data ingestion and query execution. Reliably operating such a pipeline is non-trivial because failures of one component affect the following steps of the pipeline. In contrast, a streaming analytics application which runs on a sophisticated stream processor like Flink incorporates all steps from data ingestions to continuous result computation. Therefore, it can rely on the engine’s failure recovery mechanism.

How does Flink support data analytics applications?

Flink provides very good support for continuous streaming as well as batch analytics. Specifically, it features an ANSI-compliant SQL interface with unified semantics for batch and streaming queries. SQL queries compute the same result regardless whether they are run on a static data set of recorded events or on a real-time event stream. Rich support for user-defined functions ensures that custom code can be executed in SQL queries. If even more custom logic is required, Flink's DataStream API or DataSet API provide more low-level control. Moreover, Flink's Gelly library provides algorithms and building blocks for large-scale and high-performance graph analytics on batch data sets.

What are typical data analytics applications?

- Quality monitoring of Telco networks (http://2016.flink-forward.org/kb_sessions/a-brief-history-of-time-with-apache-flink-real-time-monitoring-and-analysis-with-flink-kafka-hb/)
- Analysis of product updates & experiment evaluation (https://techblog.king.com/rbea-scalable-real-time-analytics-king/) in mobile applications
- Ad-hoc analysis of live data (https://eng.uber.com/athenax/) in consumer technology
- Large-scale graph analysis

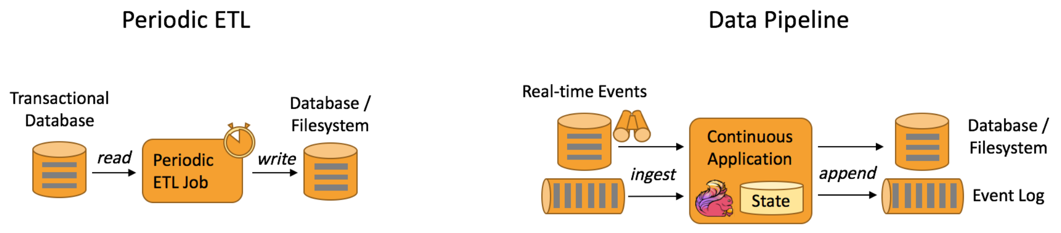
Data Pipeline Applications

What are data pipelines?

Extract-transform-load (ETL) is a common approach to convert and move data between storage systems. Often ETL jobs are periodically triggered to copy data from from transactional database systems to an analytical database or a data warehouse.

Data pipelines serve a similar purpose as ETL jobs. They transform and enrich data and can move it from one storage system to another. However, they operate in a continuous streaming mode instead of being periodically triggered. Hence, they are able to read records from sources that continuously produce data and move it with low latency to their destination. For example a data pipeline might monitor a file system directory for new files and write their data into an event log. Another application might materialize an event stream to a database or incrementally build and refine a search index.

The figure below depicts the difference between periodic ETL jobs and continuous data pipelines.



What are the advantages of data pipelines?

The obvious advantage of continuous data pipelines over periodic ETL jobs is the reduced latency of moving data to its destination. Moreover, data pipelines are more versatile and can be employed for more use cases because they are able to continuously consume and emit data.

How does Flink support data pipelines?

Many common data transformation or enrichment tasks can be addressed by Flink's SQL interface (or Table API) and its support for user-defined functions. Data pipelines with more advanced requirements can be realized by using the DataStream API which is more generic. Flink provides a rich set of connectors to various storage systems such as Kafka, Kinesis, Elasticsearch, and JDBC database systems. It also features continuous sources for file systems that monitor directories and sinks that write files in a time-bucketed fashion.

What are typical data pipeline applications?

- Real-time search index building (https://ververica.com/blog/blink-flink-alibaba-search) in e-commerce
- Continuous ETL (https://jobs.zalando.com/tech/blog/apache-showdown-flink-vs.-spark/) in e-commerce

Copyright © 2014-2019 The Apache Software Foundation (http://apache.org). All Rights Reserved.

Apache Flink, Flink®, Apache®, the squirrel logo, and the Apache feather logo are either registered trademarks or trademarks of The Apache Software Foundation.

Privacy Policy (/privacy-policy.html) · RSS feed (/blog/feed.xml)