

Apache Flink

Borja Moreno Pozo

INDICE

- ▶ ¿Que es Apache Flink Streaming?
- ▶ Arquitectura
- ▶ Ejemplo

Flink - Definición

- ▶ Es un motor de procesamiento Open Source de flujos de datos en streaming (y en batch) distribuido y tolerante a fallos escrito en Java y Scala
- ▶ Desarrollo liderado por DataArtisans
- ▶ Última versión estable 1.4

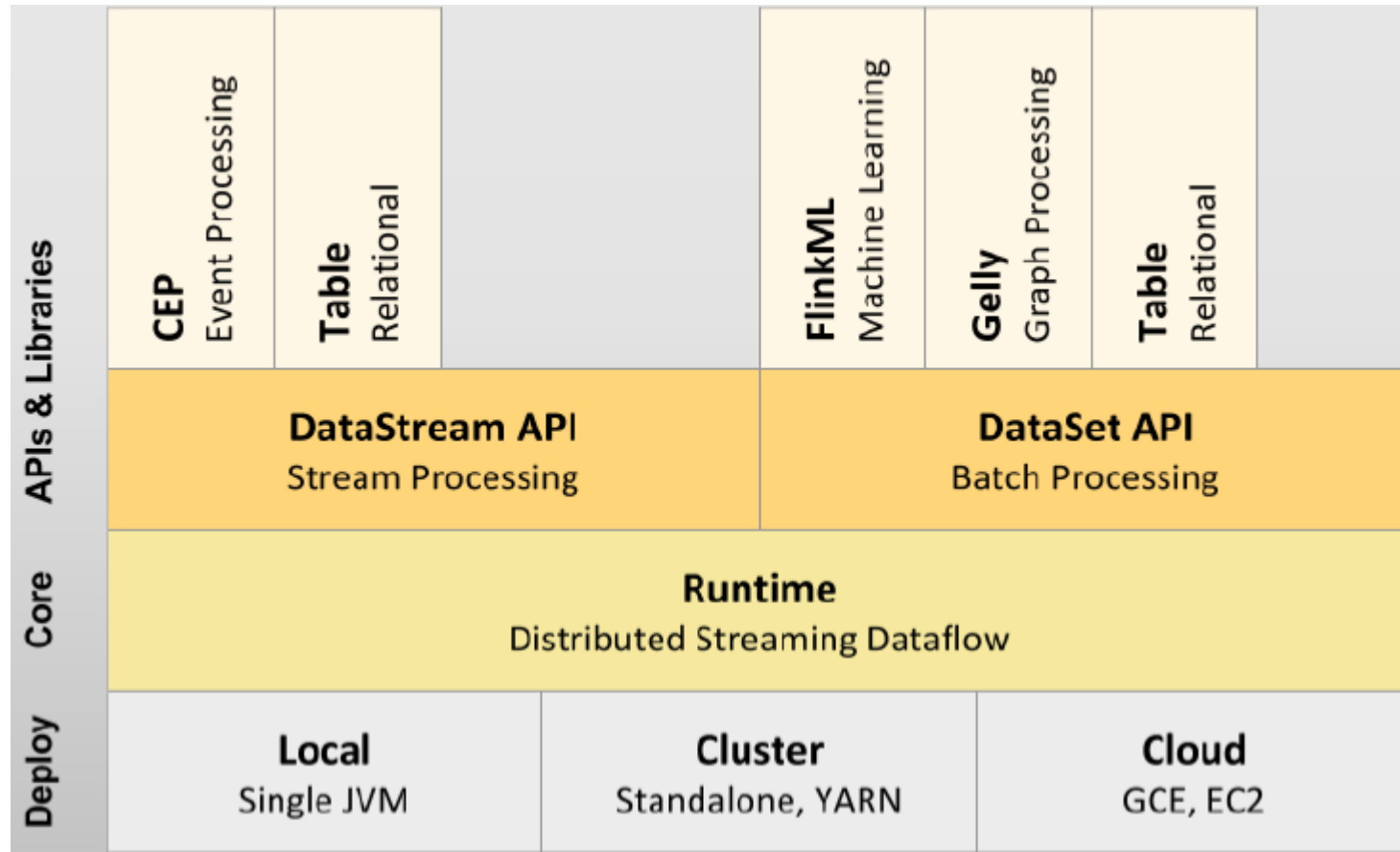
Flink - Breve historia

- ▶ **2010:** “Stratosphere: Information Management on the Cloud”
Fundado por la fundación alemana de investigaciones (DFG) empezó como una colaboración de la Universidad Técnica de Berlín (TU), la Universidad de Humboldt y el Instituto Hasso-Plattner.
- ▶ **Marzo 2014:** Flink se convierte en un fork de Stratosphere y comienza a ser un Proyecto dentro de Apache Incubator
- ▶ **Diciembre 2014:** Flink es aceptado como Proyecto top-level dentro de la Apache Foundation

Flink - Componentes

- ▶ Flink incluye varias APIs para crear aplicaciones que utilicen el motor de procesamiento de Flink:
 - ▶ Dataset API: para datos estáticos embebidos utilizando Java, Scala o Python
 - ▶ DataStream API: para streams de datos continuos utilizando Java, Scala o Python
 - ▶ Table API: utiliza un lenguaje SQL-like usando Java y Scala
- ▶ Flink incluye distintas librerías de dominio específico:
 - ▶ Flink Machine learning: algoritmos supervisados, sistemas recomendación
 - ▶ Gelly: librería y API de procesamiento de grafos
 - ▶ CEP: librería de procesamiento complejo de eventos

Flink - Componentes



Flink - Características

- ▶ Streaming first (enfoque arquitectura Kappa)
 - ▶ Alto throughput y baja latencia
 - ▶ Flow (events) vs batches
 - ▶ Ofrece política exactly-once para procesamiento con estado
 - ▶ Flink ofrece distintos tipos de políticas de garantías de procesamiento
 - ▶ Modelo de procesamiento continuo de datos basado en control de flujo y operadores persistentes (no hay necesidad de correr nuevas tareas)
 - ▶ Permite procesar eventos que llegan sin orden
 - ▶ Ofrece gestionar los eventos utilizando distintos tipos de ventanas (temporales, en función del dato, por sesiones, ...)

Flink - Características II

- ▶ Ofrece tolerancia a fallos mediante una arquitectura distribuida de snapshots (fotos de un instante) ligera
- ▶ Un único motor de procesamiento para batch y para streaming
- ▶ El procesamiento en batch funciona como un caso de uso especial de streaming
- ▶ Sistema de gestión de memoria propio
- ▶ Permite el cálculo de algoritmos iterativos e incrementales (deltas)
- ▶ Optimizador propio de código
- ▶ Amplio abanico de librerías adicionales (Gelly, FlinkML, CEP, ...)
- ▶ Integración con otras tecnologías Big Data

Flink - Elementos básicos en Flink

- ▶ **DataSet:** representación abstracta que define una colección de datos finita inmutable del mismo tipo que puede contener datos duplicados
- ▶ **DataStream:** colección de datos inmutables continua en el tiempo del mismo tipo.
- ▶ **Transformaciones:** transformaciones de datos de uno o más DataSets/DataStreams en un o varios DataSets/DataStreams:
 - ▶ Compartidas: Map, Flatmap, MapPartition, Filter, Reduce, Union
 - ▶ DataSets: Aggregate, Join, Cogroup
 - ▶ DataStreams: transformaciones de ventana (Window, Window Reduce, ...)

Flink - Orígenes y salida de datos

▶ Data Sources

▶ File-based

- ▶ `readTextFile(path)`, `readTextFileWithValue(path)`, `readFile(path)`, ...

▶ Socket-based

- ▶ `socketTextStream(streaming)`

▶ Collection-based

- ▶ `fromCollection(Seq)`, `fromCollection(iterator)`, `fromElements(elements: _*)`

▶ Propias

- ▶ Añadir fuentes desde Kafka, ...

▶ Data Sinks (similar a las acciones de Spark):

▶ `writeAsText()`

▶ `writeAsCsv()`





▶ `Print()` / `printToErr()`

▶ `Write()`

▶ `writeToSocket`

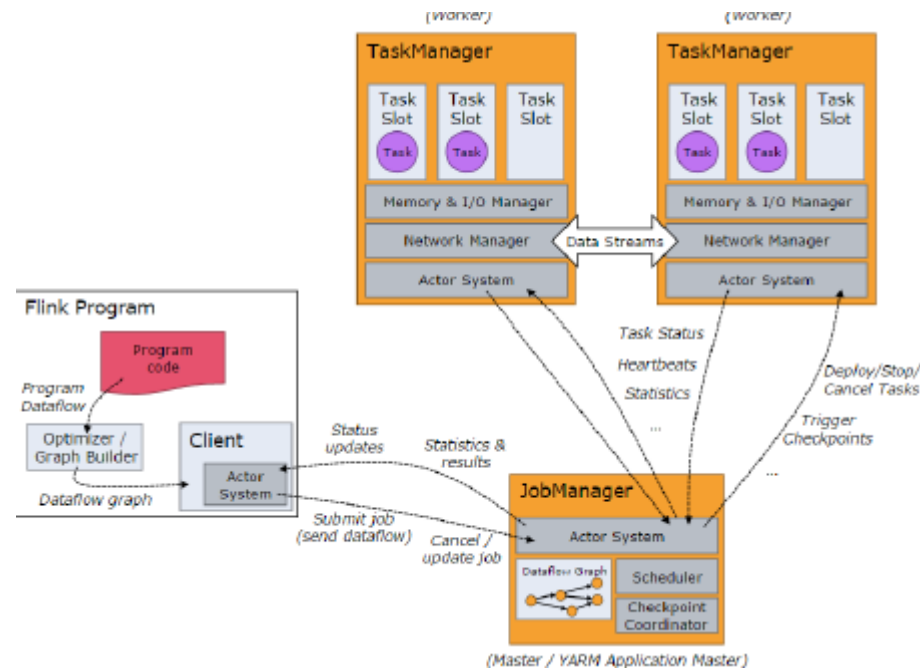
▶ `addSink` como Kafa

Comparación de motores de procesamiento

Engine comparison				
				
API	MapReduce on k/v pairs	k/v pair Readers/Writers	Transformations on k/v pair collections	Iterative transformations on collections
Paradigm	MapReduce	DAG	RDD	Cyclic dataflows
Optimization	none	none	Optimization of SQL queries	Optimization in all APIs
Execution	Batch sorting	Batch sorting and partitioning	Batch with memory pinning	Stream with out-of-core algorithms

Flink - Modelo de procesamiento

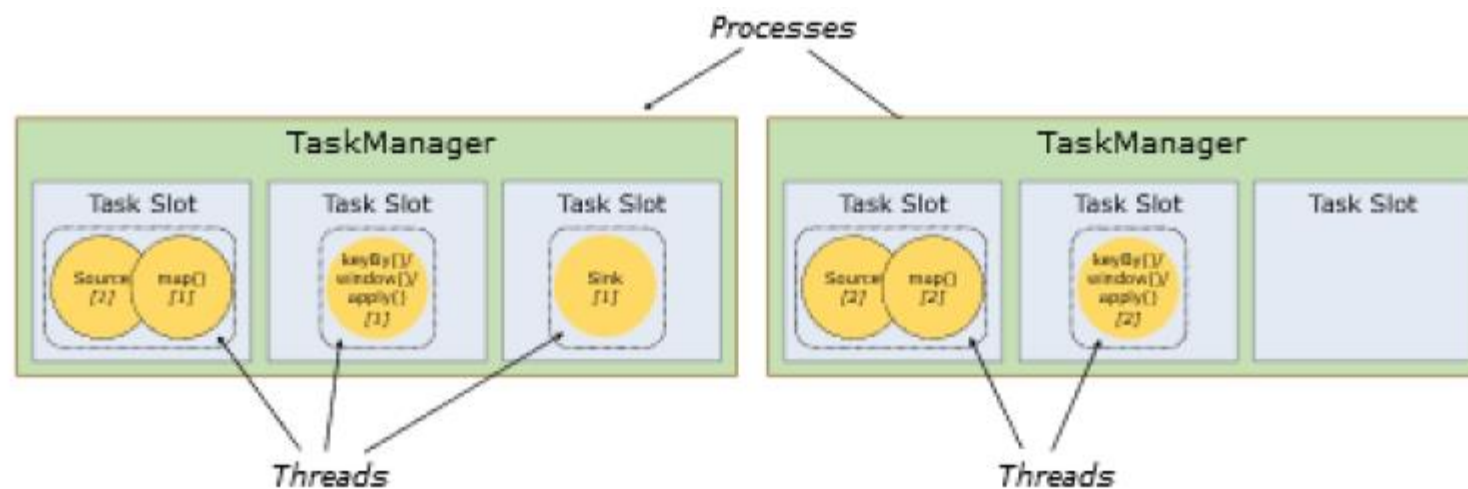
- Procesamiento:
 - JobManager: coordinador del sistema
 - TaskManagers: workers que ejecutan partes de los programas



Flink - Arquitectura de procesamiento

- ▶ Cada worker (TaskManager) es un proceso de la JVM
- ▶ Puede ejecutar un número n de subtareas en distintos threads
- ▶ El número de subtareas vendrá definido por la propiedad task slots
- ▶ A cada subtask le corresponderá la parte respectiva de memoria de la máquina (esto no ocurre con la CPU)
- ▶ Si en un worker tenemos varias subtareas esto significa:
 - ▶ Comparten la misma JVM
 - ▶ Comparten conexiones TCP
 - ▶ Comparten datos y estructuras

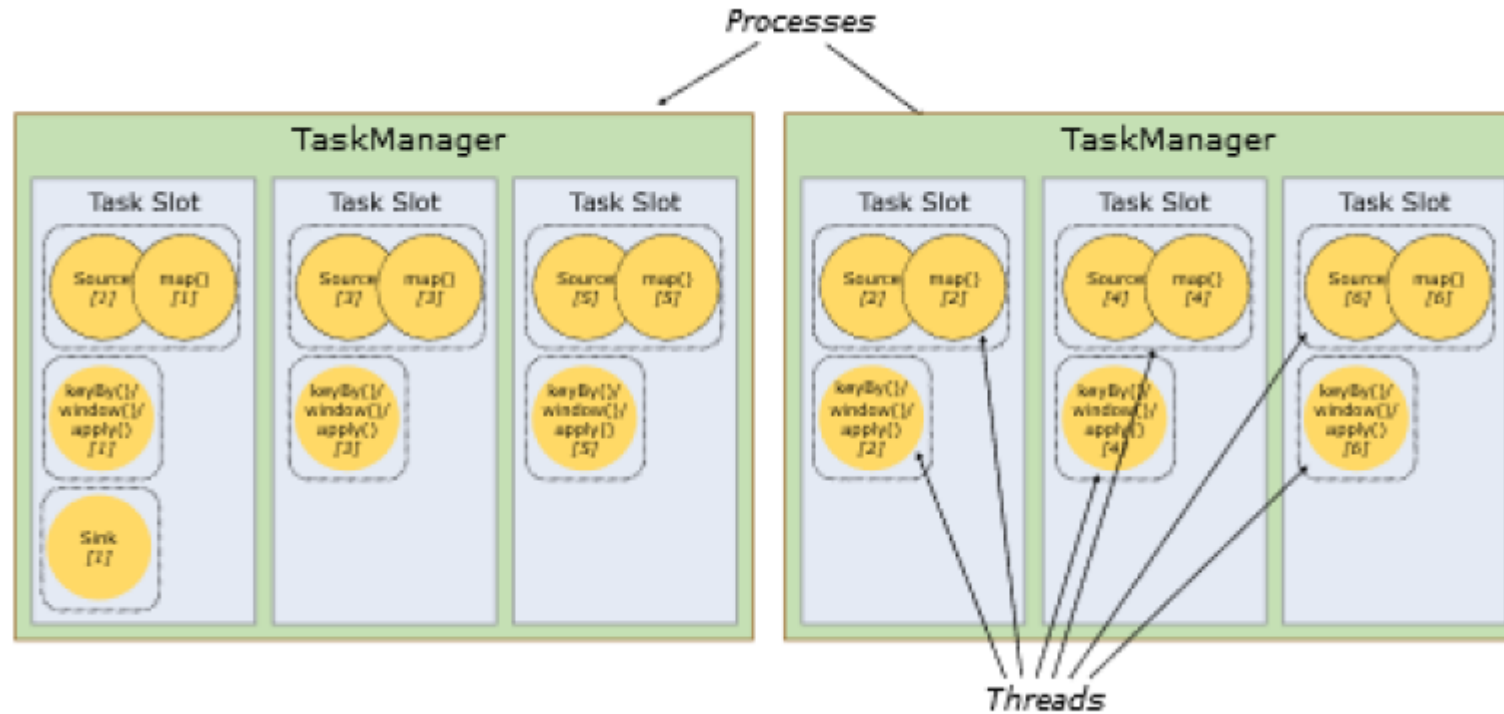
Flink - Arquitectura de procesamiento



Flink - Arquitectura de procesamiento

- ▶ Flink permite que distintas subtareas compartan slots
- ▶ Deben ser del mismo job
 - ▶ Un pipeline del job puede 'caber' en un único slot
- ▶ Beneficios:
 - ▶ Un cluster de Flink va a necesitar tantos slots como el mayor grado de paralelismo utilizado por el job
 - ▶ Incrementa el grado de paralelismo. Las tareas menos intensivas pueden moverse a un slot de manera independiente de las tareas más intensivas
- ▶ Este comportamiento se puede compartir via API
- ▶ Una buena regla es, que un número de slots de tareas de ser igual al número de cores

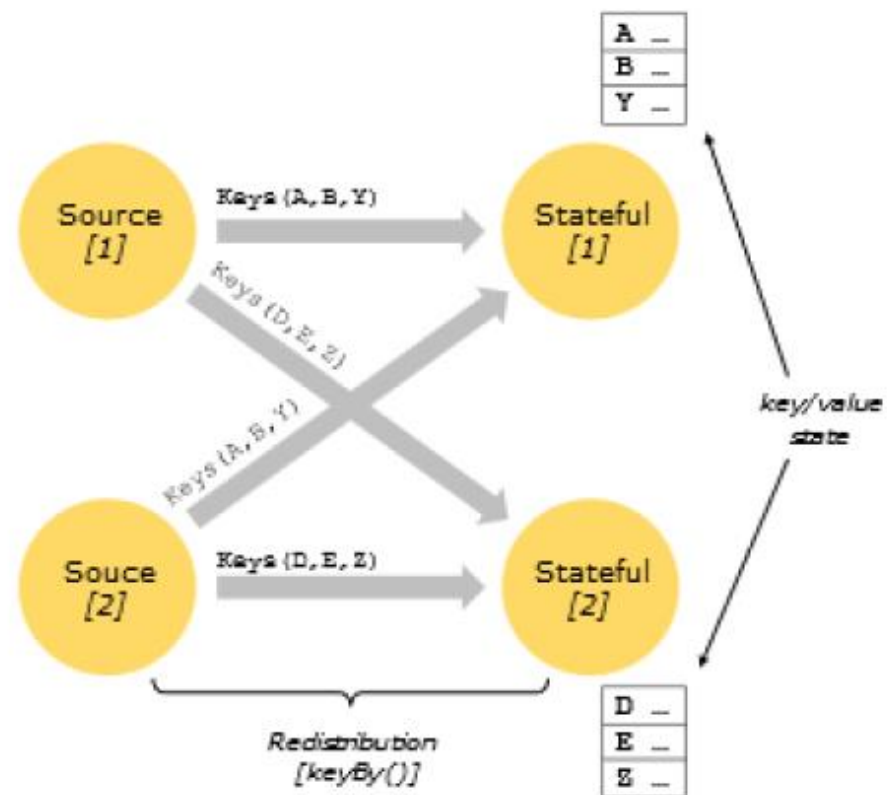
Flink - Arquitectura de procesamiento



Flink - Mantenimiento del estado

- ▶ Algunas operaciones pueden recordar estados (p.ej: ventanas)
- ▶ El estado se almacena mediante una clave valor. El tipo de estructura donde guardar el estado puede ser varias:
 - ▶ Hashmap en memoria
 - ▶ Base de datos externa (RocksDB)
- ▶ El estado es particionado y distribuido entre los streams
- ▶ Por lo tanto, el mantenimiento del estado se realiza como operaciones locales a la partición garantizando consistencia

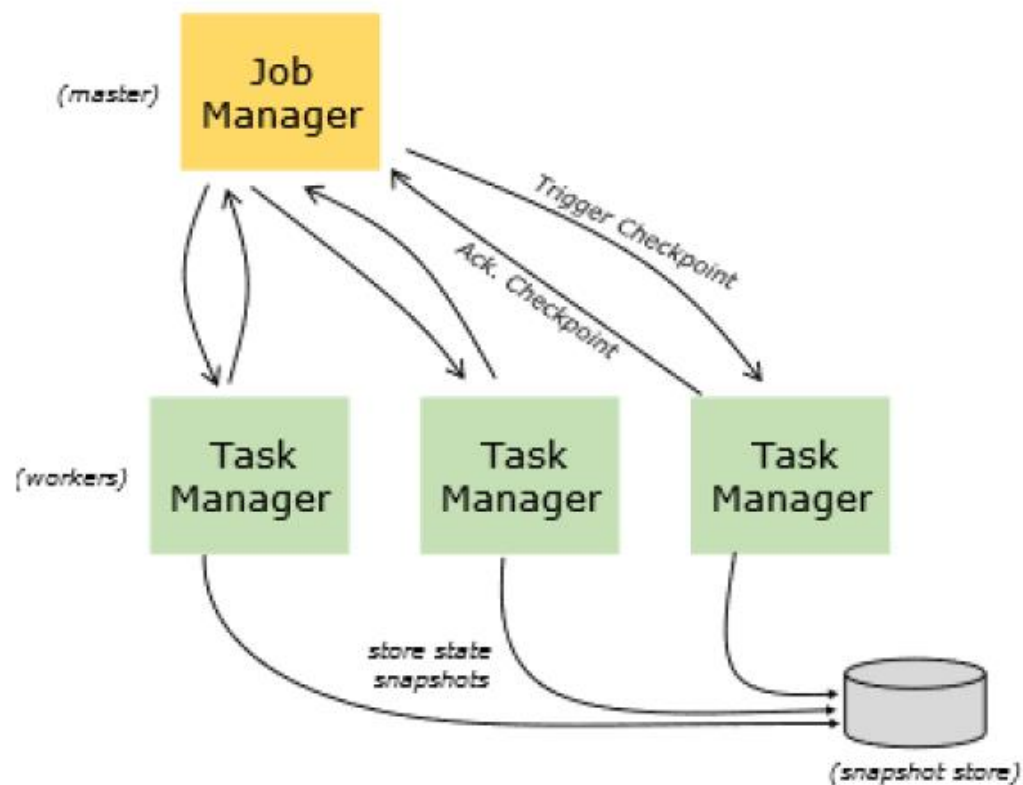
Flink - Mantenimiento del estado



Flink - Checkpoint para tolerancia a fallos

- ▶ Flink permite hacer reprocesamiento de streams desde checkpoints
- ▶ Un checkpoint define un punto y un estado en el stream desde el cual se puede volver a reprocesar
- ▶ Los workers serán los encargados de ir realizando checkpointing con un intervalo predefinido
- ▶ El mecanismo de checkpointing en Flink garantiza política de ejecución de exactamente una vez

Flink - Checkpoint para tolerancia a fallos



Flink - Modelo de ejecución

- ▶ Flink es un sistema de capas. Las diferentes capas del stack se construyen una encima de otra y funcionan como abstracciones del programa que se va a ejecutar
 - ▶ La capa de ejecución recibe un programa en forma de un JobGraph. Un JobGraph es un flujo de datos paralelo genérico con tareas que consumen y producen streams de datos
 - ▶ Tanto la API de DataStream y DataSet genera JobGraphs a través de procesos de compilación separados. El API de DataSet utiliza un optimizador para determinar el mejor plan para el programa, mientras que la API de DataStream utiliza un stream builder
 - ▶ El JobGraph es ejecutado de forma perezosa (lazy) de acuerdo a uno de los distintos métodos de despliegue disponibles en Flink (local, remoto o YARN)

Flink - Modelo de ejecución (streaming)

- ▶ Cuando se ejecuta un job de Flink, éste se mapea a un dataflow en streaming:
 - ▶ Tiene streams
 - ▶ Operadores de transformación
- ▶ Normalmente 1 transformación = 1 operador en dataflow

Flink - Modelo de ejecución (streaming)

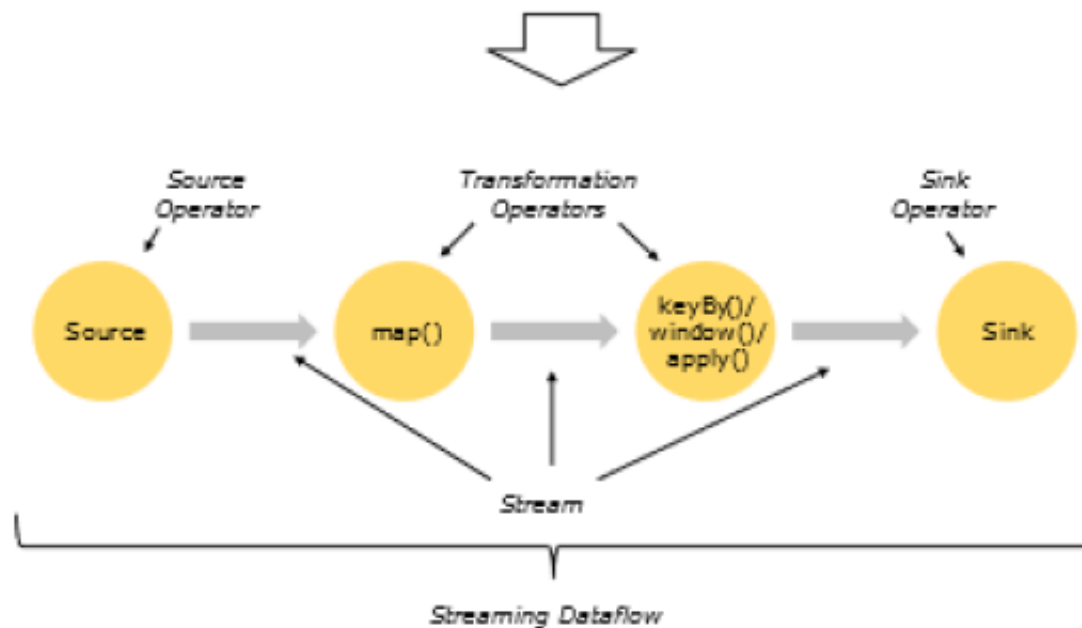
```
DataStream<String> lines = env.addSource(  
    new FlinkKafkaConsumer<> (...);  
  
DataStream<Event> events = lines.map((line) -> parse(line));  
  
DataStream<Statistics> stats = events  
    .keyBy("id")  
    .timeWindow(Time.seconds(10))  
    .apply(new MyWindowAggregationFunction());  
  
stats.addSink(new RollingSink(path));
```

Source

Transformation

Transformation

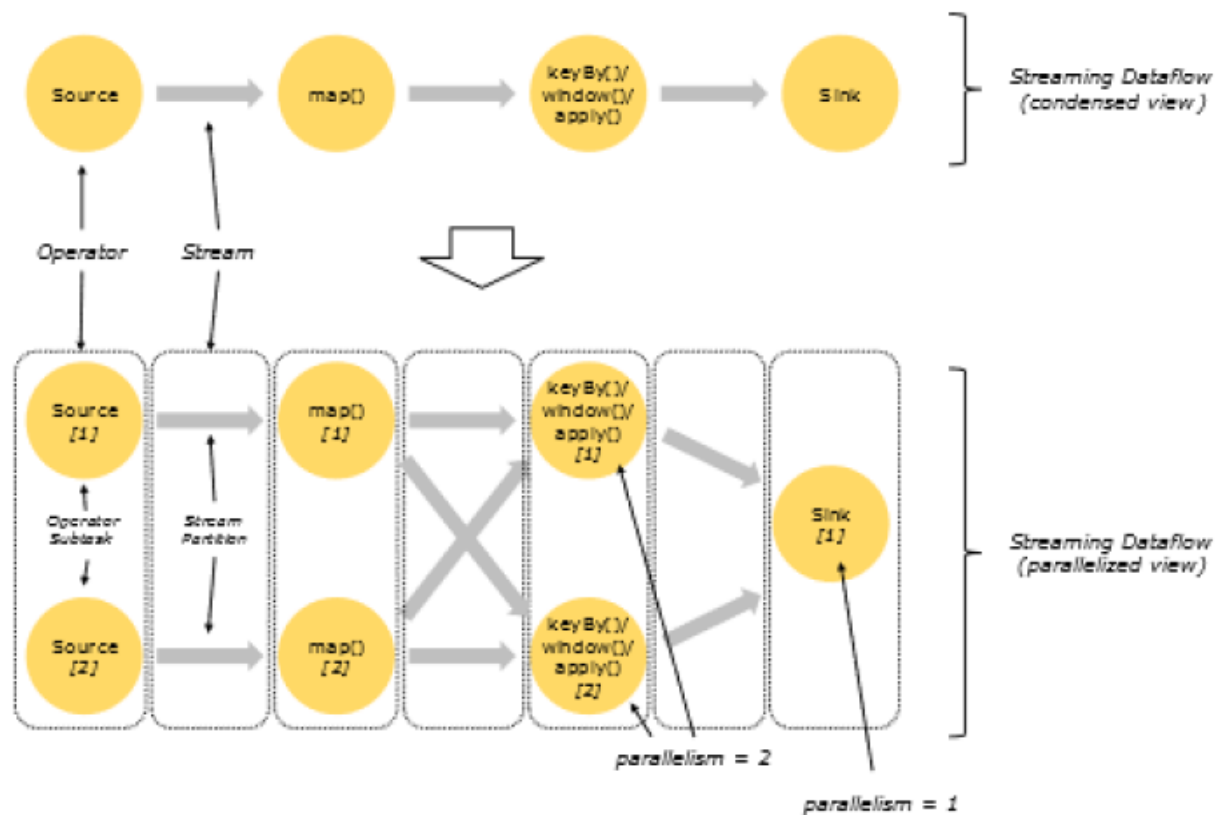
Sink



Flink - Flujos de datos paralelos

- ▶ Los programas en Flink se ejecutan de manera paralela y distribuida
- ▶ Streams se dividen en streams partitions
- ▶ Operadores se dividen en subtareas de operadores
- ▶ Cada subtask de operador se ejecuta de manera independiente una de otra, en diferentes threads y en diferentes instancias o contenedores
- ▶ El número de subtareas de un operador define el paralelismo de ese operador
- ▶ Diferentes operadores pueden tener diferentes grado de paralelismo

Flink - Flujos de datos paralelos



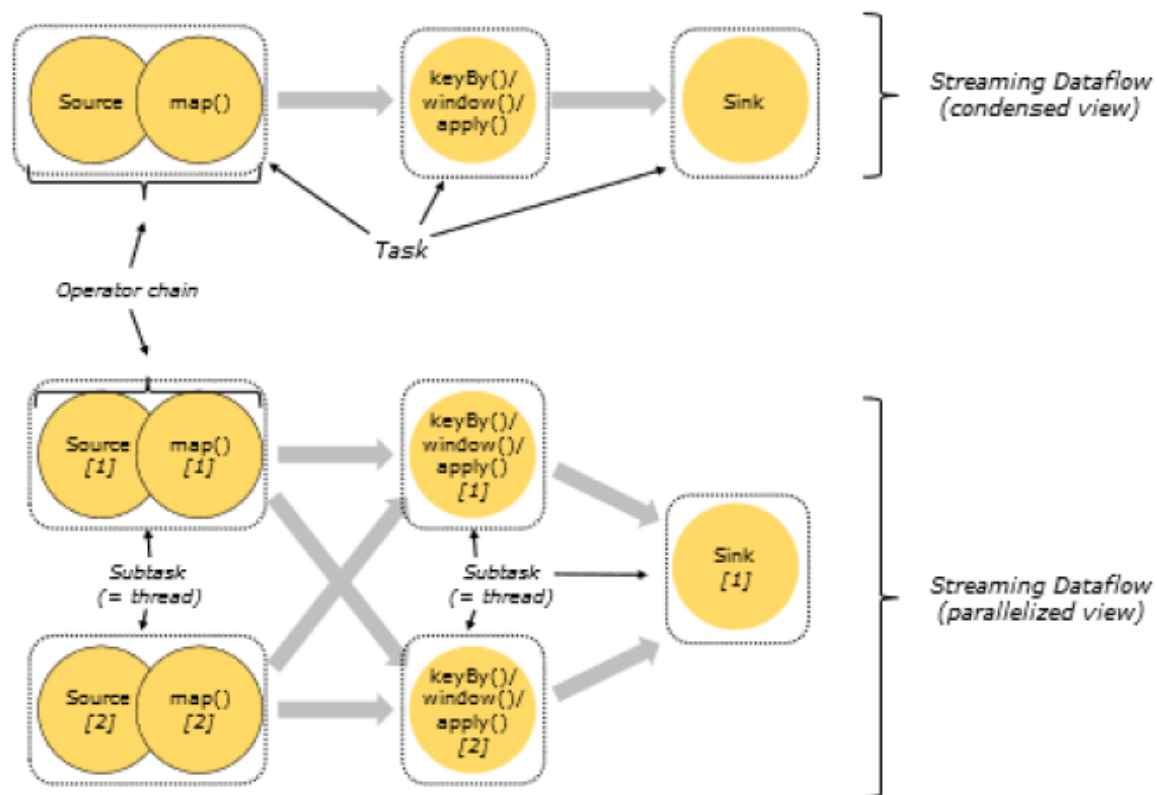
Flink - Flujos de datos paralelos

- ▶ El flujo de datos entre los streams en Flink puede ser:
- ▶ **Uno a uno:** preserva particionamiento y el orden de los elementos. Es decir, una subtask tiene los mismo elementos y en el mismo orden que la subtask de su operador fuente
 - ▶ Ejemplo: en la gráfica anterior entre `source[1]` y `map[1]`
- ▶ **Redistributiva:** cambia el particionamiento de los streams, Cada subtask de un operador manda datos a diferentes subtasks destino, dependiendo de la transformación realizada
 - ▶ Ejemplo: en la gráfica anterior entre `map[1]` y `KeyBy[2]`

Flink - Encadenamiento de Task y operadores

- ▶ Flink encadena distintas operaciones en tareas
- ▶ Cada tarea es ejecutada en un thread
- ▶ Optimización de código:
 - ▶ Reduce el overhead de gestión de threads
 - ▶ Reduce el buffering
 - ▶ Incrementa el throughput general
 - ▶ Reduce latencia
- ▶ El mecanismo de encadenamiento es configurado via API

Flink - Encadenamiento de Task y operadores



Los 8 requisitos del procesamiento de datos en streamsing

- ▶ Definido por Michael Stonebraker y otros en el siguiente paper: <http://cs.brown.edu/~ugur/8rulesSigRec.pdf>
 - ▶ **Pipelining:** Flink está construido mediante pipelines
 - ▶ **Replay:** Flink es capaz de reprocesar datos
 - ▶ **Operator state:** el flujo pasa a través de distintos operadores
 - ▶ **State backup:** Flink permite mantener el estado
 - ▶ **High-level language(s):** Java, Scala, Python (beta)
 - ▶ **Integration with static sources:** Puede leer y escribir con fuentes externas
 - ▶ **High availability**

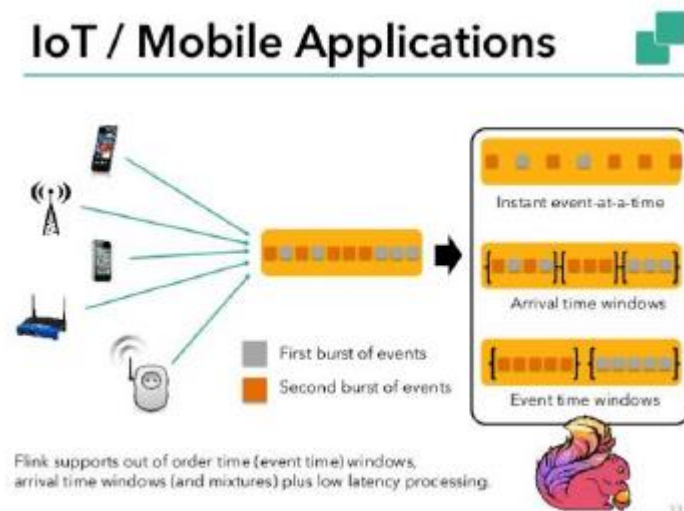
Flink - Notas I

- ▶ Arquitectura híbrida:
 - ▶ Datos intermedios son gestionados por los operadores
 - ▶ Coordina el “handshake” entre el productor y consumidor de datos
- ▶ Actualmente DataStream API soporta flexible windows
- ▶ Apache SAMOA sobre Flink para Machine Learning para streams
- ▶ Table API (definición en proceso)

Flink - Notas II

- ▶ Flink ofrece distintos tipos de ventana de procesamiento
 - ▶ Instant event-at-a-time
 - ▶ Arrival time windows
 - ▶ Event time windows

https://www.slideshare.net/FlinkForward/k-tzoumas-s-ewen-flink-forward-keynote?qid=ced740f4-8af3-4bc7-8d7c-388eb26f463f&v=qf1&b=&from_search=5



Flink - Conectores

- ▶ Las librerías necesitan ser instaladas por separado

	SOURCE	SINK
Kafka	SI	SI
ElasticSearch		SI
Hadoop FileSystem		SI
RabbitMQ	SI	SI
Twitter Streaming	SI	

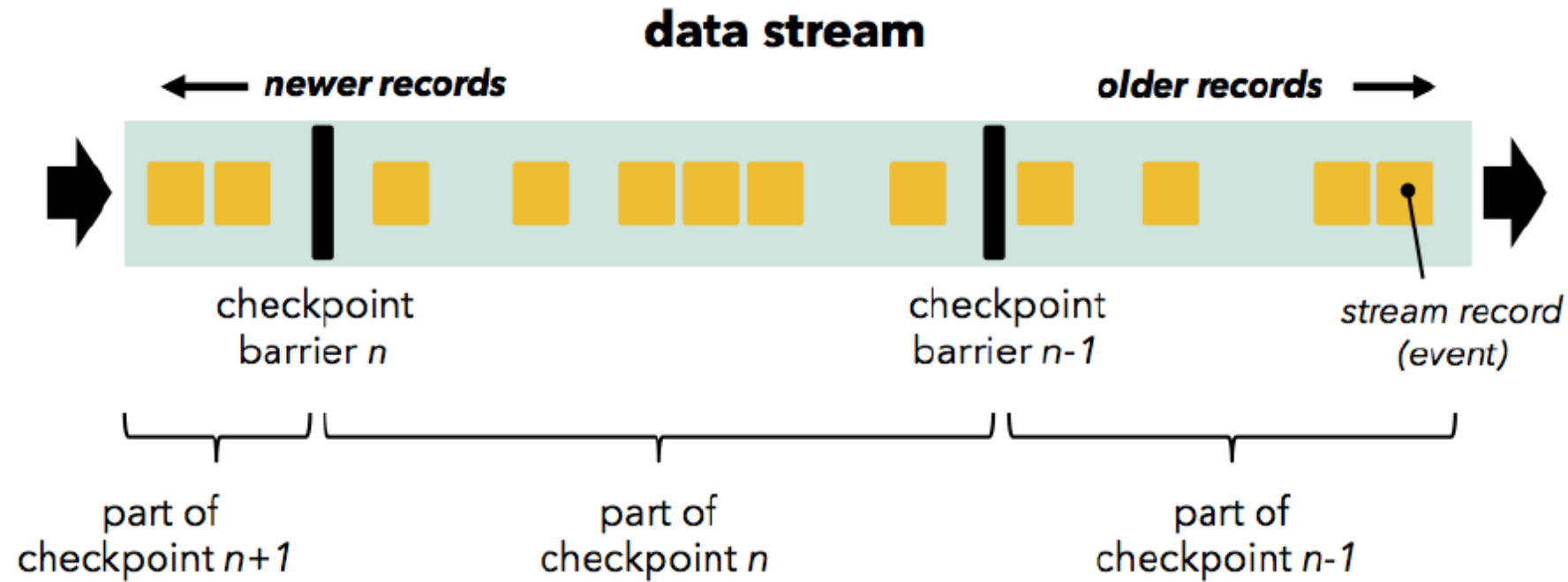
Flink - Complex Event Processing

- ▶ FlinkCEP es una librería de procesamiento de eventos
- ▶ Permite detectar patrones de comportamiento en un stream y definir una secuencia de operaciones
- ▶ Tipo de operaciones:
 - ▶ Begin
 - ▶ Next
 - ▶ FollowedBy
 - ▶ Where
 - ▶ Subtype
 - ▶ Within

Flink - ¿Por qué Flink es mejor que otros ?

- ▶ Flink almacena snapshot de manera consistente del estado actual del sistema sin perder información y sin almacenar duplicados (para ellos se utiliza el algoritmo de Chandy Lamport)
- ▶ Es similar a los motores de micro-batching como Spark o Trident:
 - ▶ Todo el procesamiento entre dos checkpoints o bien son correctos o bien fallan completamente (semántica Exactly once)
- ▶ Pero:
 - ▶ No existen micro-batches. No se necesita esperar a una ventana de tiempo prefijada.
 - ▶ El procesamiento de datos siempre está funcionando, los eventos son tratados según llegan, mientras que los checkpoints ocurren en segundo plano.

Flink - ¿Por qué Flink es mejor que otros ?



Flink - ¿Por qué Flink es mejor que otros ?

▶ Beneficios:

- ▶ Modelo real de procesamiento continuo (baja latencia, control de flujo y modelo de programación realmente orientado a streams de datos)
- ▶ Alto throughput
- ▶ Política de procesamiento exactly-once
- ▶ Separa el desarrollo de la aplicación respecto al control de flujo y respecto al control de throughput del sistema. Cambiar el intervalo de snapshotting no tiene efecto en los resultados del Job, por lo que las aplicaciones que se apoyen en Flink pueden confiar en los resultados obtenidos

Flink - Batch en streaming

- ▶ Flink ejecuta programas batch como un caso especial de programas en streaming
- ▶ Un dataset (tipo de datos) en batch es tratado como un stream de datos
- ▶ Existe una serie de excepciones en batch que no se dan en el procesamiento en streaming:
 - ▶ No se utilizan checkpointings
 - ▶ Las operaciones de estado no se almacenan en parejas clave/valor
 - ▶ El API de Dataset introduce un tipo de sincronización especial