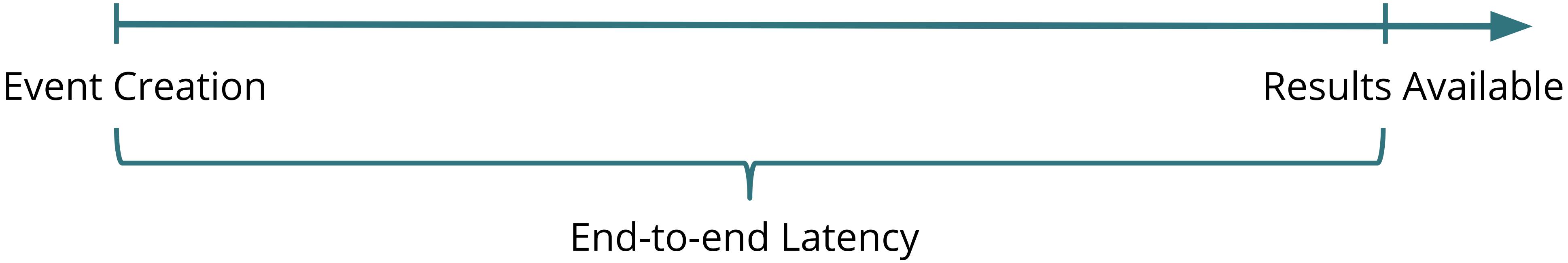
Latency

Konstantin Knauf, Solutions Architect

© 2019 Ververica

Defining Latency



- **Latency = Processing Time (of Stage)** - **Event Time (= Event-Time Lag)**
- Meaning of “Event Time” depends on logic of the application
  - timestamp of the event
  - end time of an event time-window
  - ...

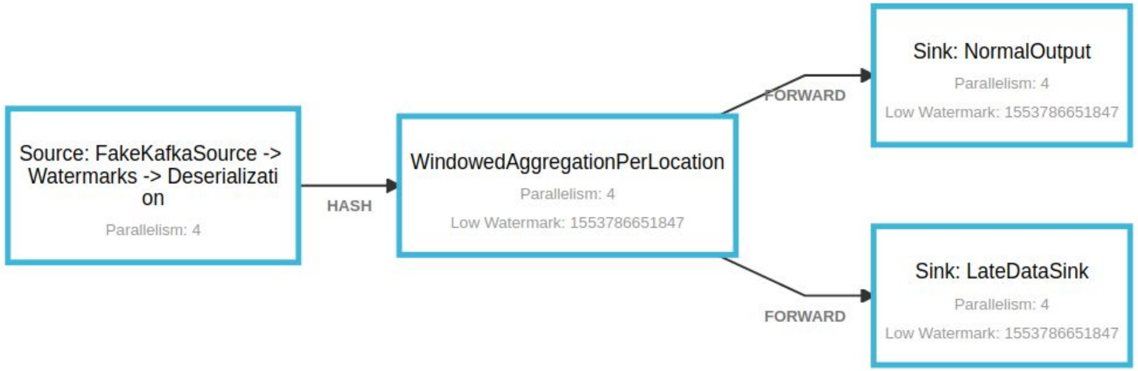
Reasons for Latency

© 2019 Ververica

Reasons for Latency

Latency: Processing Time (Publish) - Event Time

Running Example


Source: FakeKafkaSource -> Watermarks -> Deserialization
Parallelism: 4
HASH
WindowedAggregationPerLocation
Parallelism: 4
Low Watermark: 1553786651847
FORWARD
Sink: NormalOutput
Parallelism: 4
Low Watermark: 1553786651847
FORWARD
Sink: LateDataSink
Parallelism: 4
Low Watermark: 1553786651847

- Kafka Consumer
- Keyed Event Time Window
- *Exactly-Once* Kafka Producer

© 2019 Ververica

Reasons for Latency

Latency: Processing Time (Publish) - Event Time

- Latency accumulated **before Flink application**
    - time between event creation and storage of event in queue
    - time between storage in queue and consumption by application

© 2019 Ververica

Reasons for Latency

Latency: Processing Time (Publish) - Event Time

- Latency accumulated **inside Flink application**
  - latency due to event time processing
  - latency due to network (incl. network buffers)
  - latency due to processing delays
  - latency due to transactional sinks
  - latency due to checkpointing

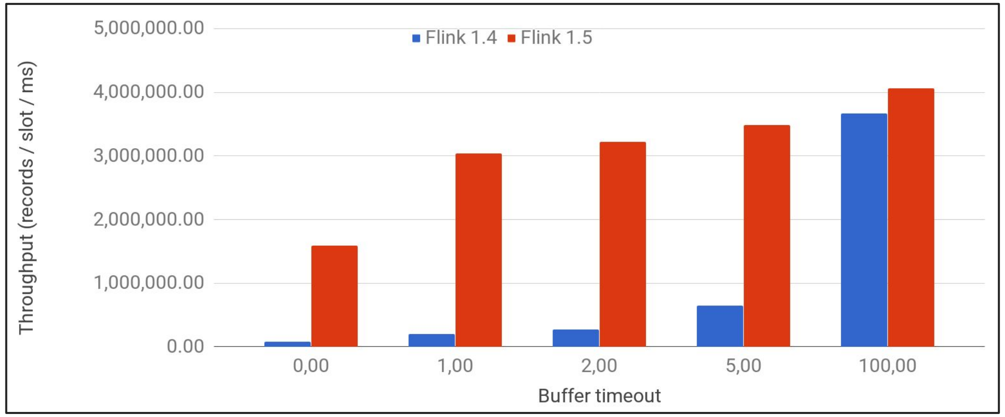
© 2019 Ververica

Latency due to Event Time Processing

- Applies to anything based on (event time) timers:
  - windows
  - process functions
- Watermark progresses with `min(all input watermarks)`
  - influenced by any upstream operator
  - allowed out-of-orderness adds to latency
  - watermarking interval matters
- Window / timer fires when watermark exceeds window end / timer

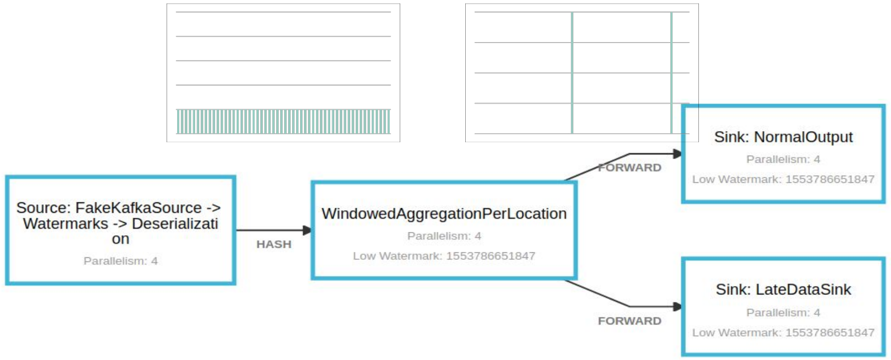
Latency due to Network Delays

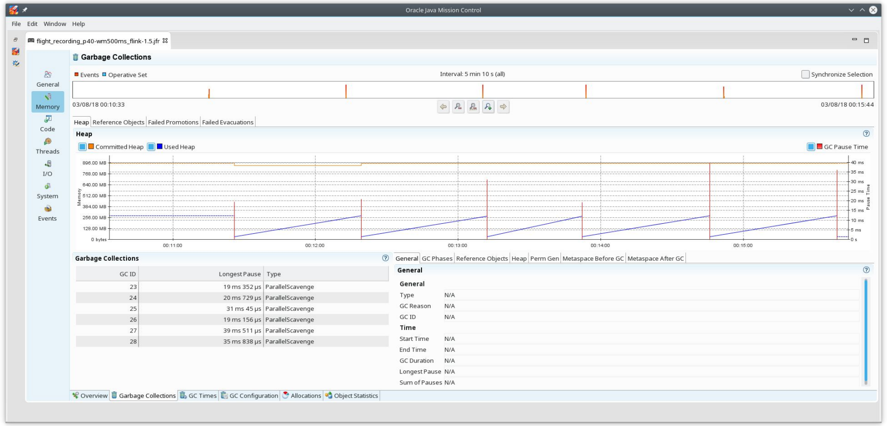
- Every transfer over the network (repartition/rebalance) adds latency
- Flink assembles (serialized) records in buffers for network/local transfer
- Buffers are sent once full or after buffer timeout
- Trade-off between throughput & latency
- `StreamExecutionEnvironment#setBufferTimeout(int)`



Latency due to Processing Delays

- Execution of user & framework code adds latency
- try to mitigate load spikes due to windowing, e.g. by pre-aggregating as much as possible
- garbage collections will cause temporary backpressure & small latency spikes


Source: FakeKafkaSource -> Watermarks -> Deserialization
Parallelism: 4
HASH
WindowedAggregationPerLocation
Parallelism: 4
Low Watermark: 1553786651847
FORWARD
Sink: NormalOutput
Parallelism: 4
Low Watermark: 1553786651847
FORWARD
Sink: LateDataSink
Parallelism: 4
Low Watermark: 1553786651847



Garbage Collections

GC ID	Longest Pause	Type
23	19 ms 352 μs	ParallelScavenge
24	20 ms 729 μs	ParallelScavenge
25	31 ms 45 μs	ParallelScavenge
26	19 ms 156 μs	ParallelScavenge
27	39 ms 511 μs	ParallelScavenge
28	35 ms 838 μs	ParallelScavenge

General

General

Type	N/A
GC Reason	N/A
GC ID	N/A

Time

Start Time	N/A
End Time	N/A
GC Duration	N/A
Longest Pause	N/A
Sum of Pauses	N/A

Latency due Transactional Sinks

- Lifecycle of a typical Transactional Sink

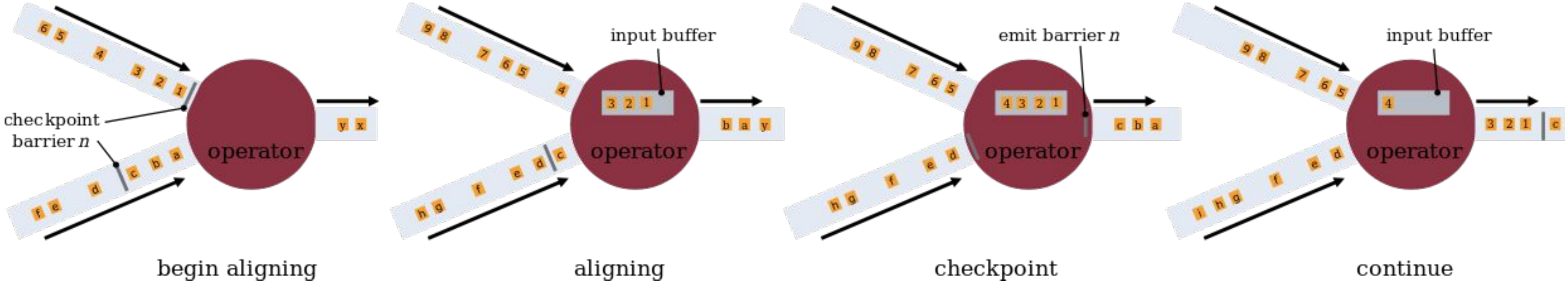
Phase	**Actions**
onElement	● asynchronously sent to sink system
onSnapshot	● flush all records & wait for acknowledgement ● create new transaction for next checkpointing epoch ● store transaction metadata in Flink state
onCheckpointComplete	● commit pending transactions & publish data

- Transactional sink adds latency up to the checkpoint interval

© 2019 Ververica

Latency due to Checkpointing

- Checkpointing consists of three phases
  - Checkpoint Alignment (synchronous)
  - Synchronous Part (synchronous)
  - Asynchronous Part
- Checkpoint Alignment



begin aligning   aligning   checkpoint   continue

- backpressure on blocked channels

© 2019 Ververica

Exercises

Troubleshooting Watermarks & Latency Tuning

Note: If you have not completed the previous exercise, please check out `ffsf-19-solution-1` to proceed.

Exercise 2
After the first exercise the job running stable, but there is no output. Investigate the issue and fix it.

Note: If you have not completed the previous exercise, please check out `ffsf-19-solution-2` to proceed.

Exercise 3
Reduce the 99th percentile of the event time lag of the `WindowedAggregationPerLocation` operator. The `eventTimeLag` metric will show the current value.

© 2019 Ververica

ververica

konstantin@ververica.com www.ververica.com @VervericaData