# Giới thiệu

tài liệu tìm hiểu về Apache Flink

Nội dung

Spark vs Flink

| Spark | Flink |
| --- | --- |
| Spark is not a true real time processing. It is Near to real time processing framework. | Flink is True processing framework. |
| At heart Spark is a Batch processing framework. | At heart Flink is a Stream processing framework. |
| Spark's streaming computation model is based on Microbatching. | Flink's Streaming model is based on Windowing and Checkpointing. (theo phút) |
| Spark is implemented in Scala. | Flink is implemented in Java. |
| Spark does not have an efficient memory manager. Frequently gets Out of Memory. (do user tự phải quản lí bộ nhớ) | Flink has it's own efficient automatic memory manager. Rarely gets Out of Memory. (user gần như không phải quản lí bộ nhớ) |
| Use DAG execution engine | Use Controlled Cyclic Dependency Graph engine |

Tổng quan

flink xử lí data dạng stream đúng nghĩa, not dạng batch (cục)

flink chạy trên: yarn, mesos, kubernete

data source: kafka, jdbc, rabit mq, elasticsearch, cassandra

# Apis

<https://ci.apache.org/projects/flink/flink-docs-release-1.13/fig/levels_of_abstraction.svg>

gồm các tầng sql > table > datastream, dataset > stateful stream processing

- The lowest level abstraction simply offers stateful and timely stream processing. It is embedded into the DataStream API via the Process Function. It allows users to freely process events from one or more streams, and provides consistent, fault tolerant state. In addition, users can register event time and processing time callbacks, allowing programs to realize sophisticated computations.

- In practice, many applications do not need the low-level abstractions described above, and can instead program against the Core APIs: the DataStream API (bounded/unbounded streams) and the DataSet API (bounded data sets). These fluent APIs offer the common building blocks for data processing, like various forms of user-specified transformations, joins, aggregations, windows, state, etc. Data types processed in these APIs are represented as classes in the respective programming languages.

- The low level Process Function integrates with the DataStream API, making it possible to use the lower-level abstraction on an as-needed basis. The DataSet API offers additional primitives on bounded data sets, like loops/iterations.

- The Table API is a declarative DSL centered around tables, which may be dynamically changing tables (when representing streams). The Table API follows the (extended) relational model: Tables have a schema attached (similar to tables in relational databases) and the API offers comparable operations, such as select, project, join, group-by, aggregate, etc. Table API programs declaratively define what logical operation should be done rather than specifying exactly how the code for the operation looks. Though the Table API is extensible by various types of user-defined functions, it is less expressive than the Core APIs, and more concise to use (less code to write). In addition, Table API programs also go through an optimizer that applies optimization rules before execution.

- One can seamlessly convert between tables and DataStream/DataSet, allowing programs to mix the Table API with the DataStream and DataSet APIs.

- The highest level abstraction offered by Flink is SQL. This abstraction is similar to the Table API both in semantics and expressiveness, but represents programs as SQL query expressions. The SQL abstraction closely interacts with the Table API, and SQL queries can be executed over tables defined in the Table API.

# Stateful Stream Processing

<https://ci.apache.org/projects/flink/flink-docs-release-1.13/docs/concepts/stateful-stream-processing/>

- dạng stream có lưu trạng thái

VD:

- muốn tìm trong stream có pattern nào đó DDOS, …

- train model AI thì state là trọng số của model

## Keyed State

- dạng key-value, stream sẽ được phân luồng, partition bởi key tương ứng

- đảm bảo cùng key ở sẽ vào cùng một luồng, cùng một partition, đảm bảo consistency

<https://ci.apache.org/projects/flink/flink-docs-release-1.13/fig/state_partitioning.svg>

-