



Módulo Profesional: Big Data Aplicado

Stream Processing

ÍNDICE

1. Resumen

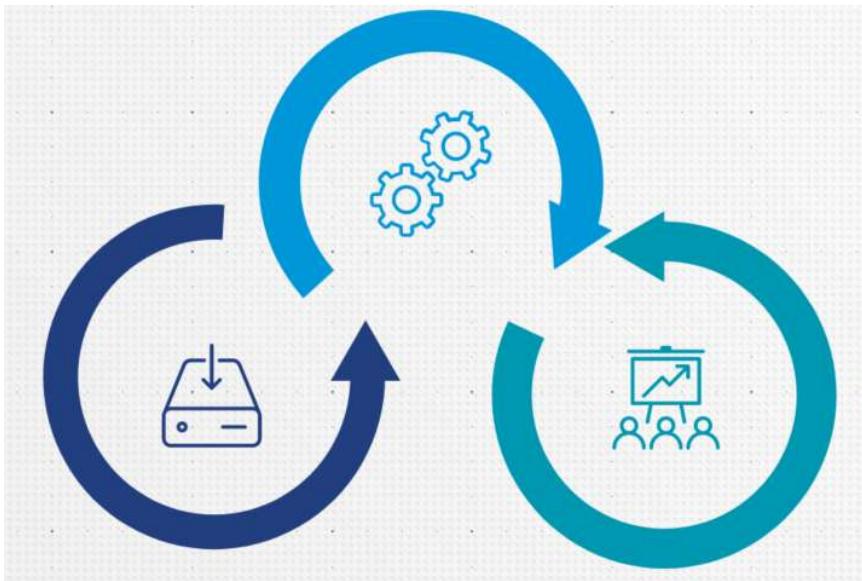
- ¿Qué es el Procesamiento de Streams?
- Beneficios
- Cómo se realiza el procesamiento en tiempo real
- Casos de uso

2. Conceptos

3. Tecnologías

1. Resumen. ¿Qué es Stream Processing?

Stream Processing permite la ingestión, procesamiento y análisis de flujos de datos de manera continua y en tiempo real.



Ingerir datos a la misma velocidad a la que se generan.

Procesar los datos a medida que se reciben.

Obtener información rápidamente, aumentando significativamente el valor de los datos.

1. Resumen. Beneficios

Estamos inmersos en la era de los datos, donde el volumen y la velocidad a la que se generan no tienen precedentes.

Con **Fast Data**, puedes:

- Responder al problema en el momento en que surge.
- Obtener información actualizada.
- Reducir el riesgo mediante datos más precisos y oportunos.
- Disminuir los costes de infraestructura al optimizar la pila tecnológica.



1. Resumen. Real-time architectures

- Las arquitecturas en tiempo real (también conocidas como arquitecturas de streaming) son modelos donde es necesario capturar los datos y reaccionar a ellos, ejecutando las acciones necesarias en el momento en que ocurre el evento.
- En este contexto, surgen nuevos conceptos como latencia, rendimiento (throughput), garantía de entrega y ventanas (windows).
- Generalmente, la arquitectura depende de la implementación utilizada.
- Actualmente, hay un gran auge en este tipo de tecnologías, lo que ha llevado al desarrollo de una multitud de herramientas en los últimos tiempos.

Cómo se realiza el procesamiento en tiempo real

Ten en cuenta que esto incluye ambas implementaciones:

- **Streaming en tiempo real** (True Streaming)
- **Streaming por micro-lotes** (Micro-batch Streaming)

Resultados de baja latencia, aproximados (respuesta lo suficientemente cercana) y/o especulativos (basados en suposiciones).

Estas características no son un requisito para el procesamiento de flujos, pero están presentes en algunas implementaciones.

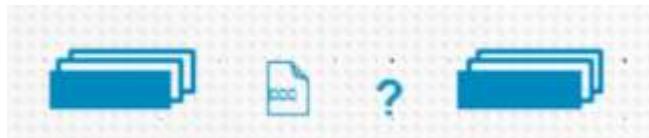
1. Resumen. Casos de uso



2. Conceptos. Garantías de entrega

Como máximo una vez - "enviar y olvidar" “fire and forget”

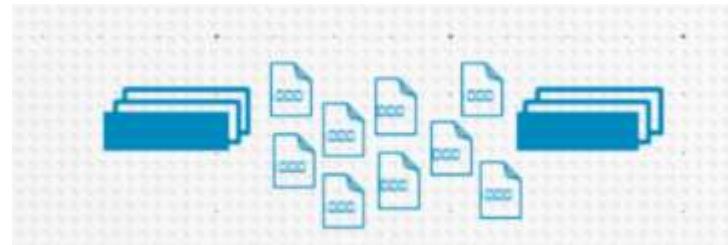
- Los mensajes pueden perderse, pero nunca se vuelven a entregar.
- El mensaje se envía, pero al remitente no le importa si se recibe o se pierde.
- Si la pérdida de datos no es una preocupación, lo que podría ser el caso en el monitoreo de telemetría, por ejemplo, este modelo no impone ningún coste adicional para garantizar la entrega del mensaje, como requerir confirmaciones de los consumidores.
- Es el comportamiento más fácil y potente de soportar.



2. Conceptos. Garantías de entrega

Al menos una vez “At least once”

- Los mensajes nunca se pierden, pero pueden ser reenviados.
- La retransmisión de un mensaje ocurrirá hasta que se reciba una confirmación.
- Dado que una confirmación retrasada del receptor puede estar en vuelo cuando el remitente reenvía el mensaje, el mensaje puede ser recibido una o más veces.
 - Este es el modelo más práctico cuando la pérdida de mensajes no es aceptable (por ejemplo, en transacciones bancarias), pero se pueden producir duplicaciones.



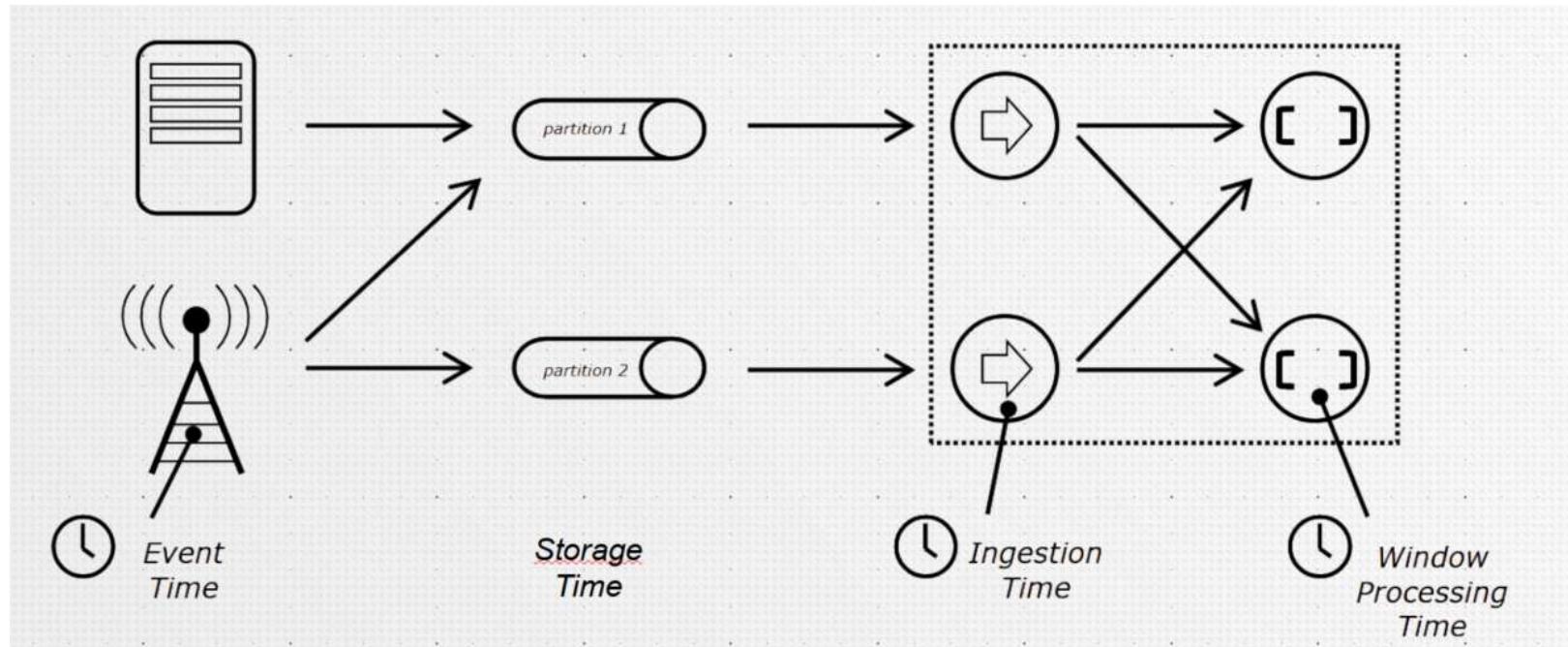
2. Conceptos. Garantías de entrega

Exactamente una vez “Exactly once”

- Los mensajes nunca se pierden ni se vuelven a entregar.
- Garantiza que un mensaje sea recibido una vez y solo una vez, y que nunca se pierda ni se repita.
- Este es el escenario ideal porque es el más fácil de razonar al considerar la evolución del estado del sistema.
 - También es imposible de implementar en el caso general, pero puede implementarse con éxito en casos específicos (al menos con un alto porcentaje de fiabilidad).
 - Kafka proporciona esta característica desde la versión v0.11.0.0, en junio de 2017.
 - Garantías de durabilidad para la publicación de un mensaje – commit de mensajes.
 - Garantías al consumir un mensaje – commit de offsets.
 - Otros sistemas proporcionan semánticas de "exactamente una vez", pero no consideran que los consumidores o productores puedan fallar.



2. Conceptos. Time



Fuente: https://ci.apache.org/projects/flink/flink-docs-release-1.3/dev/event_time.html

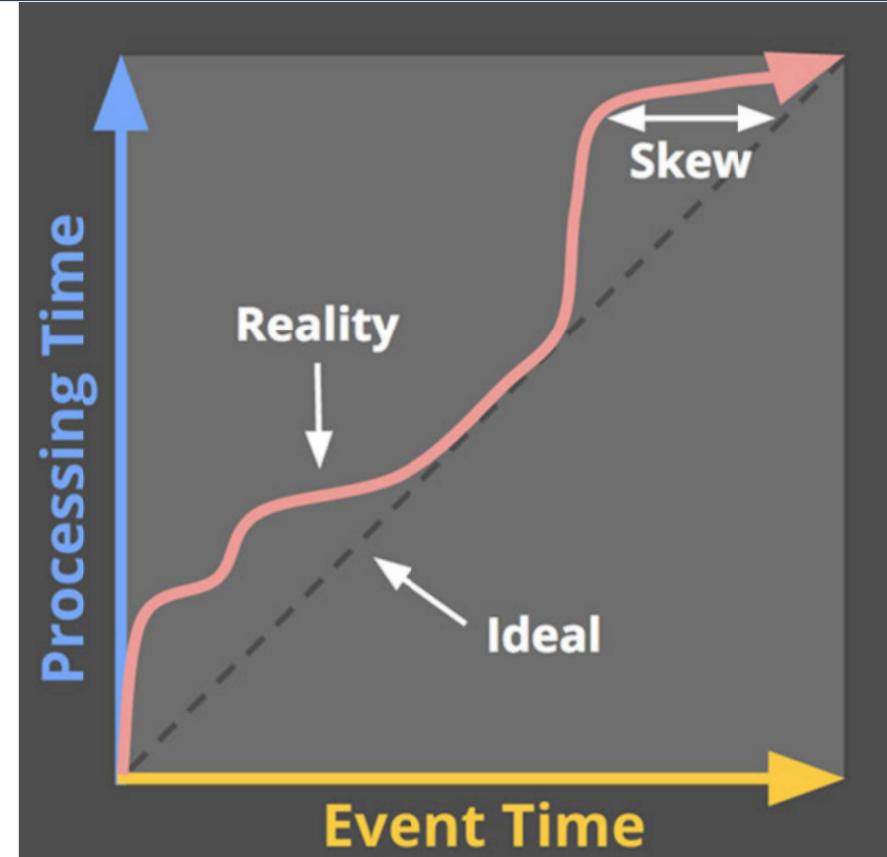
2. Conceptos. Skew

Causas:

Limitaciones de recursos compartidos, como congestión de red, particiones de red o CPU compartida en un entorno no dedicado.

Causas relacionadas con el software, como la lógica de sistemas distribuidos, la contención, etc.

Características de los propios datos, incluyendo la distribución de claves, la variación en el rendimiento (throughput) o la variación en el desorden.



2. Conceptos. Event Time

- El **tiempo del evento** es el momento en el que ocurrió cada evento individual en un dispositivo.
- Típicamente está incrustado en los registros del lado del sistema fuente, por lo que se puede extraer del registro.
- Es **determinista**. El tiempo del evento proporciona resultados correctos incluso para eventos fuera de secuencia, eventos tardíos o la reproducción de datos desde copias de seguridad o registros persistentes. El tiempo depende de los datos, no de un reloj de pared.
- La sincronización de eventos a menudo introduce **un grado de latencia** debido a la naturaleza de esperar eventos tardíos y fuera de orden.

2. Conceptos. Processing Time

- El tiempo de procesamiento se refiere al tiempo del sistema de la máquina que está ejecutando la operación respectiva.
- Todas las operaciones basadas en el tiempo (como las ventanas de tiempo) utilizarán el reloj del sistema de las máquinas que ejecutan el operador respectivo.
- Es la **noción más simple de tiempo y no requiere coordinación entre flujos y máquinas**. Por lo tanto, **ofrece el mejor rendimiento y la menor latencia**.
- En entornos distribuidos y asíncronos, **no proporciona determinismo**.
Es susceptible a la velocidad con la que los registros llegan al sistema (por ejemplo, desde la cola de mensajes) y a la velocidad con la que los registros fluyen entre los operadores dentro del sistema.

2. Conceptos. Ingestion Time

- El tiempo de ingestión es el momento en el que los eventos entran en el sistema de procesamiento de Streams.
- En el operador fuente, **a cada registro se le asigna la hora actual de la fuente como una marca de tiempo.**
- El tiempo de ingestión se encuentra conceptualmente entre el tiempo del evento y el tiempo de procesamiento.
- En comparación con el tiempo de procesamiento, es más costoso, pero proporciona resultados más predecibles.

2. Conceptos. Ingestion Time

- En comparación con el tiempo de procesamiento, es más costoso, pero proporciona resultados más predecibles.
 - El tiempo de ingestión utiliza marcas de tiempo estables (asignadas una vez en la fuente), por lo que diferentes operaciones de ventana sobre los registros se referirán a la misma marca de tiempo, mientras que en el tiempo de procesamiento, cada operador de ventana puede asignar el registro a una ventana diferente (basado en el reloj del sistema local y cualquier retraso en el transporte).
- En comparación con el tiempo del evento, es menos costoso, pero proporciona resultados menos deterministas.
 - El tiempo de ingestión determina el orden de los eventos y no permite eventos tardíos, mientras que en el tiempo del evento, los eventos pueden llegar fuera de orden.

2. Ejercicio 1. Windows type

Imagina que trabajas en una empresa de comercio electrónico y tienes un sitio web donde los clientes compran tus productos.

Necesitas hacer lo siguiente:

- Cada 4 horas, quieres agrupar todos los productos comprados para que sean enviados desde el almacén logístico principal al canal de distribución.
- Para poder identificar tus campañas publicitarias, deseas ver en cualquier momento durante los últimos 5 minutos el número de productos comprados de cada tipo.

¿Cómo lo implementarías?

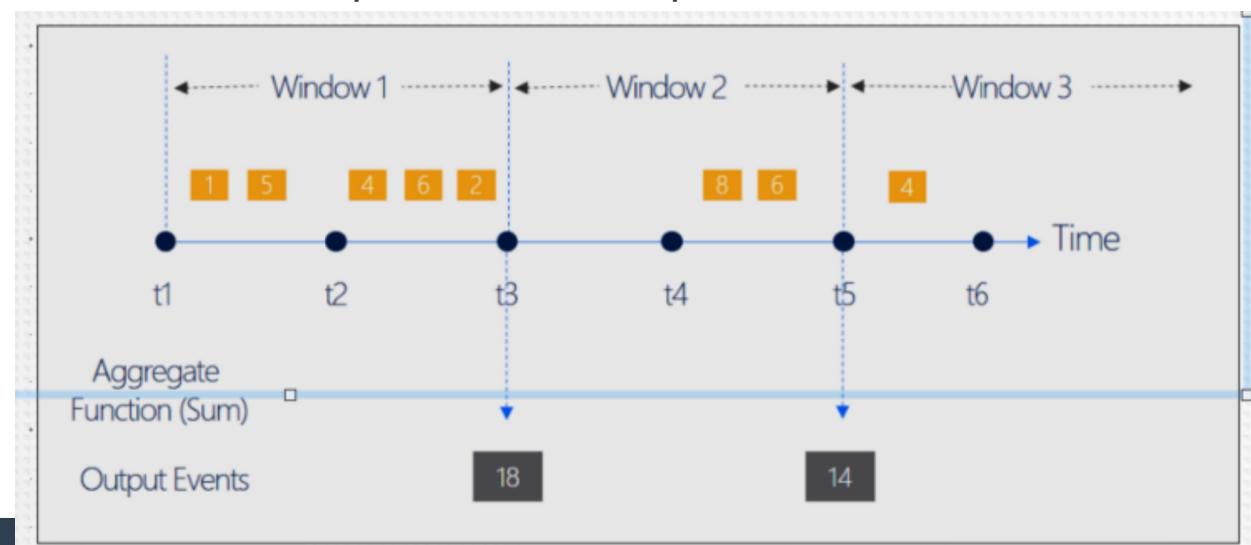
2. Conceptos. Windowing

El procesamiento de Streams hace posible realizar cálculos en tiempo real. Sin embargo, los cálculos que requieren agregaciones necesitan dividir el Stream en grupos limitados de eventos.
¡De lo contrario, se requeriría procesar todo el flujo de datos no limitado!

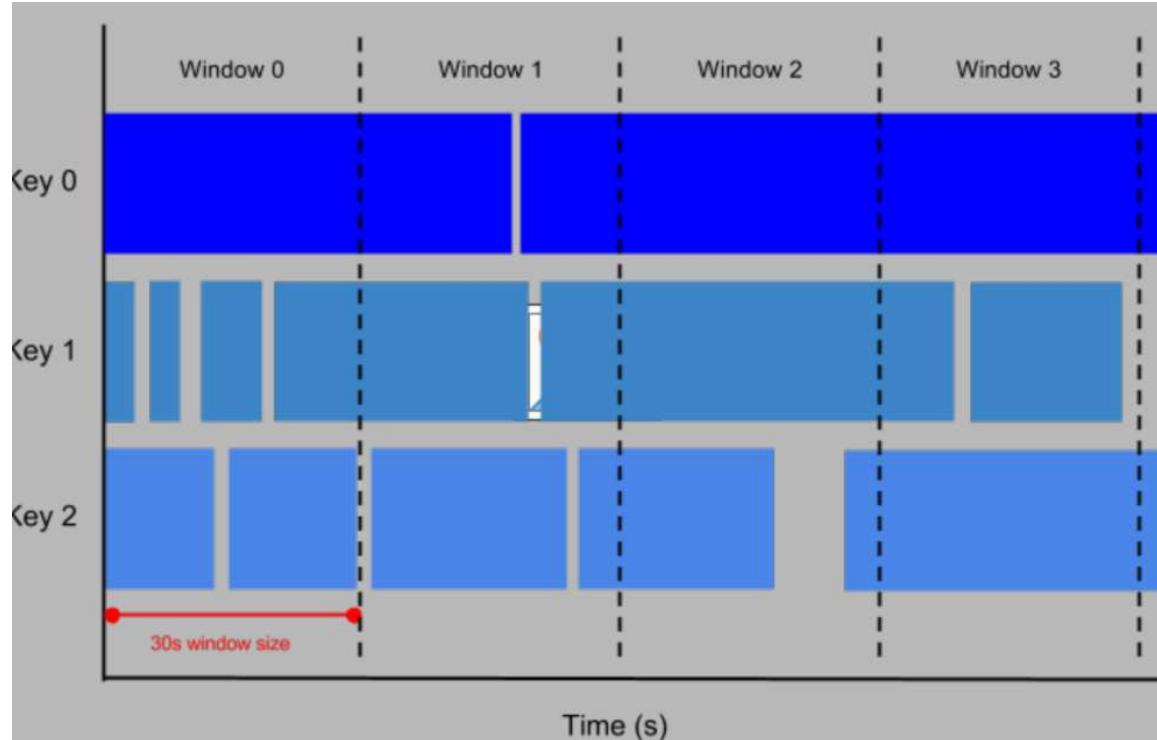
Windowing es la capacidad de realizar un conjunto de cálculos (agregación) u otra operación sobre subconjuntos de eventos que caen dentro de un período de tiempo determinado.

Tipos de ventanas:

- Fija o Tumbling
- Deslizante (Sliding)
- De sesión (Session)



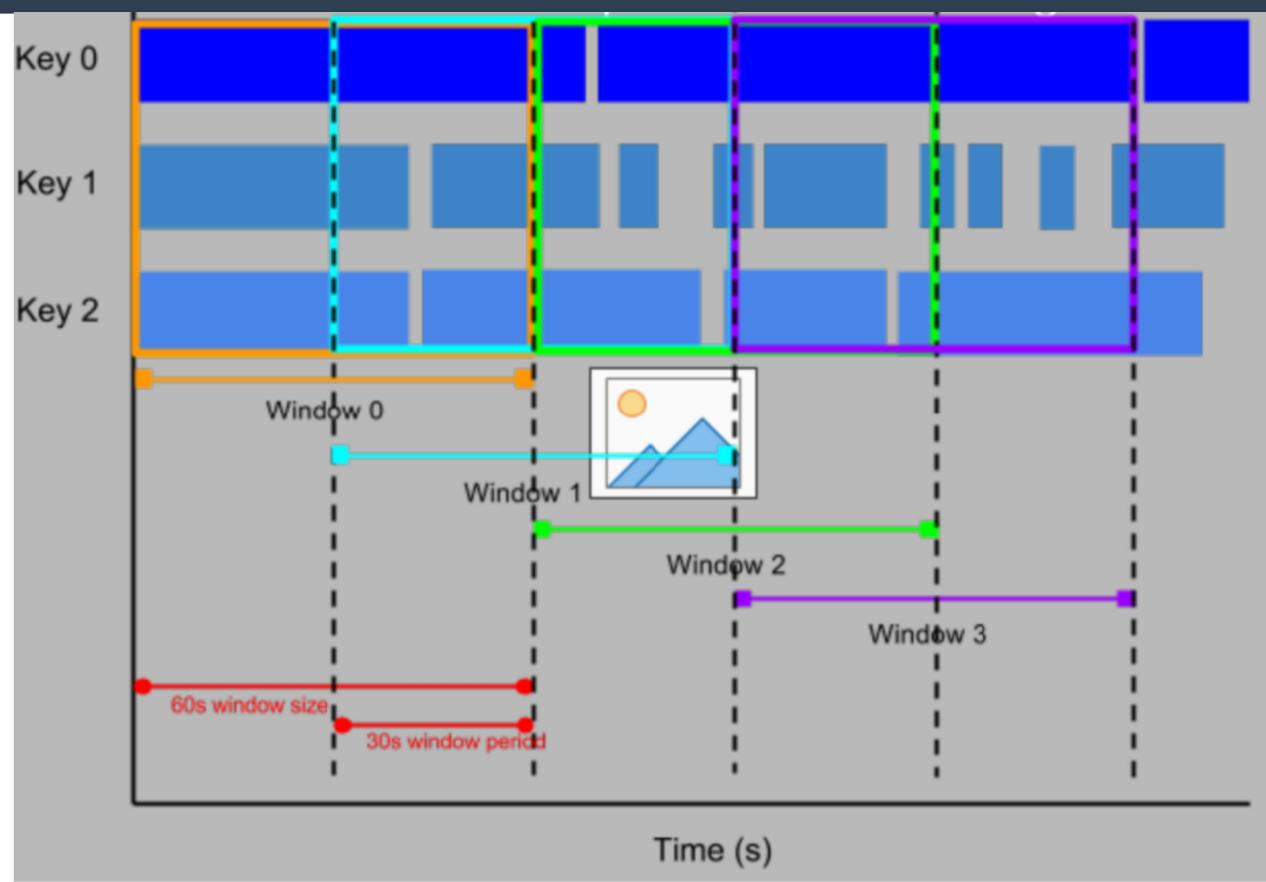
2. Conceptos. Windowing. Tumbling Window



Periódica y sin superposición

Cada elemento pertenece a exactamente una ventana.

2. Conceptos. Windowing. Sliding Window

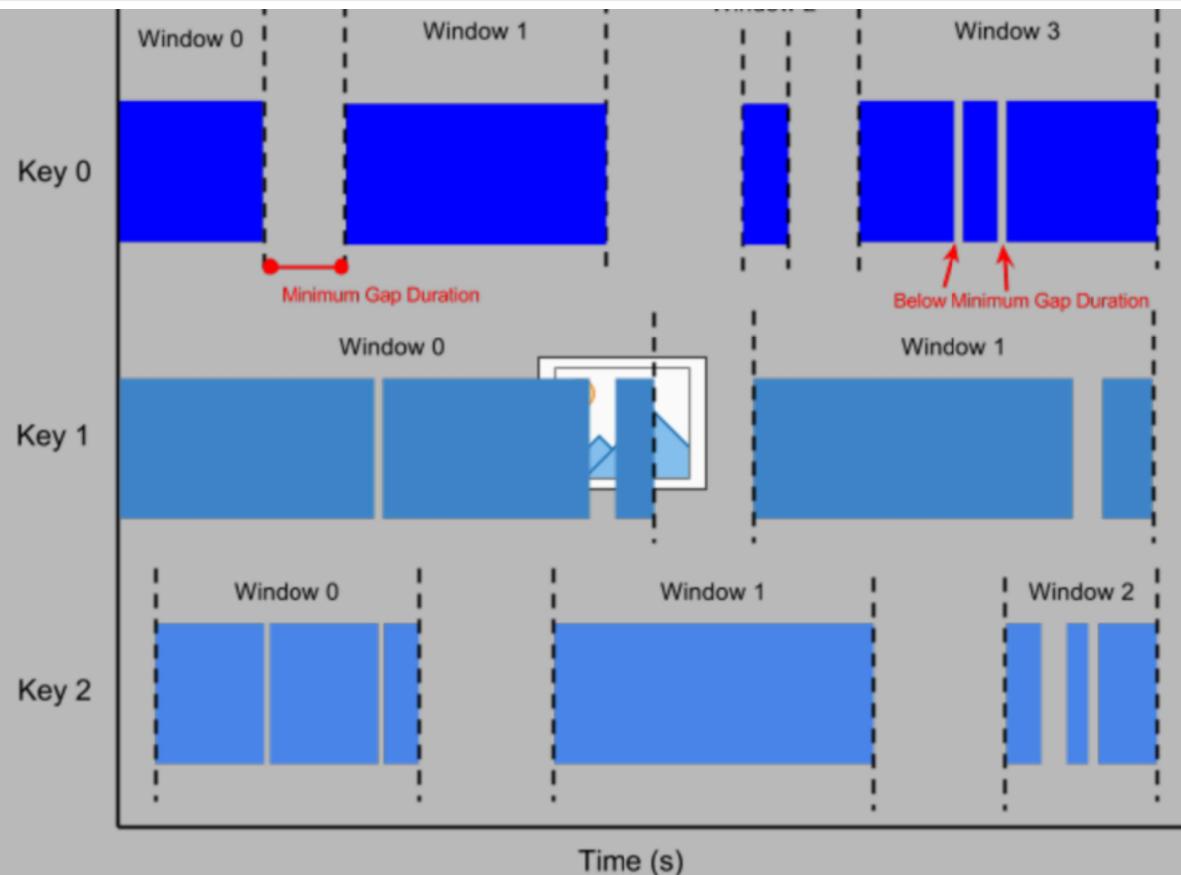


Periódica y con superposición

Cada elemento puede pertenecer a varias ventanas.

La ventana está definida por el período y el tamaño.

2. Conceptos. Windowing. Sesión Window



Dinámica y sin superposición
Solo para flujos clave-valor.
Las ventanas tienen tamaños variados y se definen en función de los datos.
Típicamente, las ventanas se enfocan en áreas de concentración: duración mínima de intervalo.
Útil para datos que están distribuidos de manera irregular con respecto al tiempo.

2. Conceptos. Windowing Challenges

Las semánticas poderosas rara vez vienen sin coste, y las ventanas de tiempo de evento no son una excepción.

Almacenamiento en buffer

- Las vidas útiles extendidas de las ventanas requieren el almacenamiento en buffer de datos (estado) → **Estado**
- ¿Cómo limitamos el tamaño de este estado? → **Marcas de agua (Watermarks)**

Compleitud. ¿Cómo sabemos cuándo los resultados de la ventana están listos para materializarse?

- Heurística → Retraso permitido
- Además, darle responsabilidad al creador del pipeline para decidir:
 1. Cuándo quieren que los resultados de las ventanas se materialicen → **Disparador (Trigger)**
 2. Cómo deben refinarse esos resultados a lo largo del tiempo (múltiples ejecuciones) → **Modo de acumulación (Accumulation mode)**

2. Conceptos. Watermarks

Watermarks es esencialmente una marca de tiempo basada en el tiempo del evento.

- Es un umbral móvil que ayuda a entender qué tan lejos estamos del tiempo del evento.
- ¿Cuándo están listos los resultados de la ventana para materializarse?
- ¿Cuándo han llegado ya todos los eventos para una ventana?

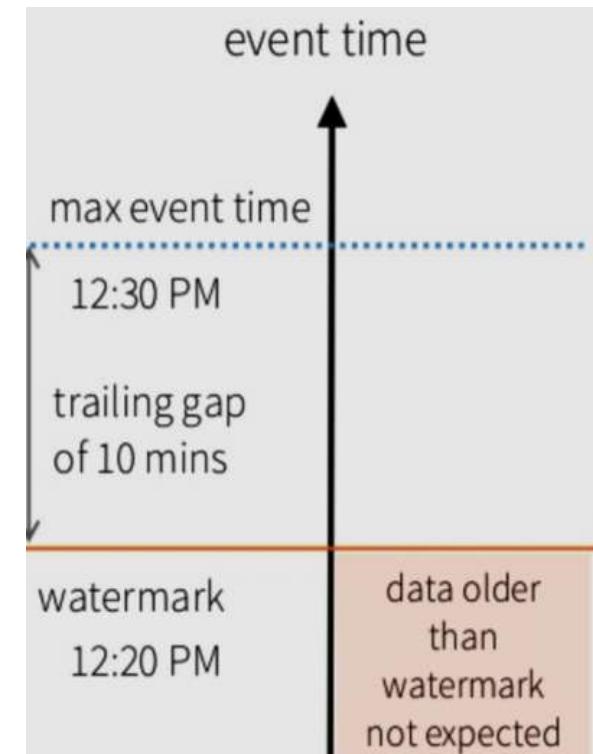
Watermarking para limitar el estado mientras se manejan datos tardíos

- Eliminar el almacenamiento de estado "antiguo".

Un mecanismo para el "retraso permitido"

Aplica para **ventanas definidas sobre el tiempo del evento**.

- De lo contrario (tiempo de procesamiento, tiempo de ingestión), el retraso no tiene sentido.



2. Conceptos. Watermarks

Watermark con un valor de tiempo X hace la siguiente afirmación: "todos los datos de entrada con tiempos de evento menores que X han sido observados."

A medida que llegan los eventos, watermark se mueve detrás de ellos.

Depende de la fuente de datos → se establece en el lector.

```
eventsDF = ...  
  
windowedCountsDF = eventsDF  
    .withWatermark("eventTime", "10 minutes")  
    .groupBy("deviceId", window("eventTime", "10 min", "5 min"))  
    .count()
```

The diagram shows a code snippet for creating a DataFrame with watermarking and grouping by window. Three annotations are present: 'Watermark' points to the 'withWatermark' method call, 'Window Size' points to the '10 min' argument in the 'window' method, and 'Window Period' points to the '5 min' argument in the 'window' method.

2. Conceptos. Watermarks

Retraso permitido para evitar perder eventos tardíos.

Demasiado poco

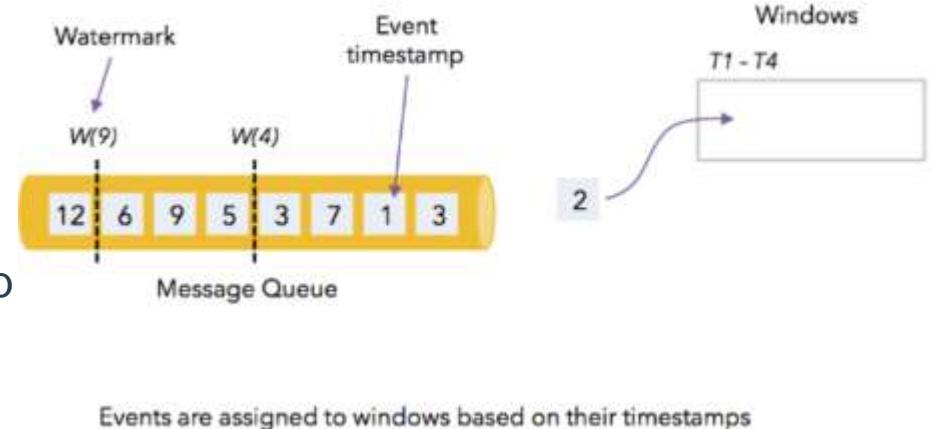
Aumenta los datos tardíos.

Demasiado

Aumenta la latencia y los requisitos de almacenamiento en buffer.

Mitigación con resultados especulativos → ejecuciones tempranas

→ ¡múltiples resultados para cada ventana!



Fuente: <https://data-artisans.com/blog/how-apache-flink-enables-new-streaming-applications-part-1>

2. Conceptos. Late Events

Ciertos elementos pueden violar la condición watermark.

Después de que ocurra la $\text{Watermark}(t)$, más elementos con marca de tiempo $t' \leq t$ pueden ocurrir.

Estrategias

- Los elementos tardíos se descartan cuando watermark ha pasado el final de la ventana.
- **Retraso máximo permitido** para los operadores de ventana.
 - Los elementos que llegan después de watermark haya pasado el final de la ventana, pero antes de que pase el final de la ventana más el retraso permitido, aún se agregan a la ventana.
 - Un elemento tardío pero no descartado puede hacer que la ventana se ejecute nuevamente.
 - Mantener el estado de las ventanas hasta que expire su retraso permitido.

2. Conceptos. Triggers

Determina cuándo una ventana está lista para ser procesada por la función de ventana.

Tipos de disparadores (triggers):

- Basados en tiempo
 - Tiempo del evento – por ejemplo, después watermark
 - Tiempo de procesamiento
- Basados en datos – por ejemplo, un número determinado de elementos de datos
- Disparadores compuestos

Estrategias:

- Disparar (Fire): activar el cálculo
- Eliminar (Purge): borrar los elementos en la ventana
- Disparar y eliminar (Fire & Purge).

Fuente: <https://ci.apache.org/projects/flink/flink-docs-release-1.3/dev/windows.html#triggers>

2. Conceptos. Accumulation

Para algunas configuraciones, los resultados de la ventana pueden ser calculados varias veces (múltiples disparadores).

- Ejecuciones tempranas → resultados especulativos
- A tiempo
- Ejecuciones tardías → debido a eventos tardíos

La acumulación determina cómo los resultados emitidos desde una ventana se relacionan con los resultados previamente emitidos para la misma ventana.

¿Cómo manejar los nuevos valores?

- **Descartar** el valor antiguo, usar solo el nuevo resultado de la ventana.
- **Acumular** el nuevo valor con el valor antiguo (por ejemplo, sumar los resultados a un contador).
- **Retraer y acumular:** especificar el valor antiguo a retraer, reemplazarlo con el nuevo valor.

2. Conceptos. Accumulation

First trigger firing: $[A \rightarrow 5, B \rightarrow 8]$

Second trigger firing: $[C \rightarrow 15, B \rightarrow 10]$

Third trigger firing: $[A \rightarrow 2]$

First trigger firing: $[A \rightarrow 5, B \rightarrow 8]$

Second trigger firing: $[A \rightarrow 5, B \rightarrow 18, C \rightarrow 15]$

Third trigger firing: $[A \rightarrow 7, B \rightarrow 18, C \rightarrow 15]$

First trigger firing: $[A \rightarrow 5, B \rightarrow 8]$

Second trigger firing: $[A \rightarrow -5, B \rightarrow -8, A \rightarrow 5, B \rightarrow 18, C \rightarrow 15]$

Third trigger firing: $[A \rightarrow -5, B \rightarrow -18, C \rightarrow -15, A \rightarrow 7, B \rightarrow 18, C \rightarrow 15]$

- **Discarding Mode**

- **Accumulating Mode**

- **Retract & accumulate**

3. Tecnologías

Mature Frameworks



Spark Streaming



Apache Flink



Apache Beam



Kafka Streams

New players



Hazelcast Jet



Apache Pulsar Functions



Redis Streams

3. Tecnologías

Monitoring and Control



Infrastructure monitoring



Application monitoring



Logging aggregation

Ingestion



Data Ingestion Services

Stream Processing



Event Notification Services

Event Processing
Windowing
Checkpointing



Business Rules



External Data Access

Microservices

Analysis



Data Analysis

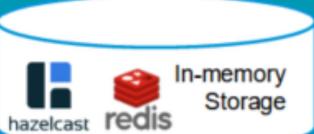


Data Visualization

Data Access Layer



Data Access Services



In-memory Storage



Historic Storage



Data Support Services



Administration



Downstream Delivery



Data as a Service

Infrastructure



DevOps Services



Infrastructure Abstraction Services



Infrastructure Services

