Aalto University
School of Science

Apache Flink for Stream Processing in Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

Stateful computations

- **Many computations on streaming data need to “memorize” data and use such data for computations**
  - calculate the number of records over a period of time
  - detect anomalies in data

⇒ stateful computations

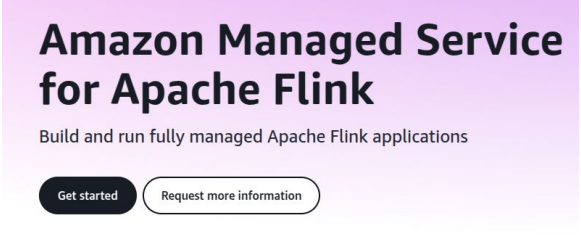
- **Requires to manage and store data for real-time processing**

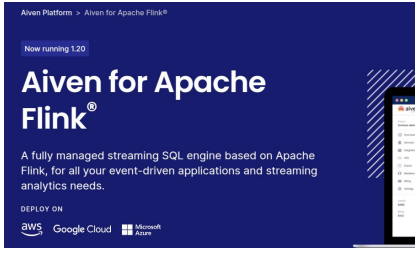
- **Apache Flink supports stateful computations with streaming data at very large scale, high performance, low latency**

Some impression

Four Billion Records per Second! What is Behind Alibaba Double 11 - Flink Stream-Batch Unification Practice during Double 11 for the Very First Time

Alibaba Clouder December 2, 2020 7,274 0





Apache Flink

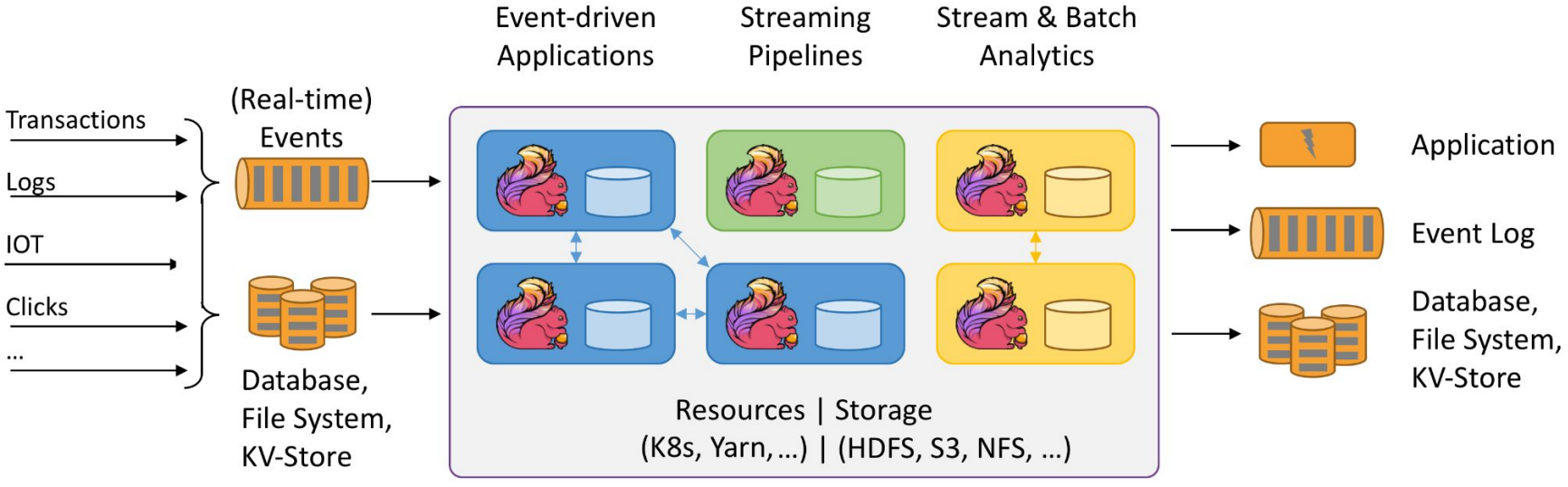


Figure source: https://flink.apache.org/

Flink runtime view

- Parallelism
- Checkpointing
- Monitoring

Remember: stream applications running 24/7

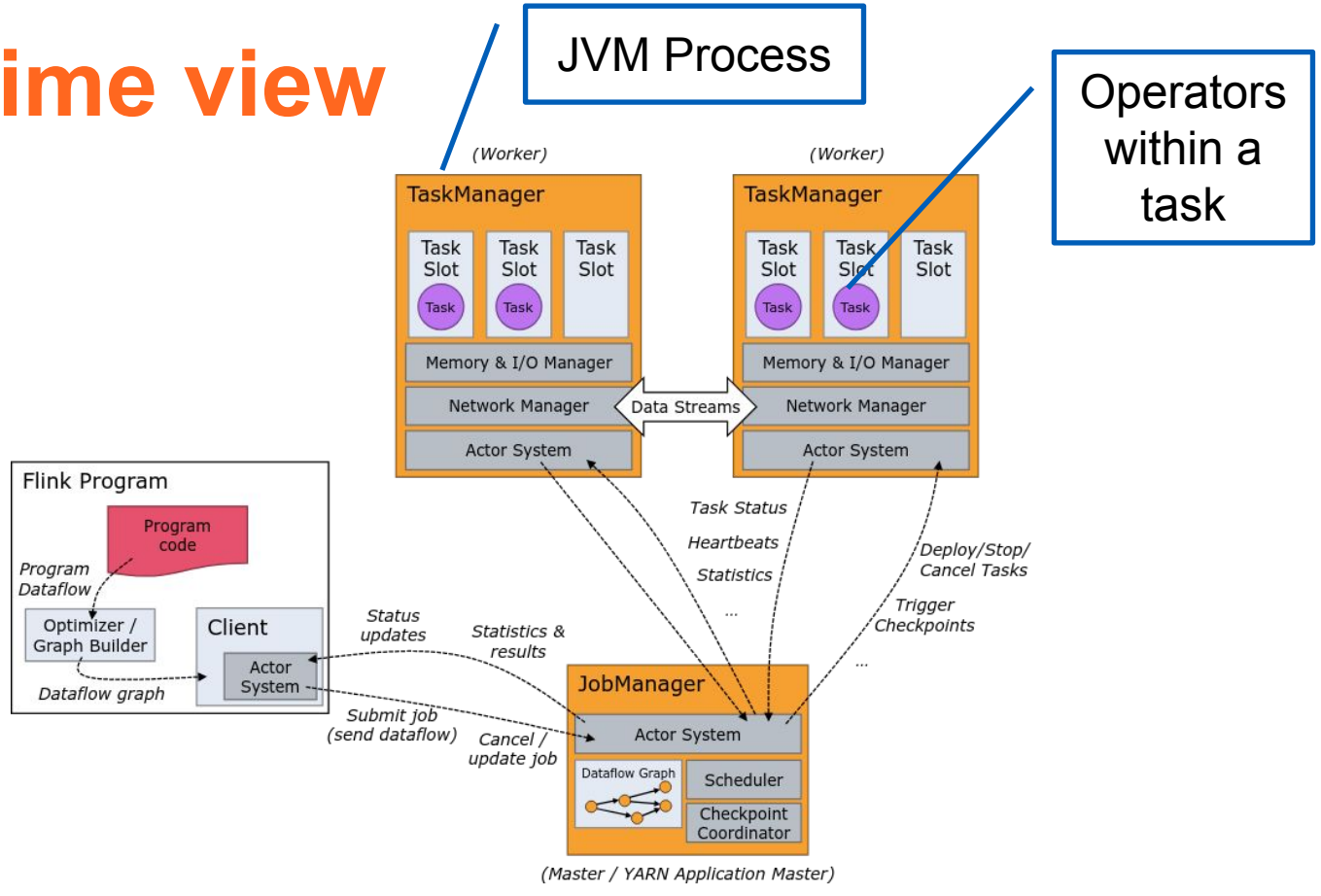
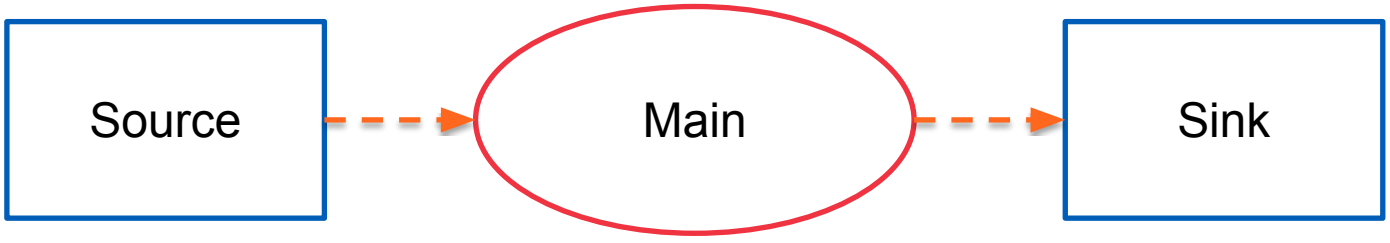


Figure source: https://nightlies.apache.org/flink/flink-docs-release-1.20/docs/concepts/flink-architecture/

Main elements in Flink applications



- **Rich set of sources and sinks via many connectors**

Connectors

- **Major systems in big data ecosystems**
- **We have used many of them in our study**
  - Apache Kafka
  - Apache Cassandra
  - Elasticsearch (sink)
  - Hadoop FileSystem
  - RabbitMQ
  - Apache NiFi
  - Google PubSub

Main programming primitives

- **Setting environments**
- **Handling inputs and outputs via data streams**
- **Key functions for processing data**
- **Stream processing flows**

Bounded and unbounded streams

Stream processing flows

Split streaming data into different windows using a key for analytics purposes

Keyed data/Keyed window: if we can separate data via keys

```
stream
       .keyBy(...)               <-  keyed versus non-keyed windows
       .window(...)              <-  required: "assigner"
      [.trigger(...)]            <-  optional: "trigger" (else default trigger)
      [.evictor(...)]            <-  optional: "evictor" (else no evictor)
      [.allowedLateness(...)]    <-  optional: "lateness" (else zero)
      [.sideOutputLateData(...)] <-  optional: "output tag" (else no side output for late data)
       .reduce/aggregate/apply()      <-  required: "function"
      [.getSideOutput(...)]      <-  optional: "output tag"
```

Source: https://nightlies.apache.org/flink/flink-docs-master/docs/dev/datastream/operators/windows/

Stream processing flows

Handling streaming data without a key for analytics purposes

```
stream
       .windowAll(...)           <-  required: "assigner"
      [.trigger(...)]            <-  optional: "trigger" (else default trigger)
      [.evictor(...)]            <-  optional: "evictor" (else no evictor)
      [.allowedLateness(...)]    <-  optional: "lateness" (else zero)
      [.sideOutputLateData(...)] <-  optional: "output tag" (else no side output for late data)
       .reduce/aggregate/apply()      <-  required: "function"
      [.getSideOutput(...)]      <-  optional: "output tag"
```

Source: https://nightlies.apache.org/flink/flink-docs-master/docs/dev/datastream/operators/windows/

Dataflow vs Execution Graph

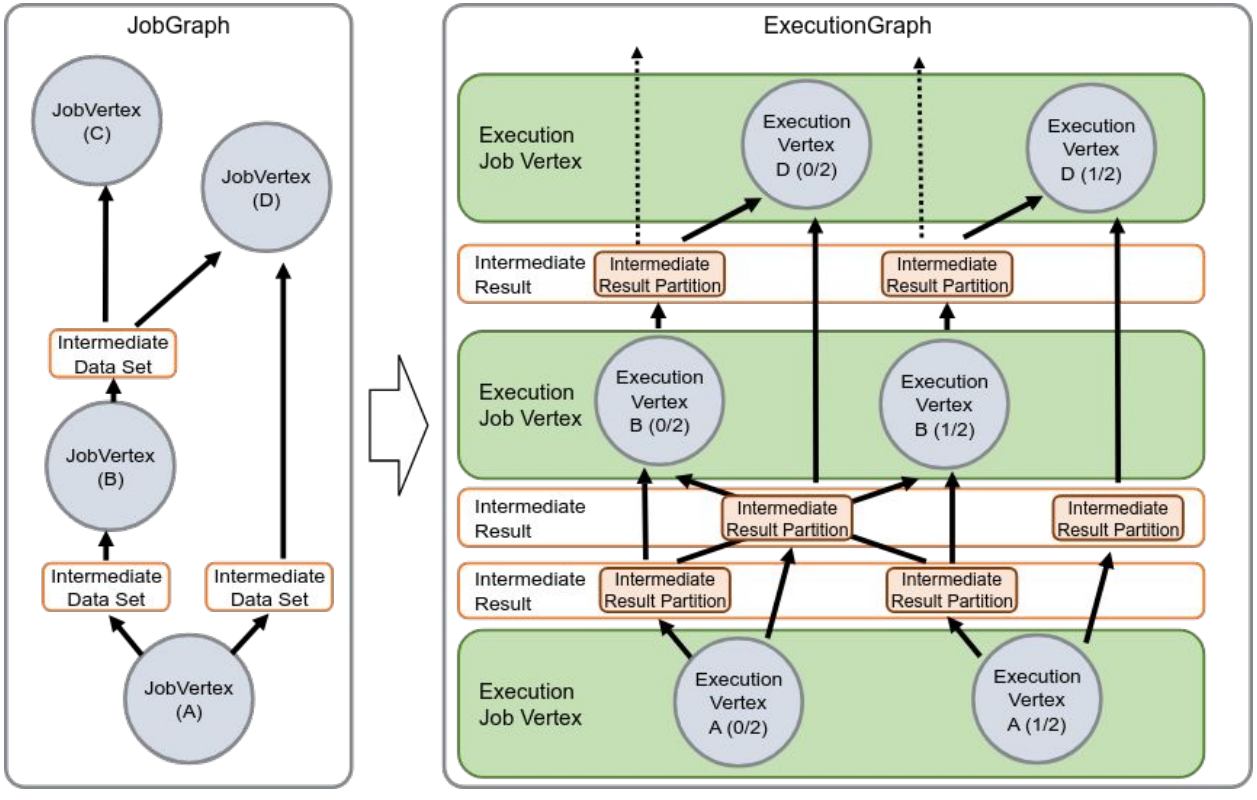
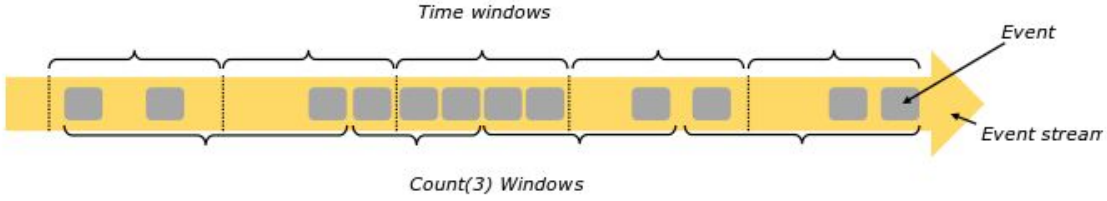


Figure source: https://nightlies.apache.org/flink/flink-docs-master/docs/internals/job_scheduling/

Windows and Times

Windows



Times

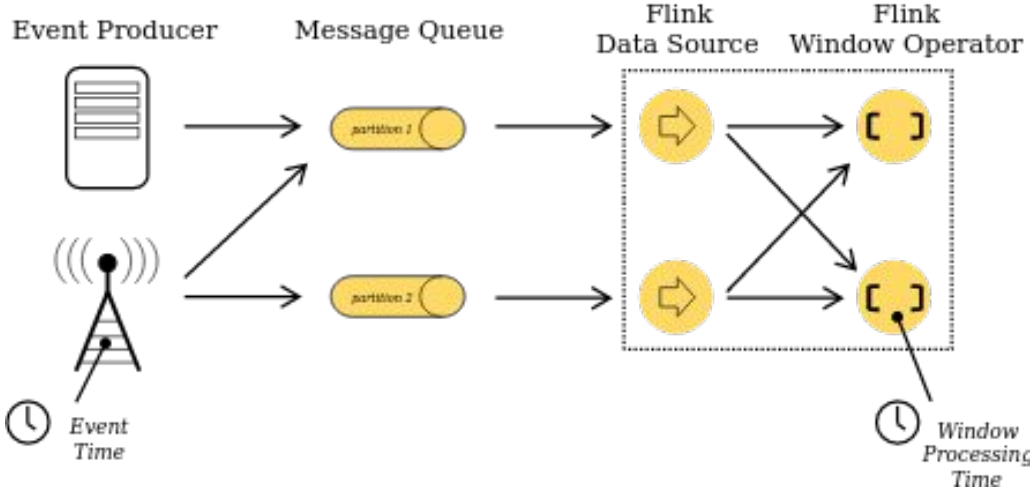
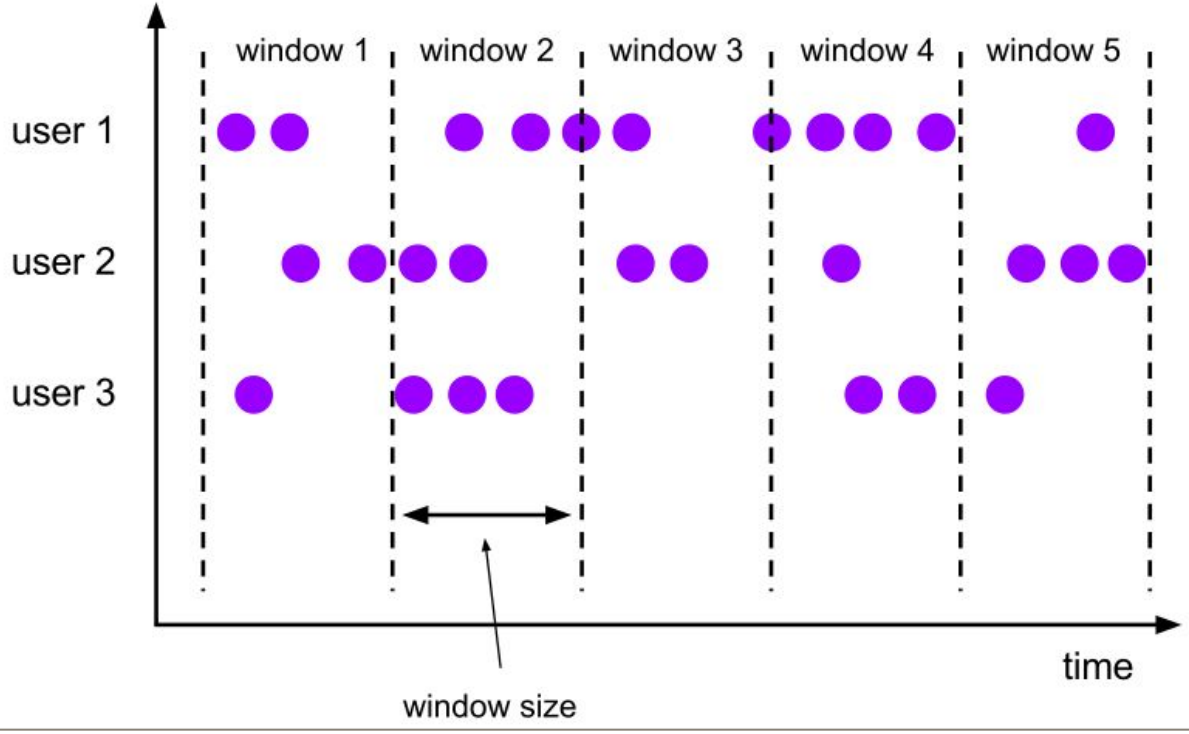


Figure source: https://nightlies.apache.org/flink/flink-docs-master/docs/concepts/time/

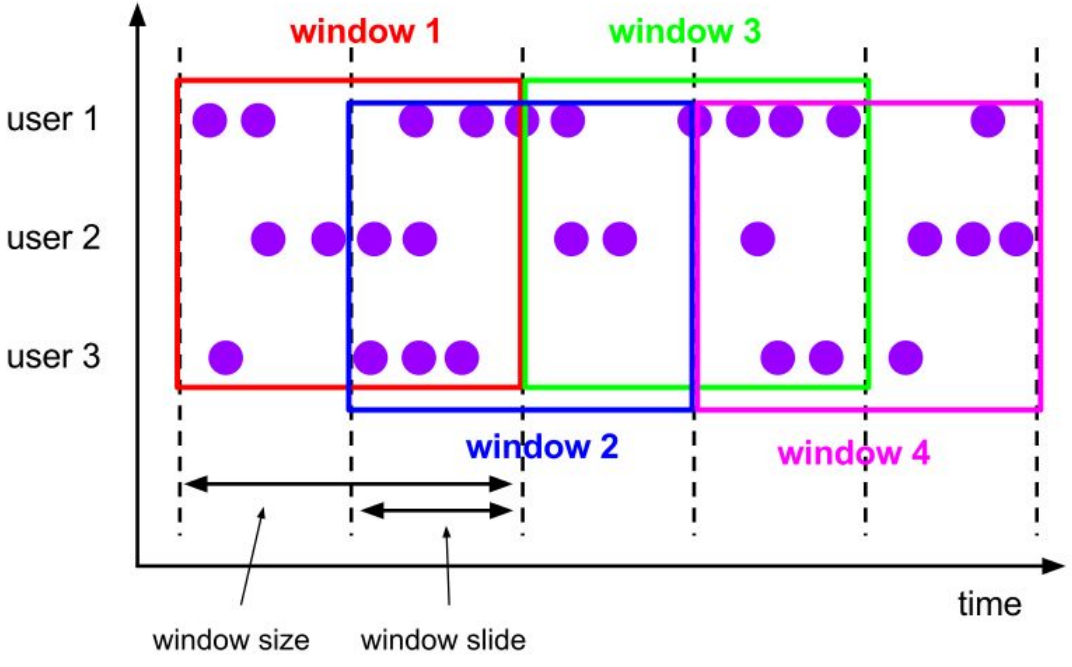
Batch/Tumbling Windows



Use cases: period computation (e.g. stock, temperature, IoT data)

Figure source: https://nightlies.apache.org/flink/flink-docs-master/docs/dev/datastream/operators/windows/

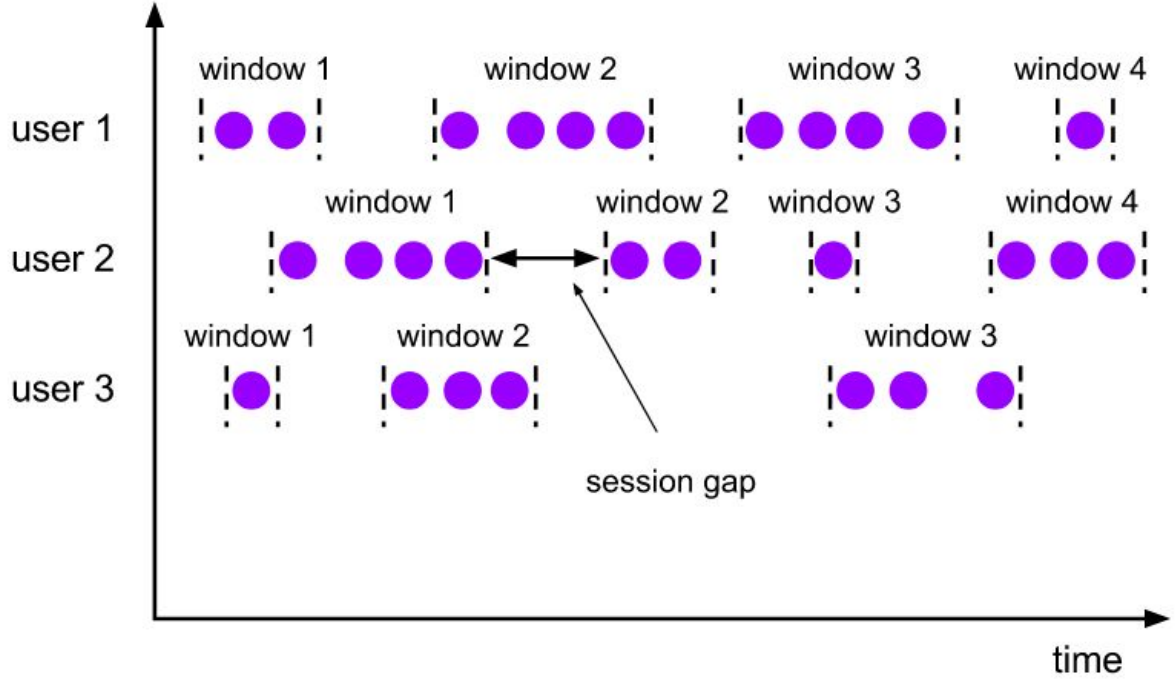
Sliding windows



Use cases:
Moving average

Figure source: https://nightlies.apache.org/flink/flink-docs-master/docs/dev/datastream/operators/windows/

Session Windows



Use cases:
Web/user activities
clicks

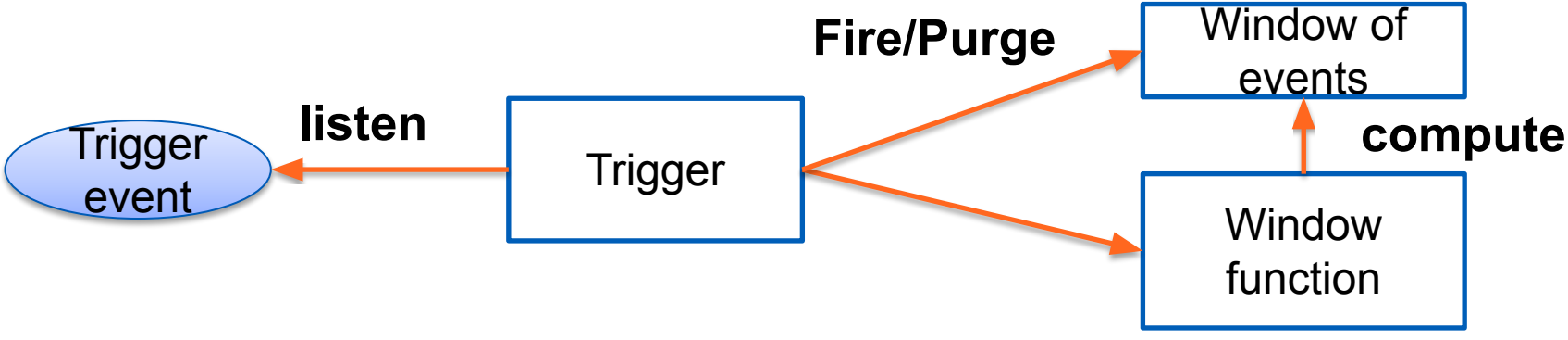
Figure source: https://nightlies.apache.org/flink/flink-docs-master/docs/dev/datastream/operators/windows/

Window Functions

- **Reduce Function**
  - Reduce through the combination of two inputs
- **Aggregate Function**
  - Add an input into an accumulator
- **ProcessWindow Function**
  - Get all elements of the windows and many other information so that you can do many tasks

Triggers & Evictor

- **Trigger: determine if a window is ready for window functions**



Evictor: actions after the trigger fires and before and/or after the windows function is called

Fault tolerance

- **Principles: checkpointing, restarts operators from the latest successful checkpoints**
- **Need support from data stream sources/sinks w.r.t. (end-to-end) exactly once message receiving and result delivery**

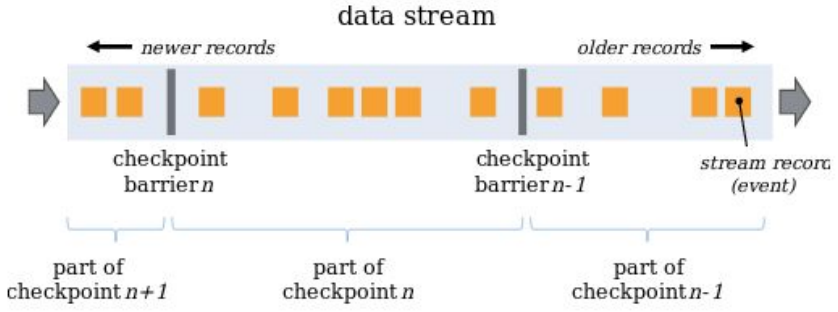


Figure source:
https://nightlies.apache.org/flink/flink-docs-release-1.18/docs/concepts/stateful-stream-processing/

Example with Base Transceiver Station

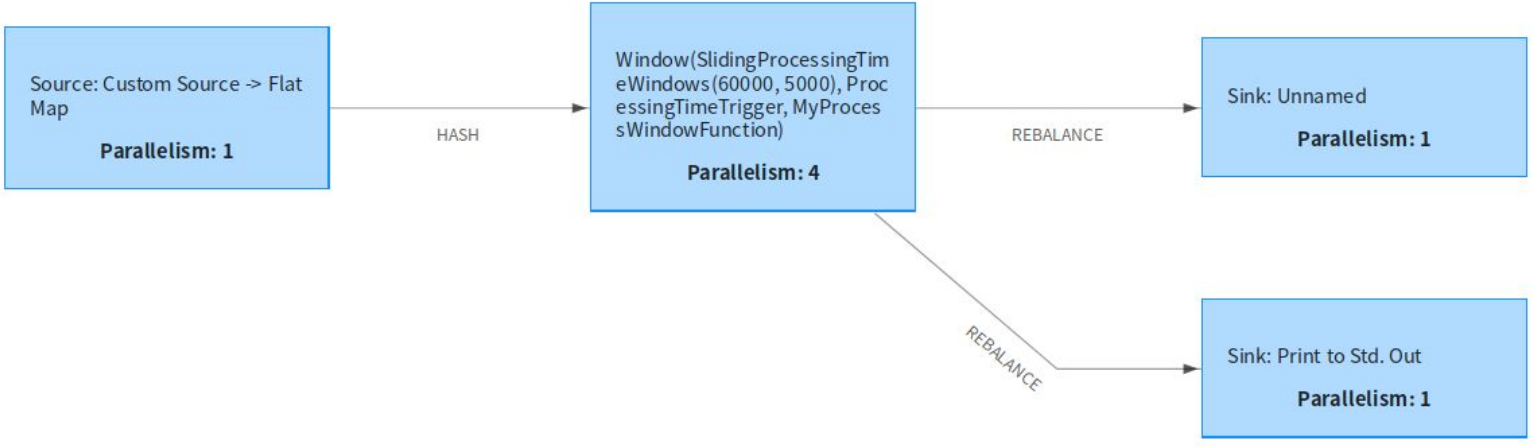
Data in our git

station_id,datapoint_id,alarm_id,event_time,value,valueThreshold,isActive,storedtime
1161115016,121,308,2017-02-18 18:28:05 UTC,240,240,false,
1161114050,143,312,2017-02-18 18:56:20 UTC,28.5,28,true,
1161115040,141,312,2017-02-18 18:22:03 UTC,56.5,56,true,
1161114008,121,308,2017-02-18 18:34:09 UTC,240,240,false,
1161115040,141,312,2017-02-18 18:20:49 UTC,56,56,false,

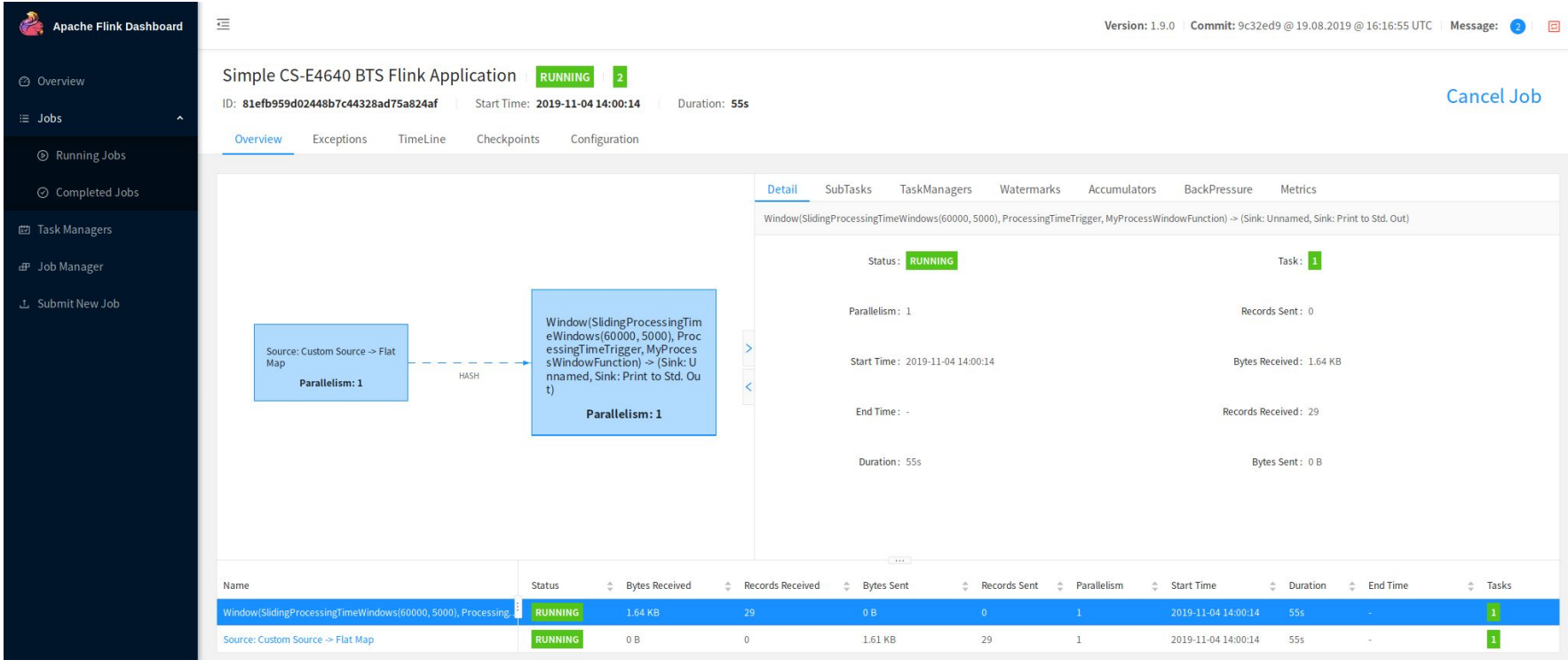
See the code in our git:

https://github.com/rdsea/bigdataplatforms/tree/master/tutorials/streamingwithflink/

Simple example



Monitoring


Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit New Job
Version: 1.9.0 | Commit: 9c32ed9 @ 19.08.2019 @ 16:16:55 UTC | Message: 2
Simple CS-E4640 BTS Flink Application RUNNING 2
ID: 81efb959d02448b7c44328ad75a824af | Start Time: 2019-11-04 14:00:14 | Duration: 55s
Cancel Job
Overview Exceptions TimeLine Checkpoints Configuration
Source: Custom Source -> Flat Map
Parallelism: 1
HASH
Window(SlidingProcessingTimeWindows(60000, 5000), ProcessingTimeTrigger, MyProcessWindowFunction) -> (Sink: Unnamed, Sink: Print to Std. Out)
Parallelism: 1
Detail SubTasks TaskManagers Watermarks Accumulators BackPressure Metrics
Window(SlidingProcessingTimeWindows(60000, 5000), ProcessingTimeTrigger, MyProcessWindowFunction) -> (Sink: Unnamed, Sink: Print to Std. Out)
Status: RUNNING
Task: 1
Parallelism: 1
Records Sent: 0
Start Time: 2019-11-04 14:00:14
Bytes Received: 1.64 KB
End Time: -
Records Received: 29
Duration: 55s
Bytes Sent: 0 B
Name Status Bytes Received Records Received Bytes Sent Records Sent Parallelism Start Time Duration End Time Tasks
Window(SlidingProcessingTimeWindows(60000, 5000), Processing... RUNNING 1.64 KB 29 0 B 0 1 2019-11-04 14:00:14 55s - 1
Source: Custom Source -> Flat Map RUNNING 0 B 0 1.61 KB 29 1 2019-11-04 14:00:14 55s - 1

Summary

- **Focus:**
  - Practical programming with one of the stacks:
    - *Apache Flink Stream API (with different connectors)*
    - *Kafka Streams*
  - Check the common concepts in other tools/systems
- **Action:**
  - Work on use cases where you can use stream analytics (as a user/developer) ⇒ there are many interesting analytics
  - Provision services for stream processing (as a platform)

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io