Flink

An overview

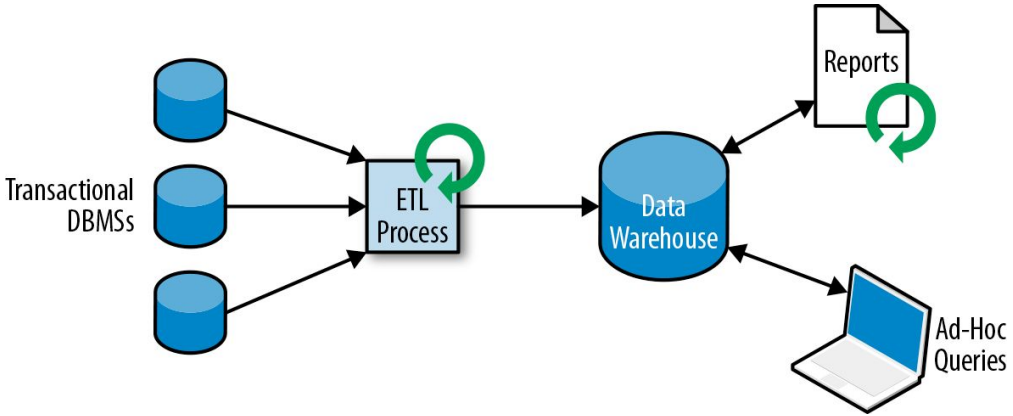
Agenda

- What the (**Flink**)?
- **Evolution** of Stream Processing
- **HLA** & Stream processing **Concepts**
- Common **APIs** (DataStream, DataSet, Table API)
- **Run** your own Flink application.
- Best Practices
- **Resources** & Exploring further.

FLINK

DISTRIBUTED DATA STREAM PROCESSING FRAMEWORK

Evolution



Traditional DW Architecture

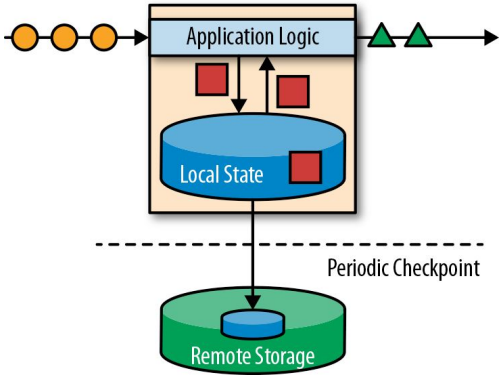
Evolution

All DATA that is created or obtained from a source is essentially a

stream of

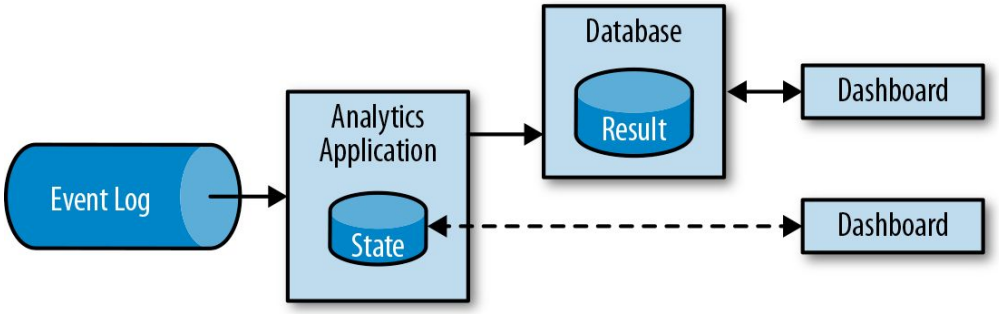
EVENTS

In order to process a data stream ..



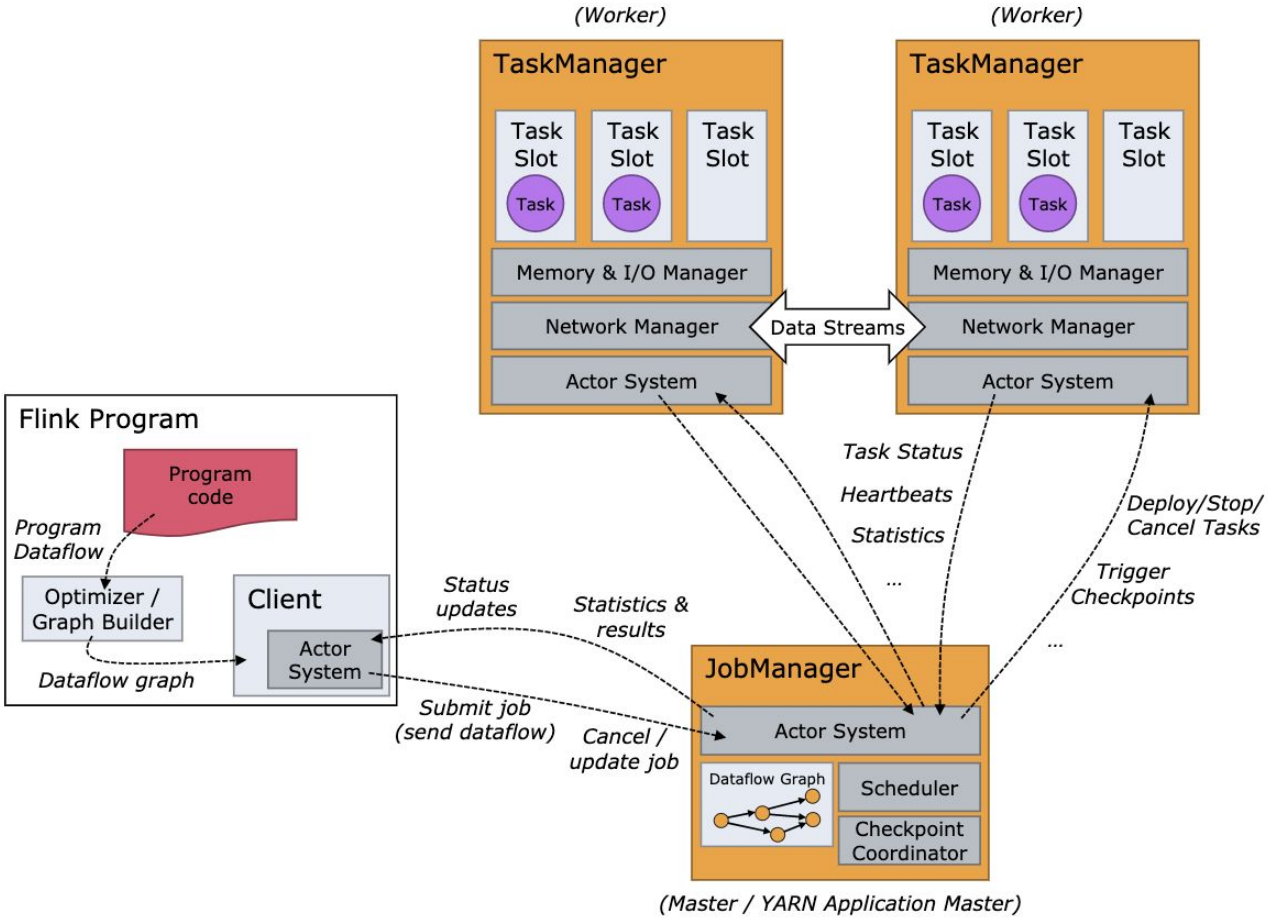
Stateful Streaming Application

Evolution

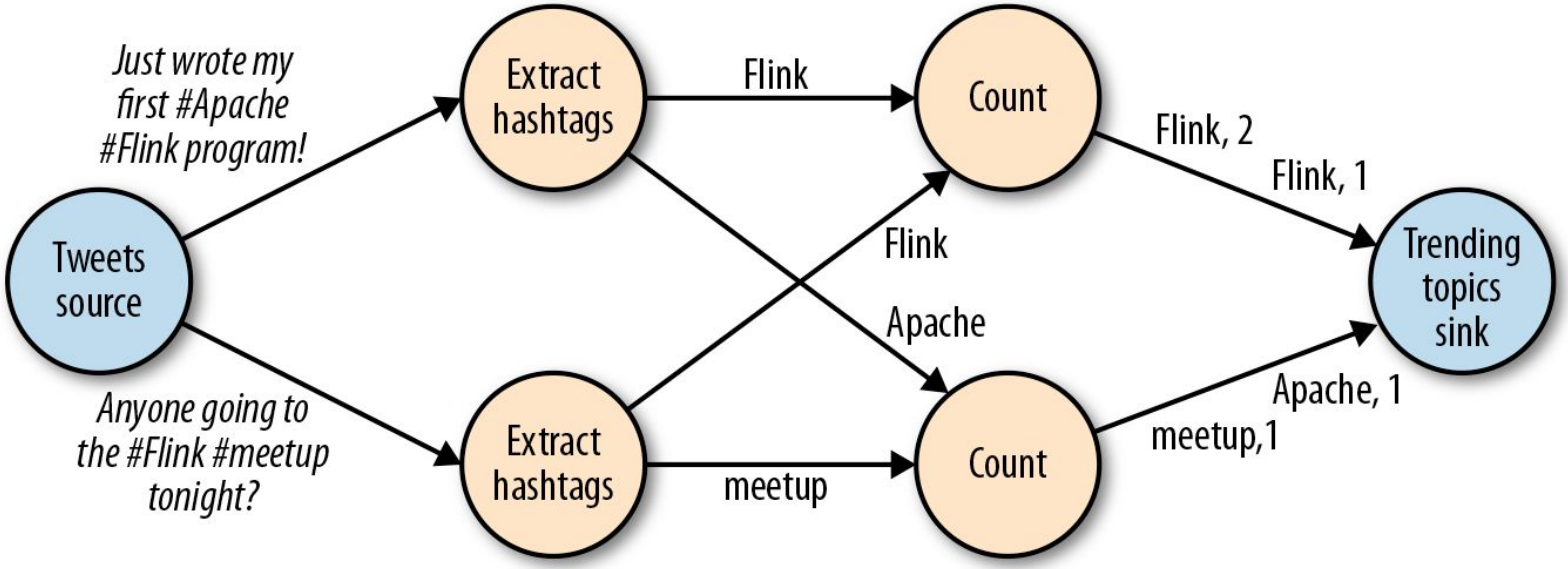


Analytics Streaming Architecture

HLA

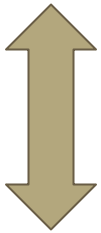

(Worker)
(Worker)
TaskManager
TaskManager
Task Slot
Task
Memory & I/O Manager
Network Manager
Actor System
Data Streams
Task Status
Heartbeats
Statistics
...
Deploy/Stop/ Cancel Tasks
Trigger Checkpoints
...
Flink Program
Program code
Program Dataflow
Optimizer / Graph Builder
Client
Actor System
Dataflow graph
Status updates
Statistics & results
Submit job (send dataflow)
Cancel / update job
JobManager
Actor System
Dataflow Graph
Scheduler
Checkpoint Coordinator
(Master / YARN Application Master)

Concepts : DATAFLOW



Concepts : PERFORMANCE

Latency Time taken for an event to be processed.



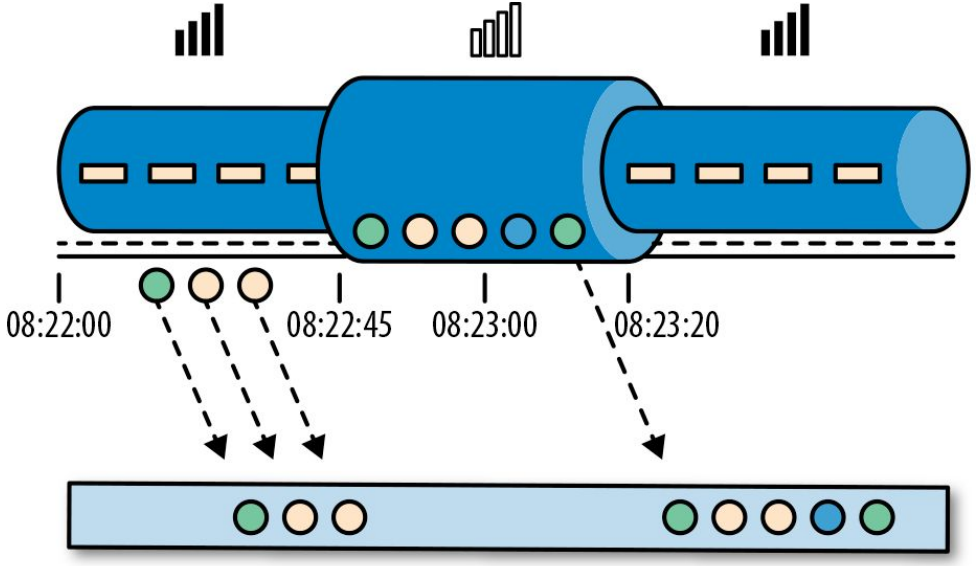
Throughput A measure of the system’s processing capacity.

Concepts: TIME SEMANTICS

Processing Time

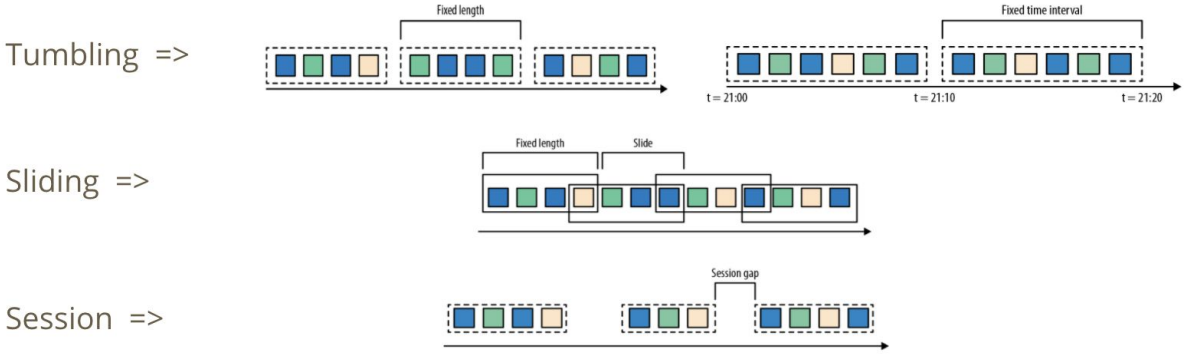
vs

Event Time

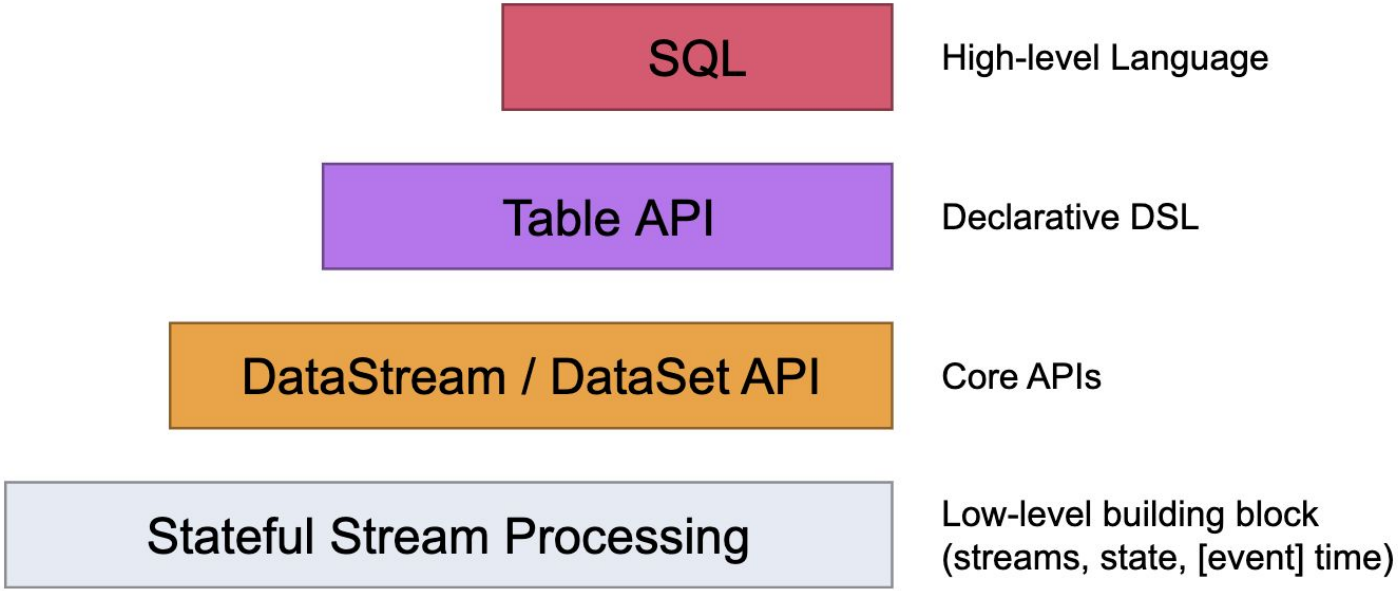


Concepts: OPERATIONS

- **Stateless vs Stateful**
- **Transformation Operations & Rolling Aggregations**
- **Window operations (Holistic aggregate, Most recent data ..)**



Flink APIs


SQL
High-level Language
Table API
Declarative DSL
DataStream / DataSet API
Core APIs
Stateful Stream Processing
Low-level building block
(streams, state, [event] time)

DataStream

```
val inputStream: DataStream[(Int, Int, Int)] = env.fromElements(
  (1, 2, 2), (2, 3, 1), (2, 2, 4), (1, 5, 3))

val resultStream: DataStream[(Int, Int, Int)] = inputStream
  .keyBy(0) // key on first field of the tuple
    .sum(1)   // sum the second field of the tuple in place
```

DataSet

```
    val env = ExecutionEnvironment.getExecutionEnvironment

    val text: DataSet[String] = env.fromElements("Who's there?", "I think I hear them. Stand, ho!")

    val counts = text.flatMap { _.toLowerCase.split("\\W+") filter { _.nonEmpty } }

      .map { (_, 1) }

      .groupBy(0)

      .sum(1)

    counts.print()
```

Table API

```
val tableEnv = ... // create a Table Environment for specific planner batch or streaming

tableEnv.executeSql("CREATE TEMPORARY TABLE table1 ... WITH ( 'connector' = ... )" ) //
create an input Table

tableEnv.executeSql("CREATE TEMPORARY TABLE outputTable ... WITH ( 'connector' = ... )" ) //
register an output Table


val table2 = tableEnv.from("table1").select(...) // create a Table from a Table API query

val table3 = tableEnv.sqlQuery("SELECT ... FROM table1 ..." ) // create a Table from a SQL
query


// emit a Table API result Table to a Table Sink, same for SQL result

val tableResult = table2.executeInsert("outputTable")
```

FLINK RUN

Best Practices

- Build a solid CICD pipeline on your infra.
- Leverage event streaming platforms such as Kafka, Kinesis
- Periodic Checkpoints & Savepoints.
- Choose wisely between Event Time and Process Time.
- Develop your own configurable app framework.
- Provision for batch using the same pipeline.
- Setup Monitoring and alert.

References

- https://ci.apache.org/projects/flink/flink-docs-release-1.12/
- https://www.flink-forward.org/
- **Stream Processing with Apache Flink** - by Fabian Hueske; Vasiliki Kalavri
- https://eng.uber.com/kappa-architecture-data-stream-processing/
- Flink Courses (LinkedIn Learning)
- https://github.com/nupsea/flink-wlm

?’s

Thank you

- eanups@yahoo.com