Configuring Processing Models

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Stream processing models

Micro-batch execution and continuous processing

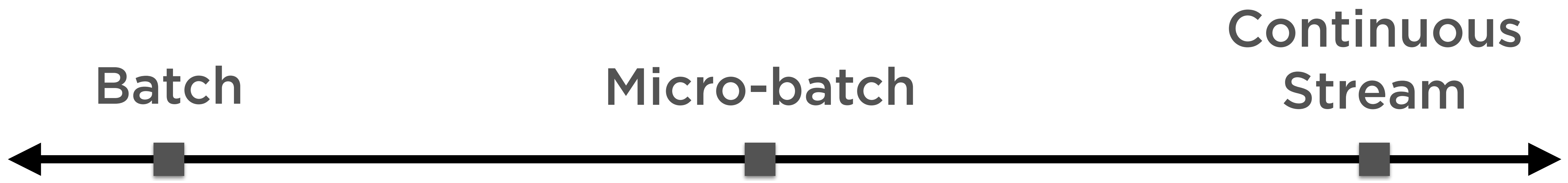
Considerations of latency, scaling, and recovery

At-least-once guarantees

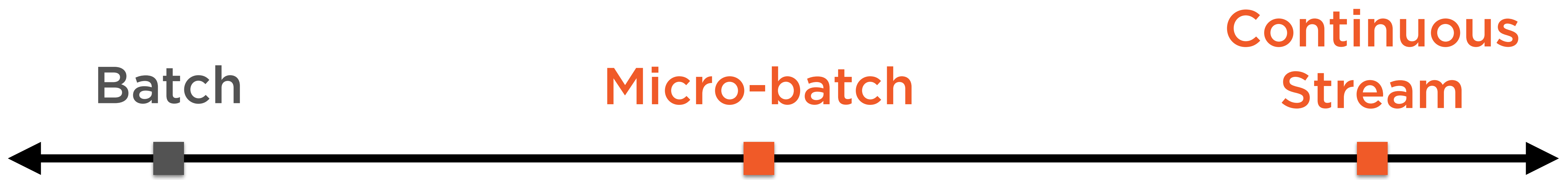
Running Spark in continuous processing mode

Stream Processing Models

Stream Processing Models



Stream Processing Models



Stream processing does not necessarily mean continuous real-time processing

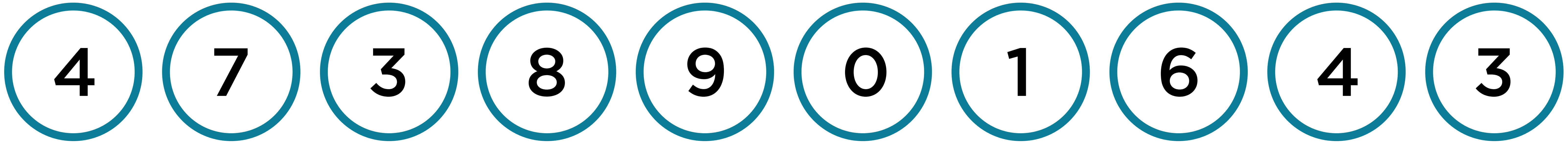
Micro-batch Processing

Run transformations on smaller accumulations of data

Collect say less than one minute of data

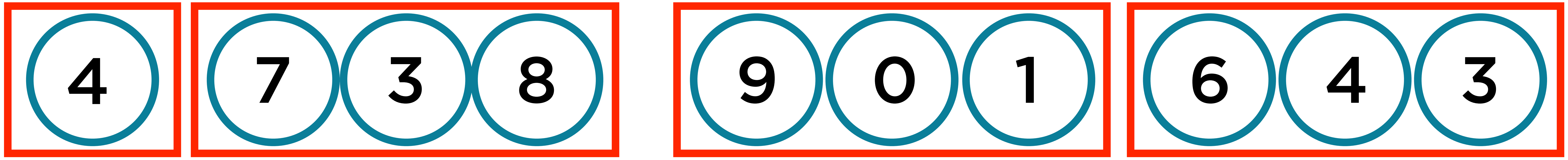
Process this micro-batch in near real-time

Micro-batch Processing



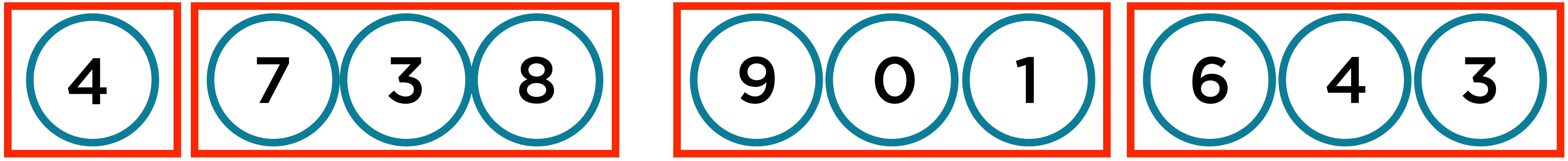
A stream of integers

Micro-batch Processing



Grouped into batches

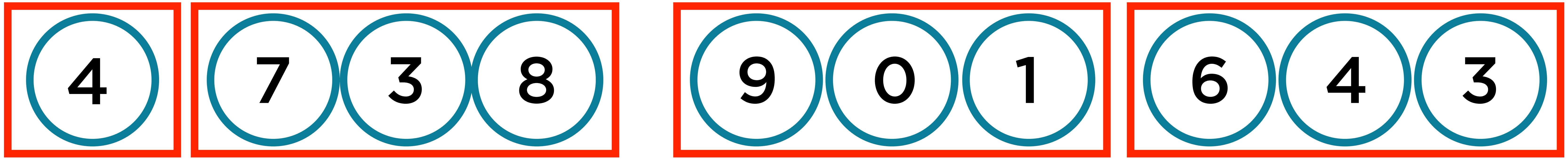
Micro-batch Processing



If the batches are small enough...

Close to real-time processing

Micro-batch Processing



Exactly once semantics

Replay micro-batches

Latency-throughput trade-off based on batch sizes

Batch Processing for Streams

Latency, freshness of data are not considerations

Complex analytical operations

Joins on relational data

- Data might be in a data warehouse, need not be in an RDBMS

Continuous Stream Processing for Streams

Latency and freshness of data are most important considerations

Rate of arrival is high

- Latency in seconds/milliseconds only possible with continuous processing

Micro-batch Processing for Streams

Latency and freshness of data are important

but

Real-time processing is overkill

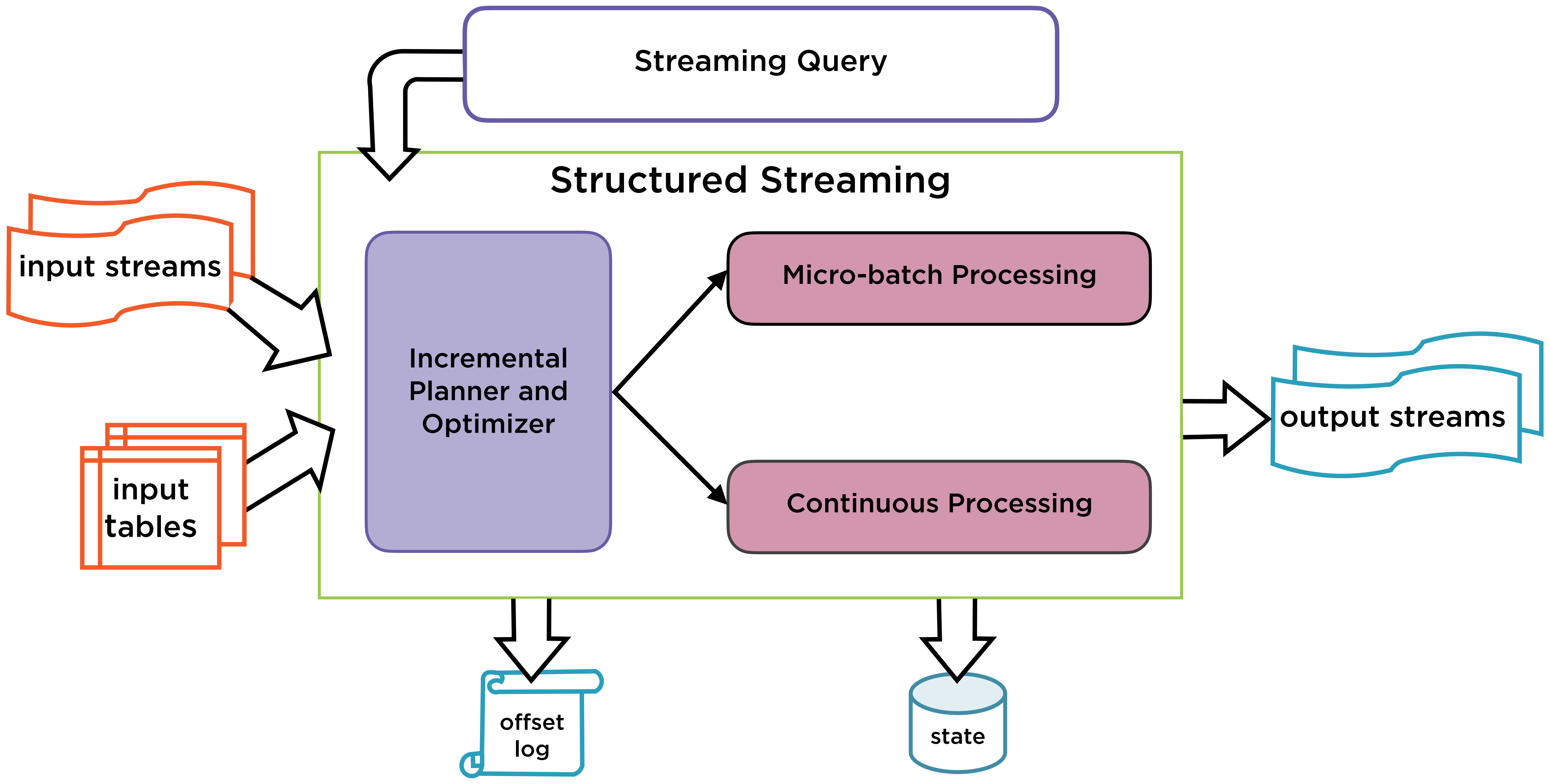
Rate of arrival is low/moderate

- Latency in seconds/milliseconds less important
- Acceptable latency possible with micro-batches

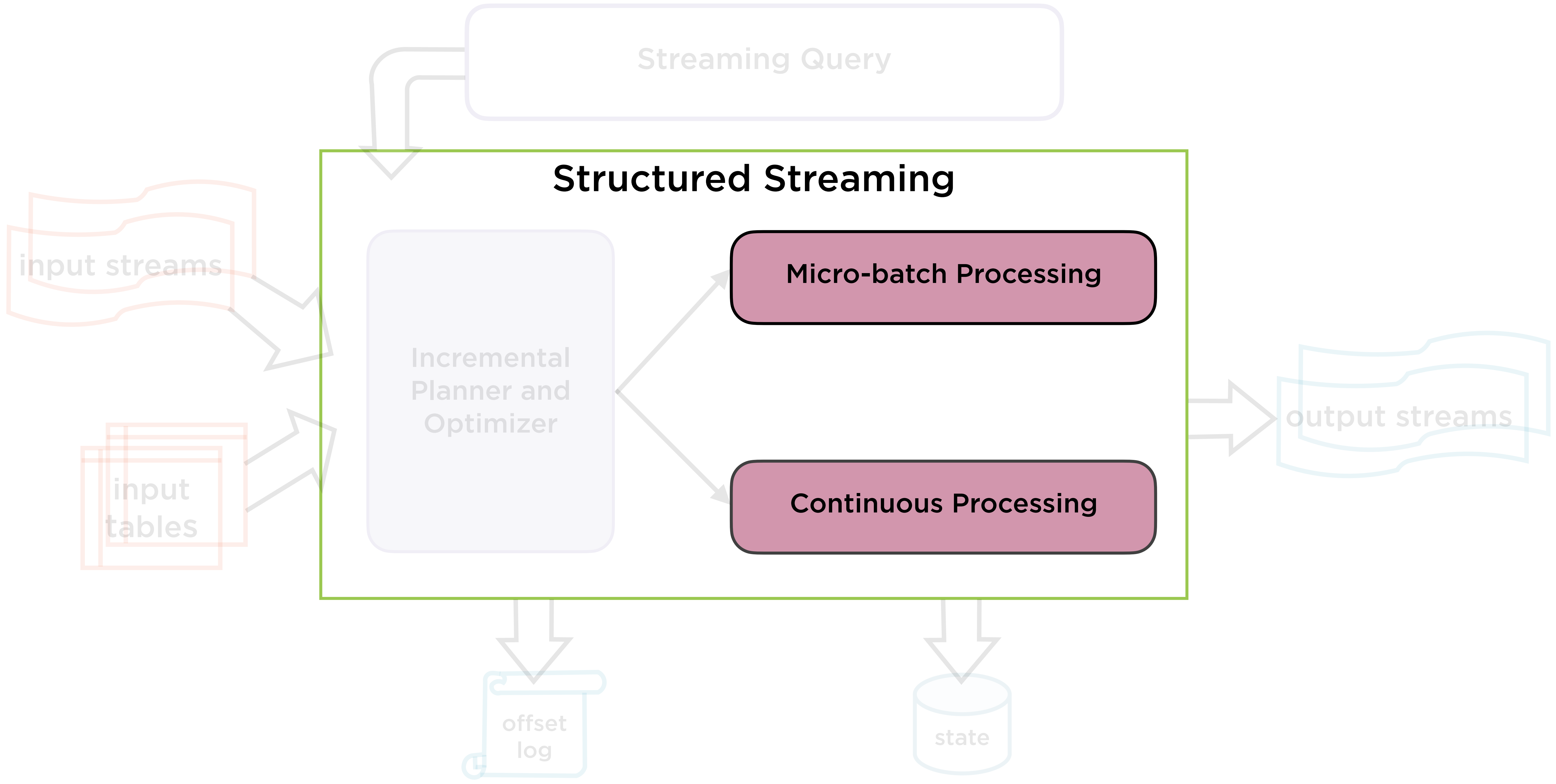
Micro-batch processing is the **default** mode in Spark

Spark also supports a new **experimental, continuous processing** mode

Micro-batch and Continuous Processing



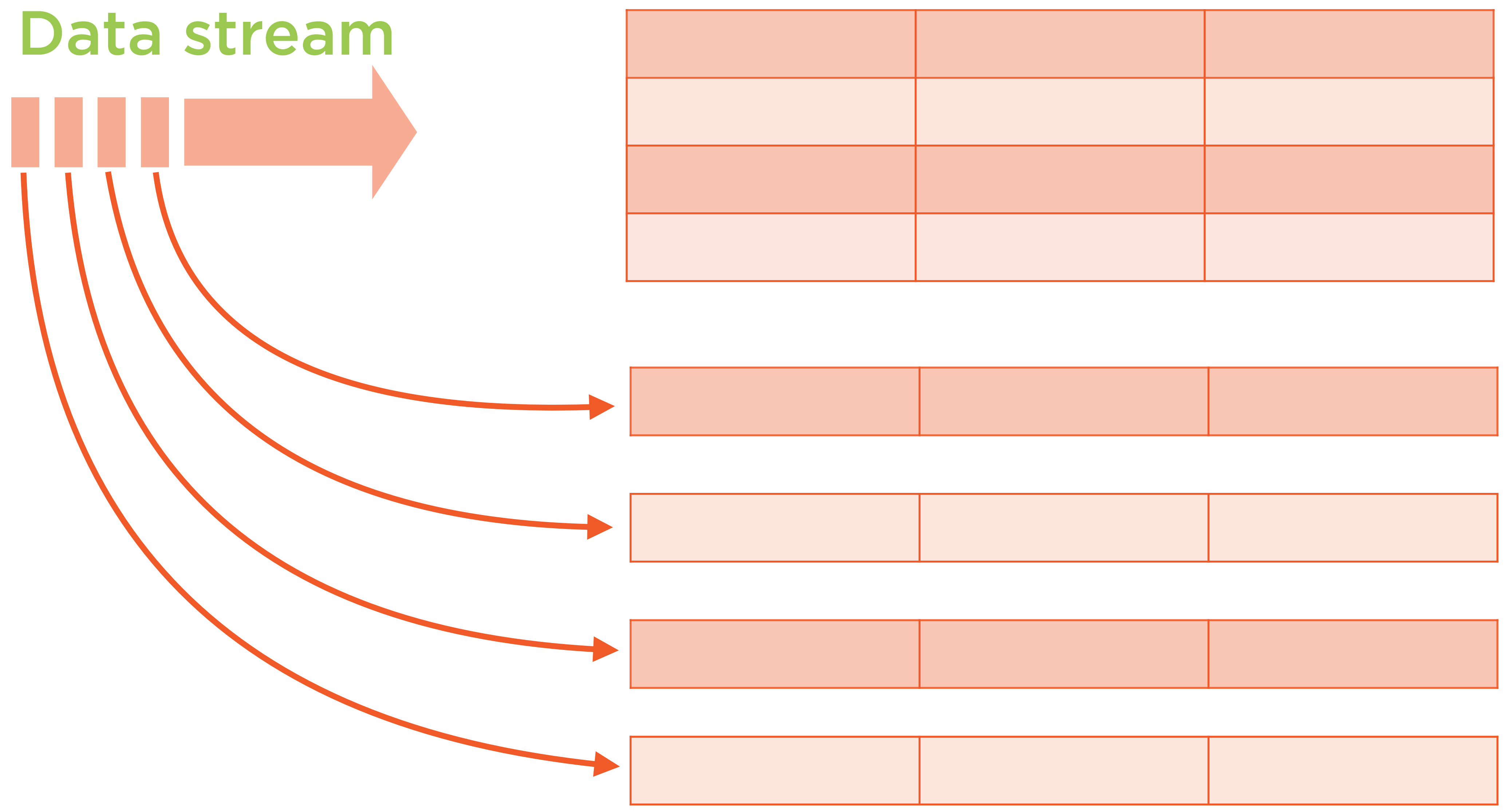
Micro-batch and Continuous Processing



Micro-batch Processing in Spark

Structured Streaming treats a live data stream as a table that is being **continuously** appended

Streaming Data Spark 2.x



Every data item that is arriving on the stream is like a new row being appended to the input table

Data stream as an unbounded input table

Micro-batch Processing in Spark

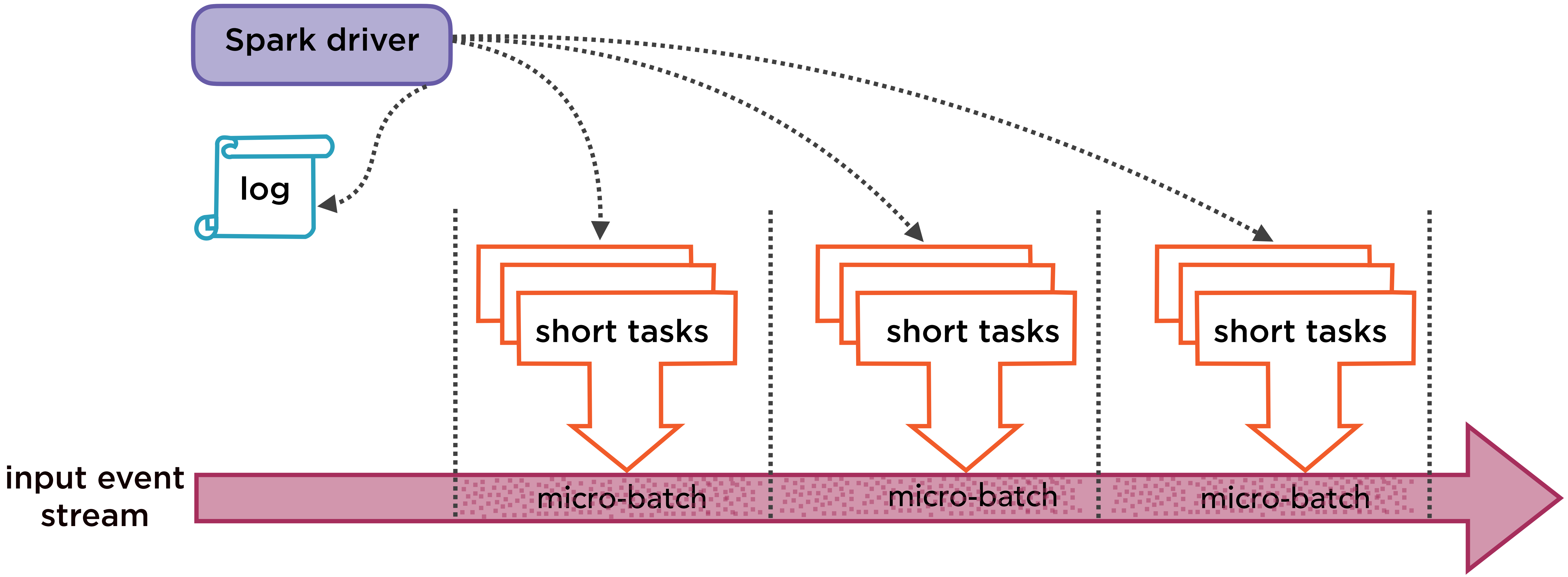
Default stream processing mode in Spark

Data streams processed as a series of batch jobs

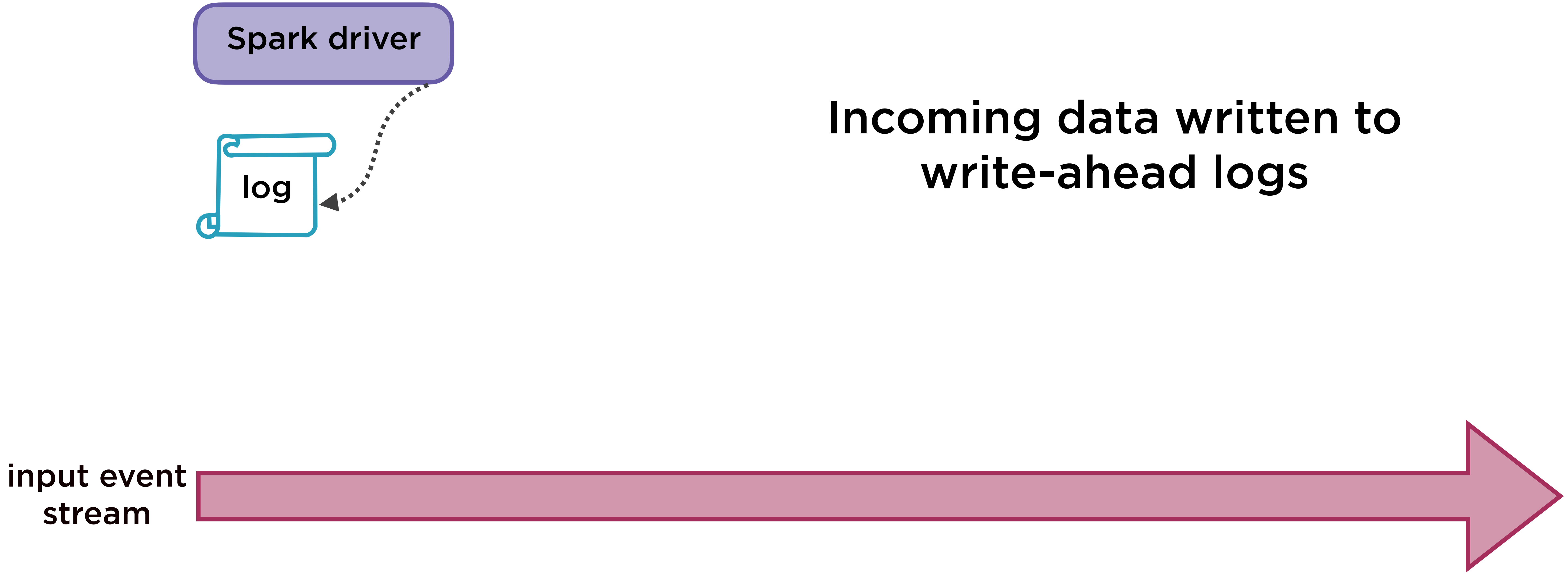
End-to-end latencies as low as 100ms

Exactly-once fault-tolerance guarantees

Micro-batch Processing in Spark

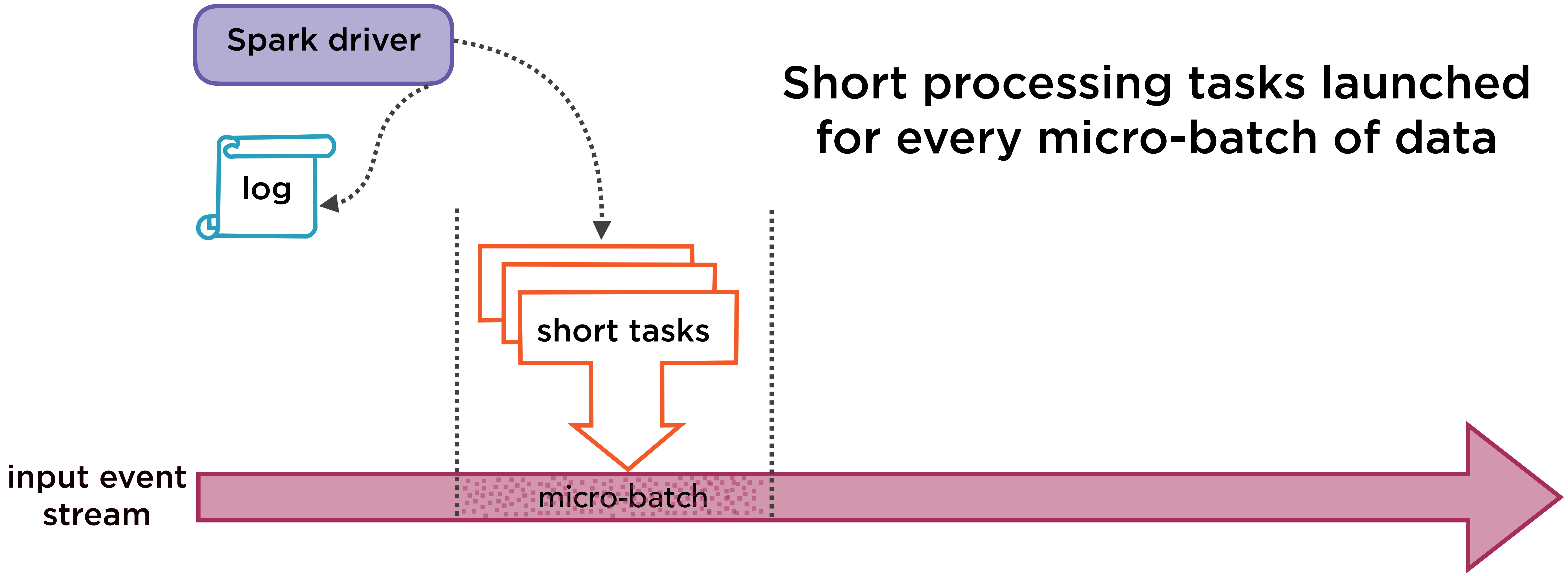


Micro-batch Processing in Spark



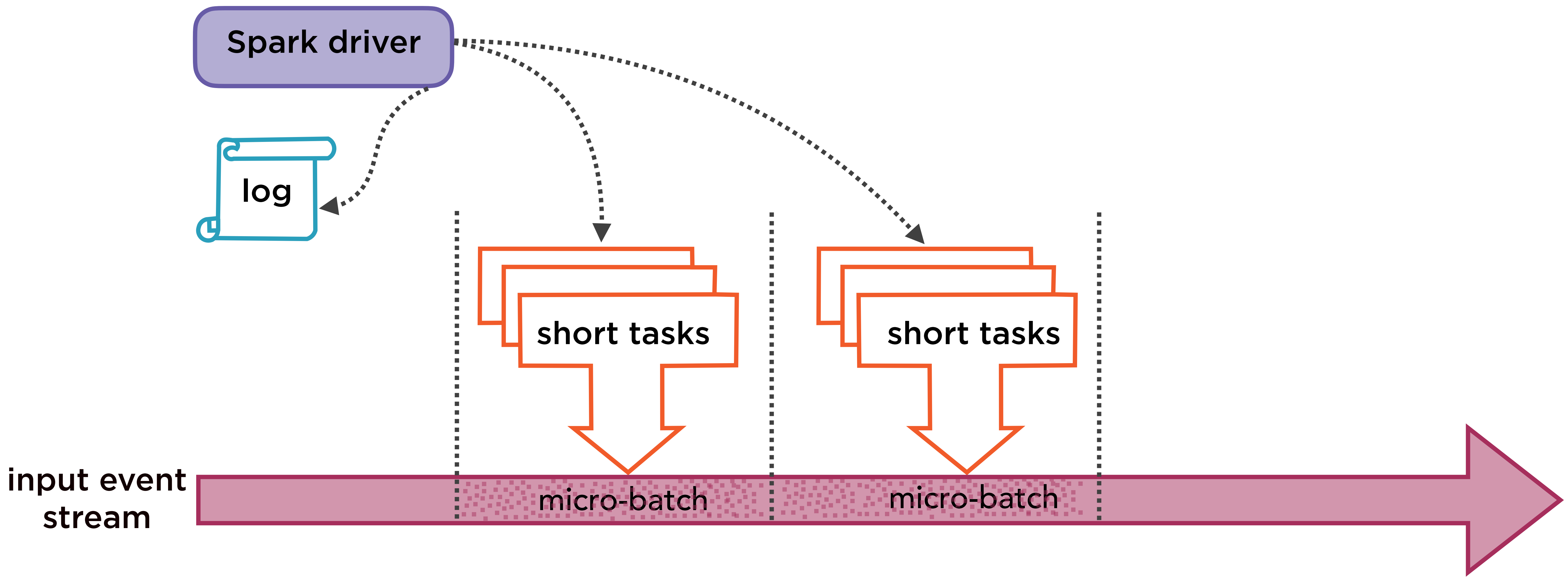
Incoming data written to write-ahead logs

Micro-batch Processing in Spark

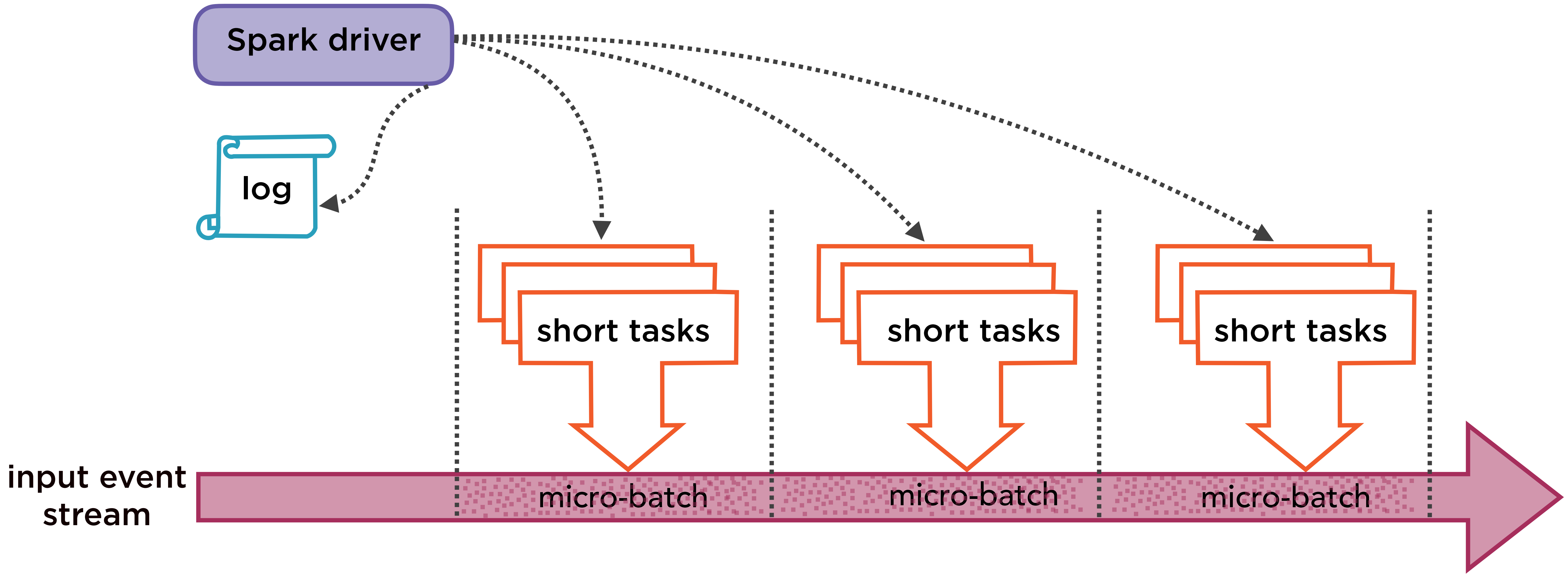


Short processing tasks launched for every micro-batch of data

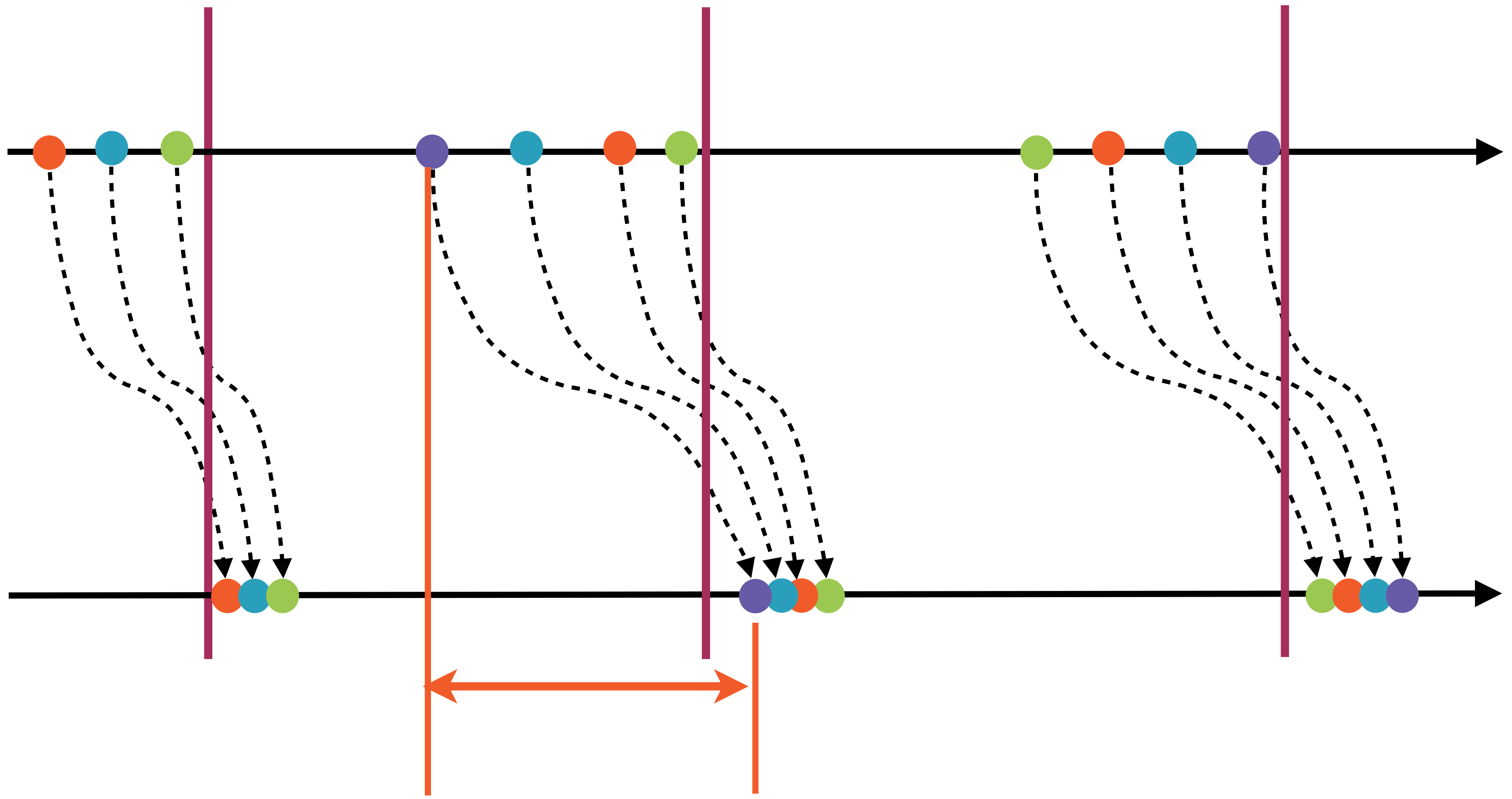
Micro-batch Processing in Spark



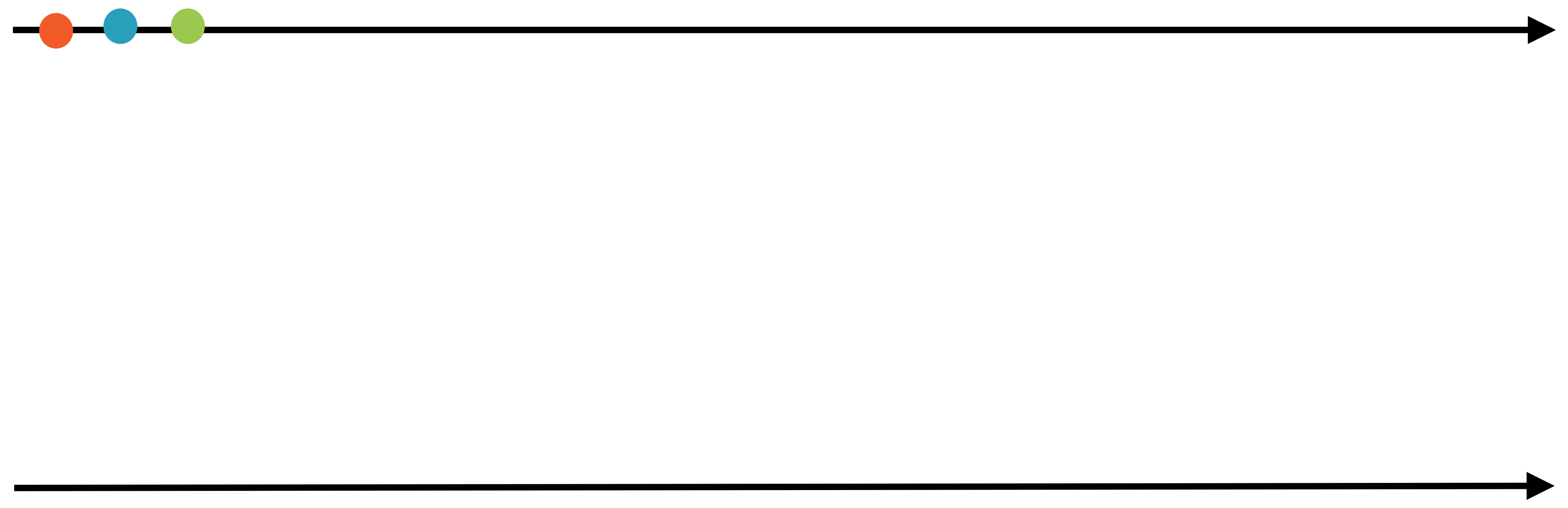
Micro-batch Processing in Spark



Higher End-to-end Latencies

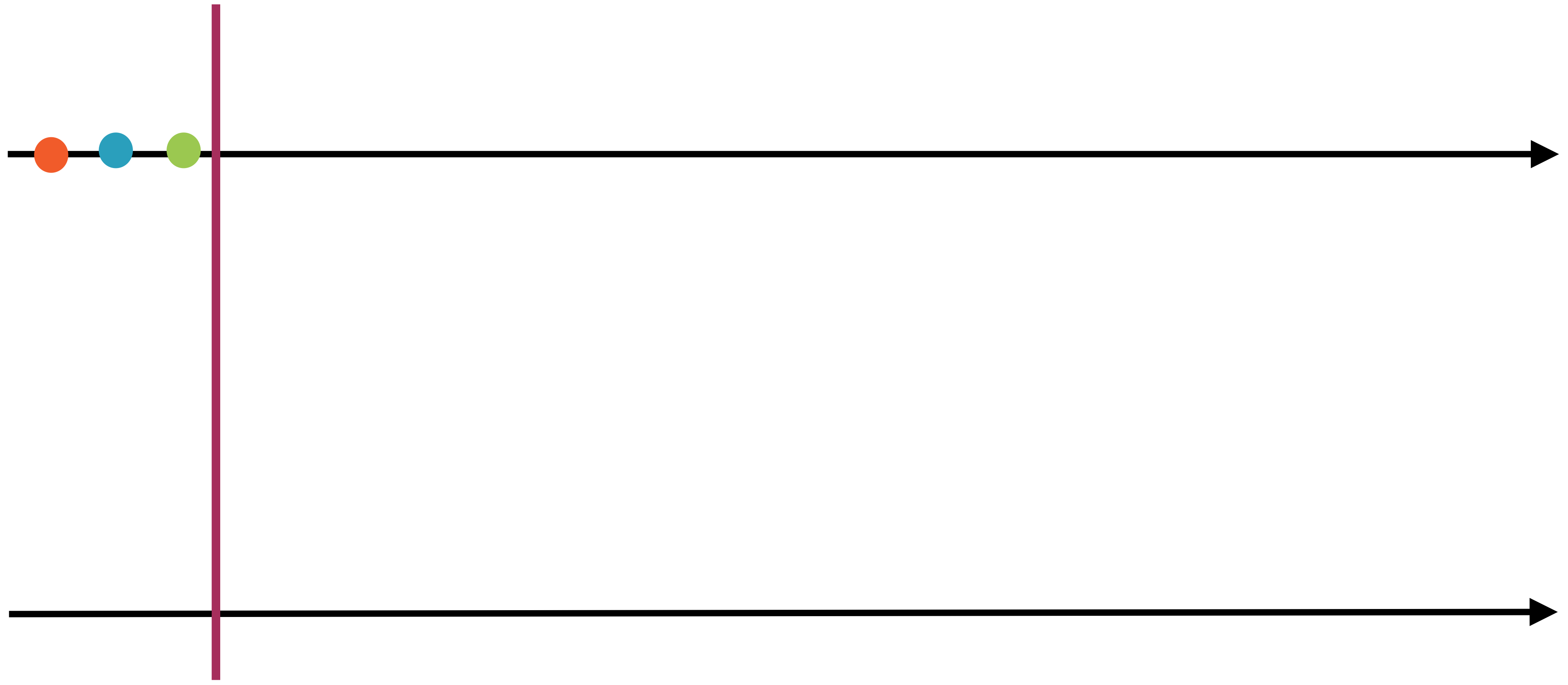


Higher End-to-end Latencies



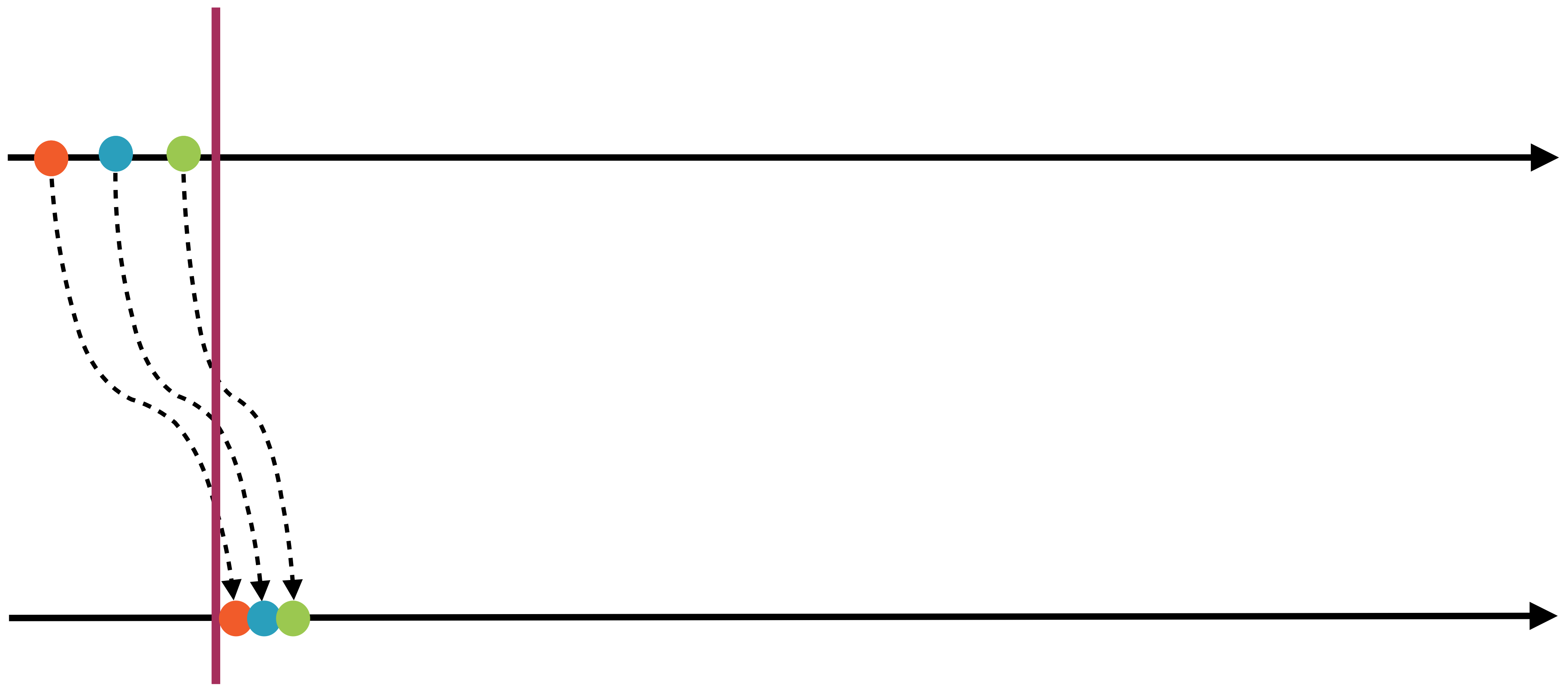
Incoming streaming data at the source

Higher End-to-end Latencies



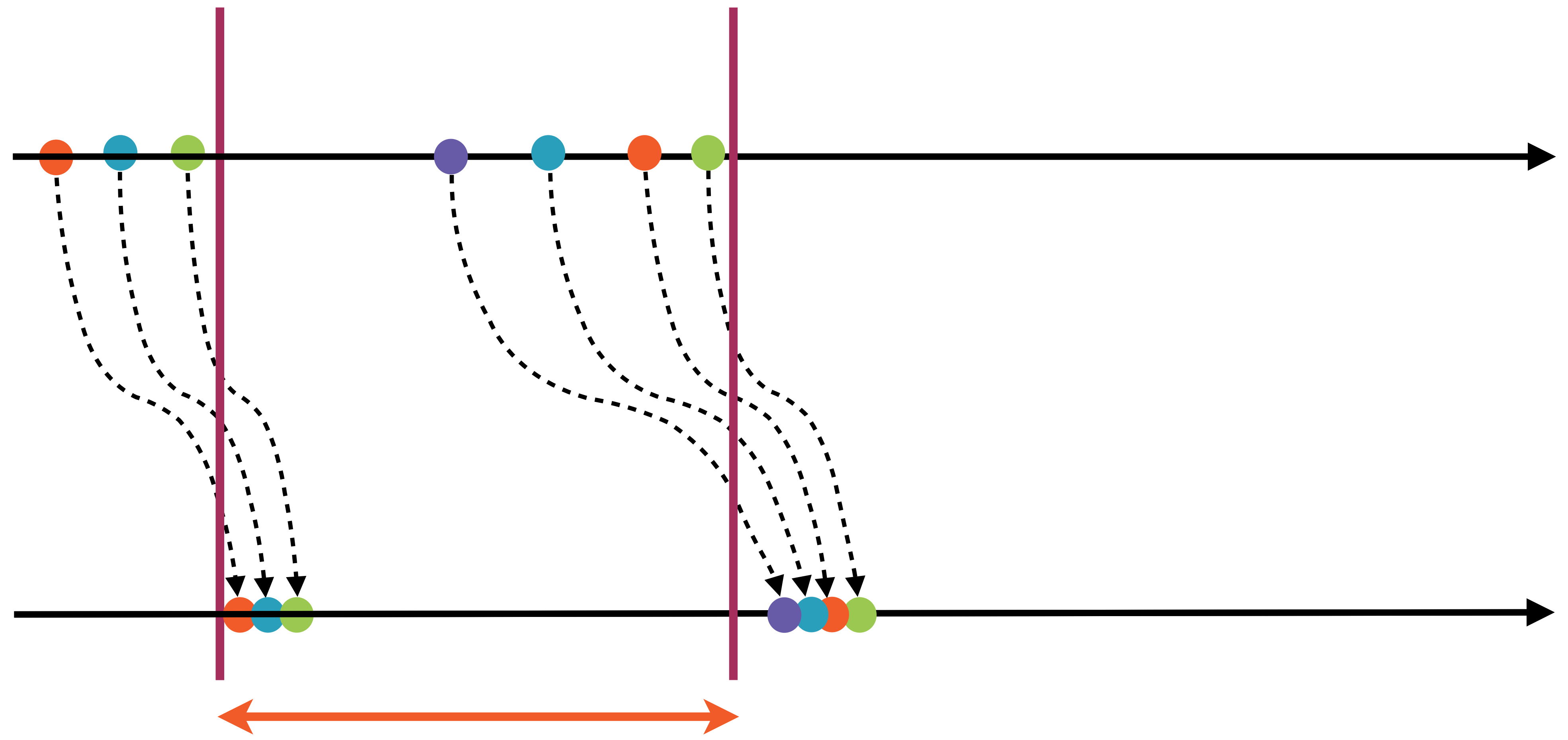
Collected into a micro-batch which is processed and written out to the sink

Higher End-to-end Latencies



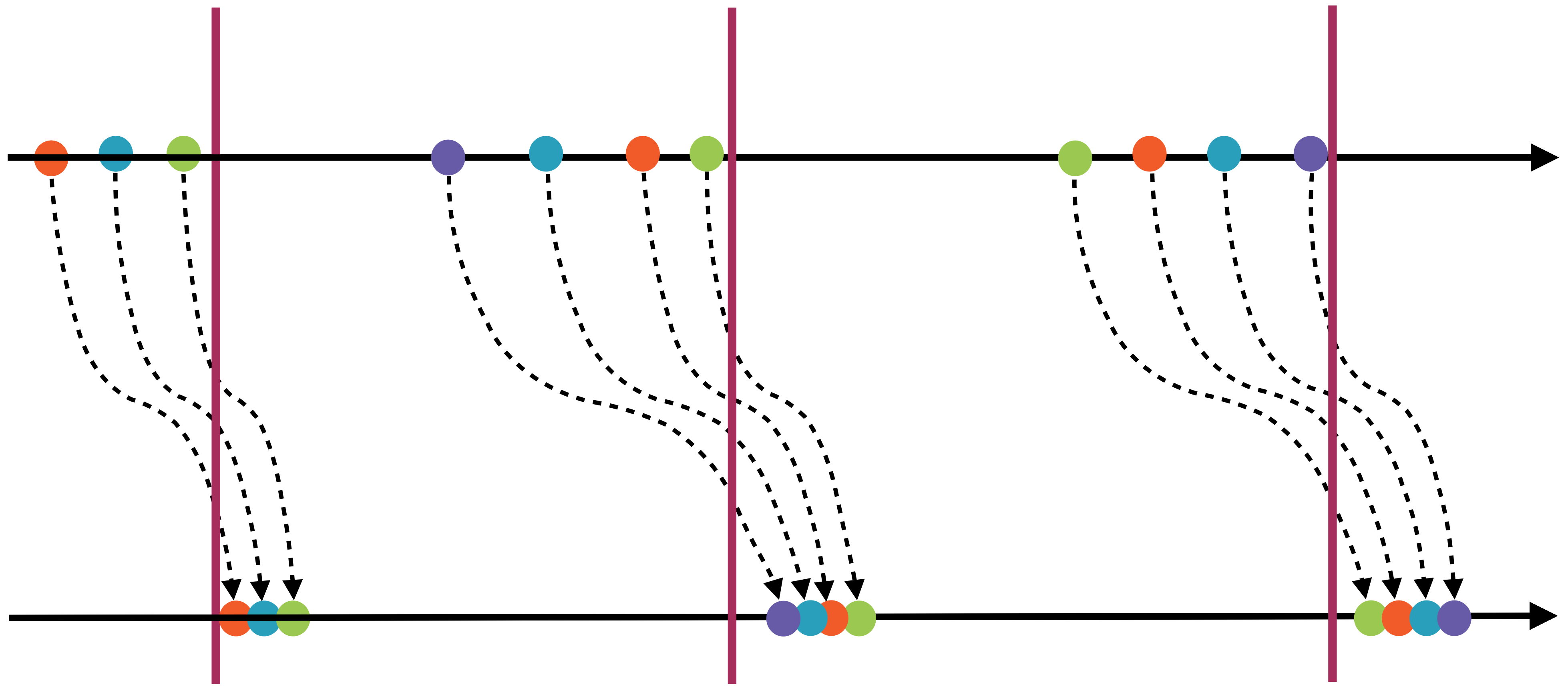
Collected into a micro-batch which is processed and written out to the sink

Higher End-to-end Latencies

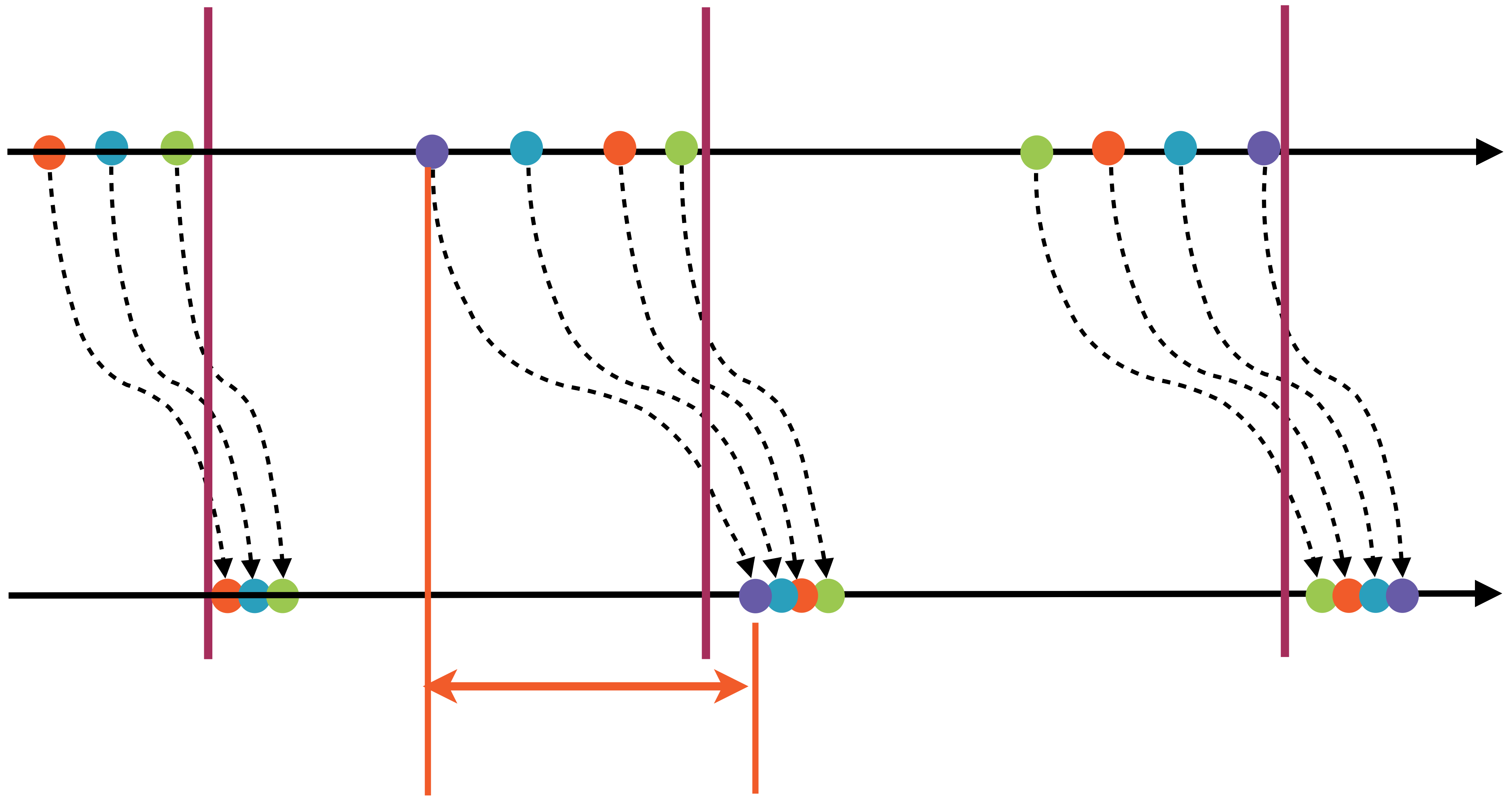


Interval before another micro-batch is created and processed

Higher End-to-end Latencies



Higher End-to-end Latencies



End-to-end latency for processing is usually in the order of seconds

Continuous Processing in Spark

Continuous Processing in Spark

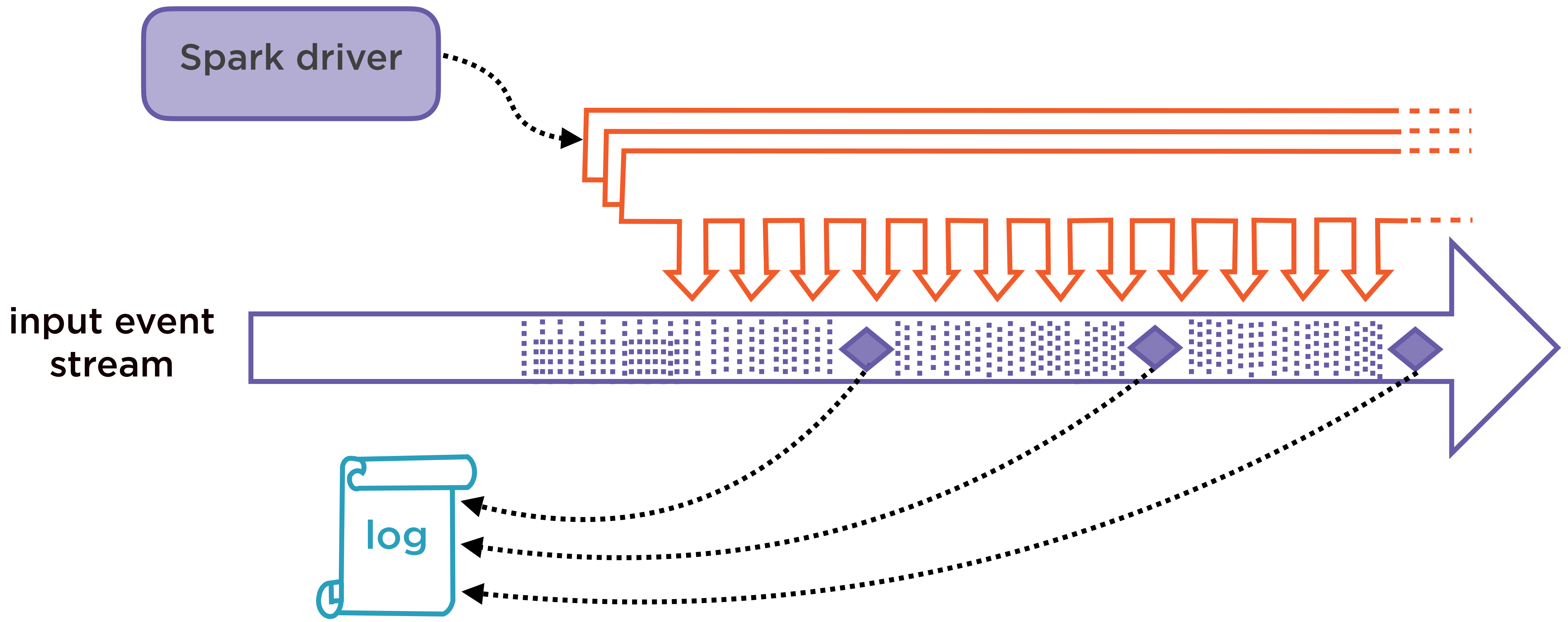
Experimental stream processing mode in Spark

Data streams processed using long-running tasks

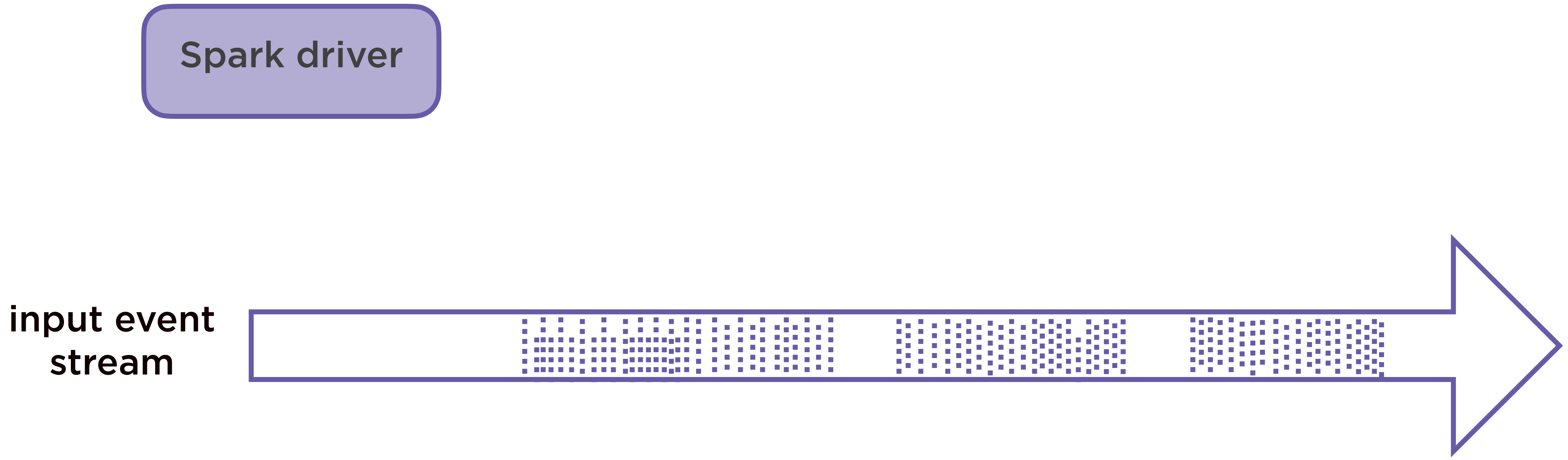
End-to-end latencies a few milliseconds

At least-once fault-tolerance guarantees

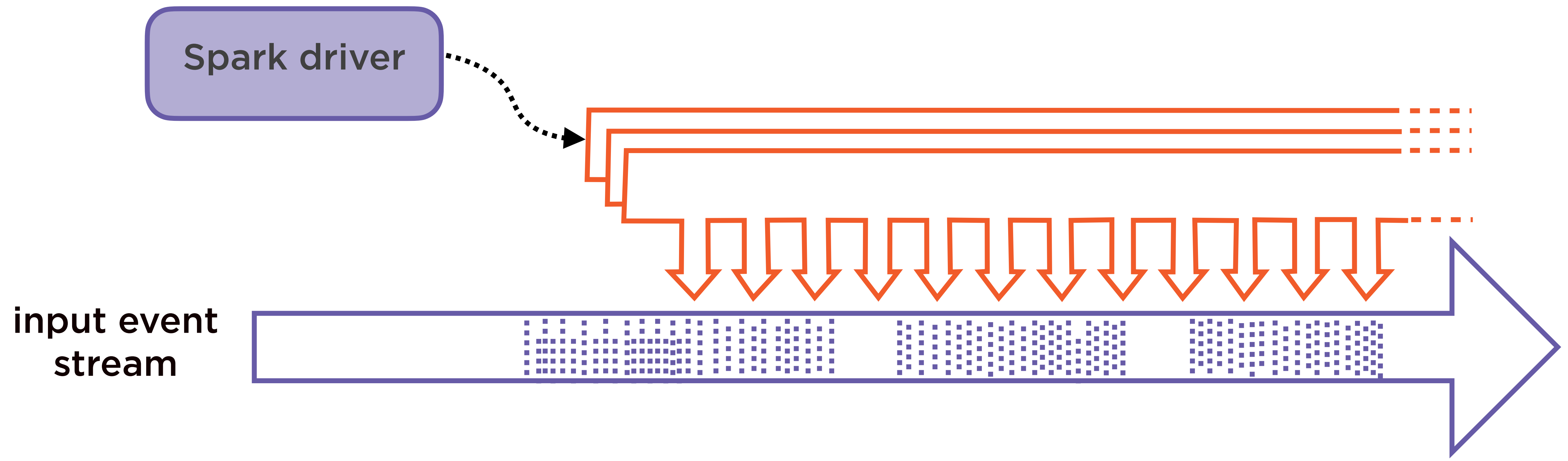
Continuous Processing in Spark



Continuous Processing in Spark

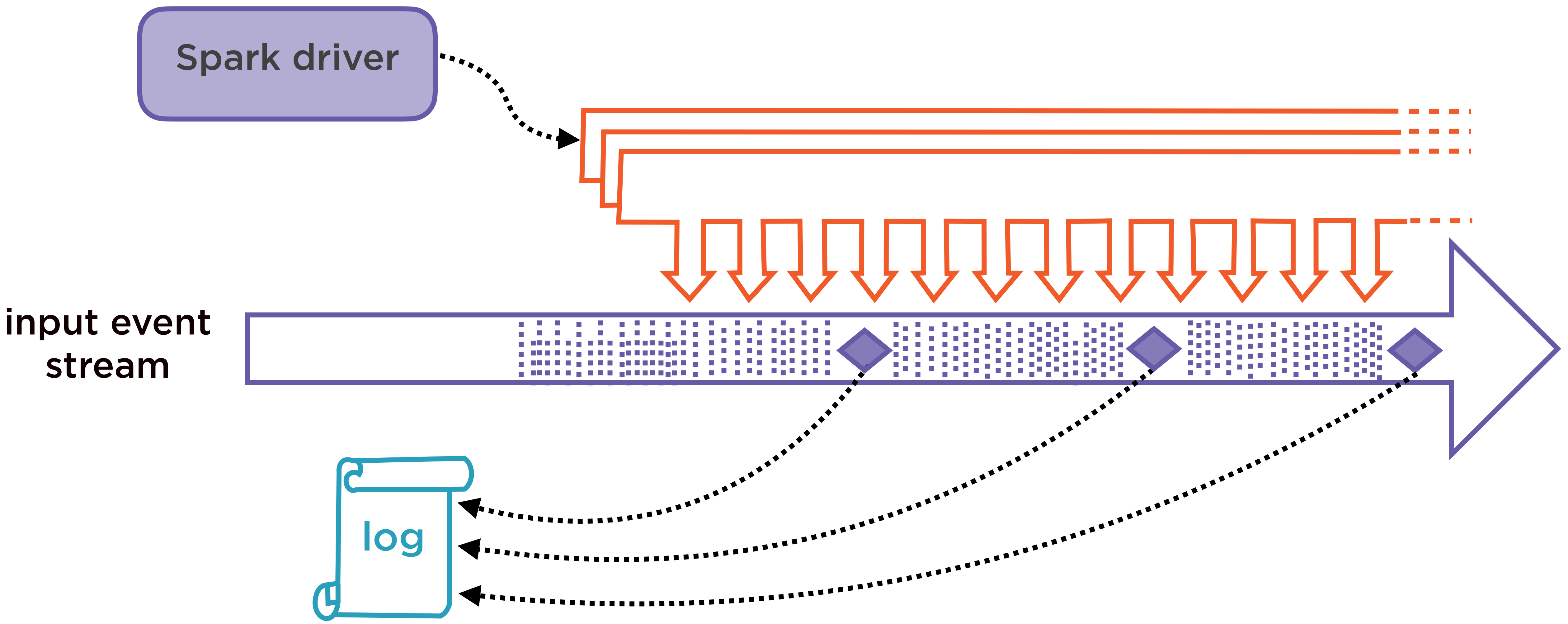


Continuous Processing in Spark



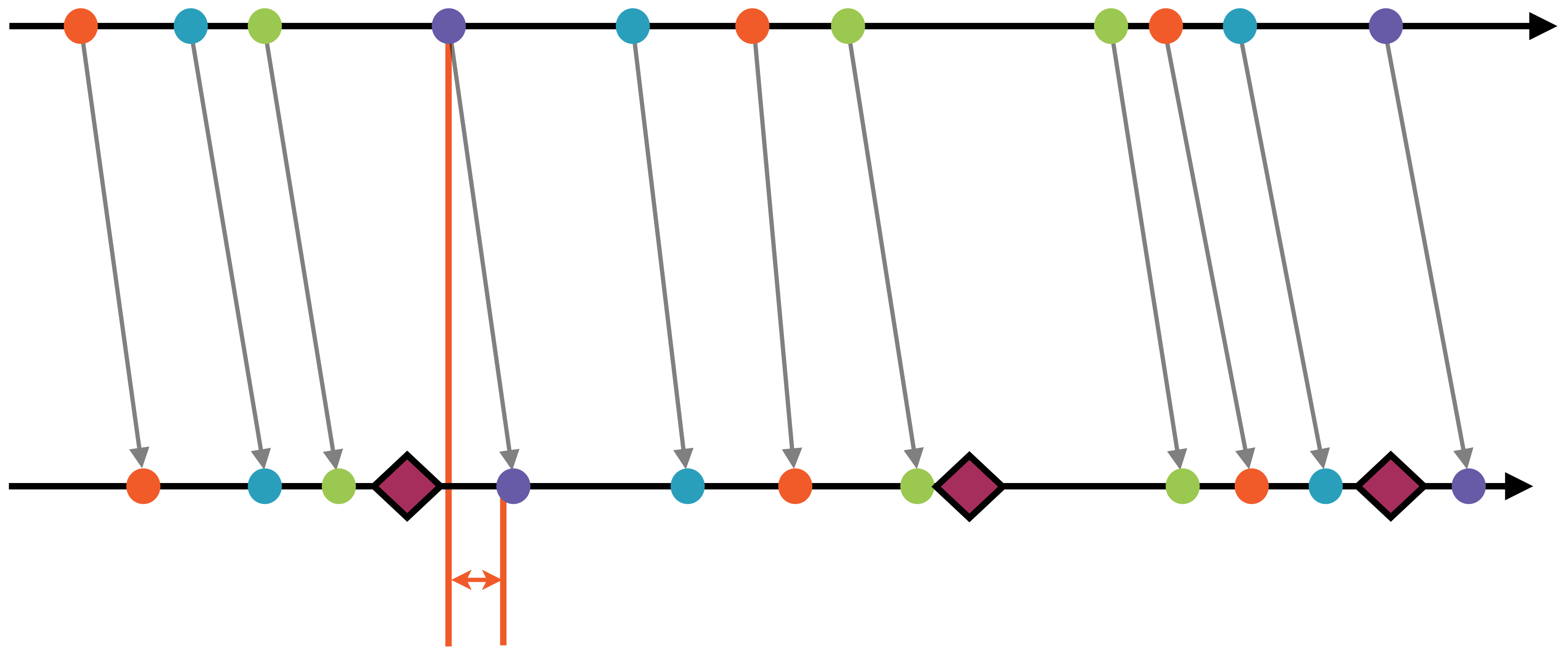
Long-running tasks to continuous process the input stream of events

Continuous Processing in Spark



Processed data written to write-ahead logs every epoch

Lower End-to-end Latencies



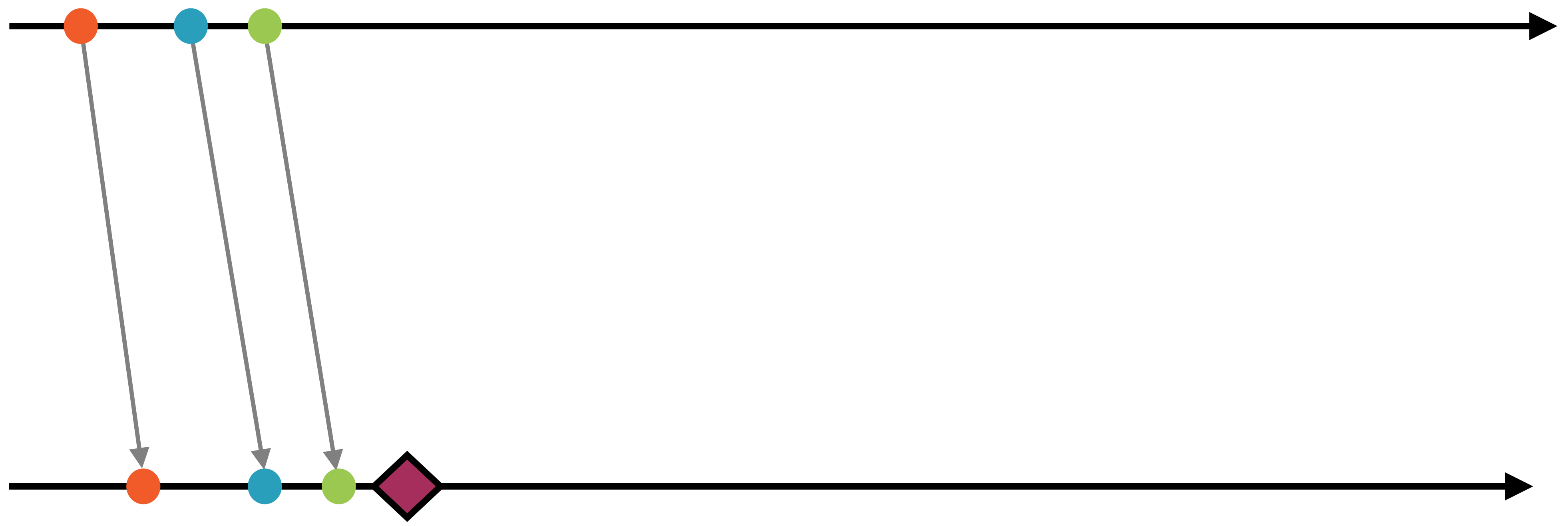
Lower End-to-end Latencies

Incoming streaming data at the source

Lower End-to-end Latencies

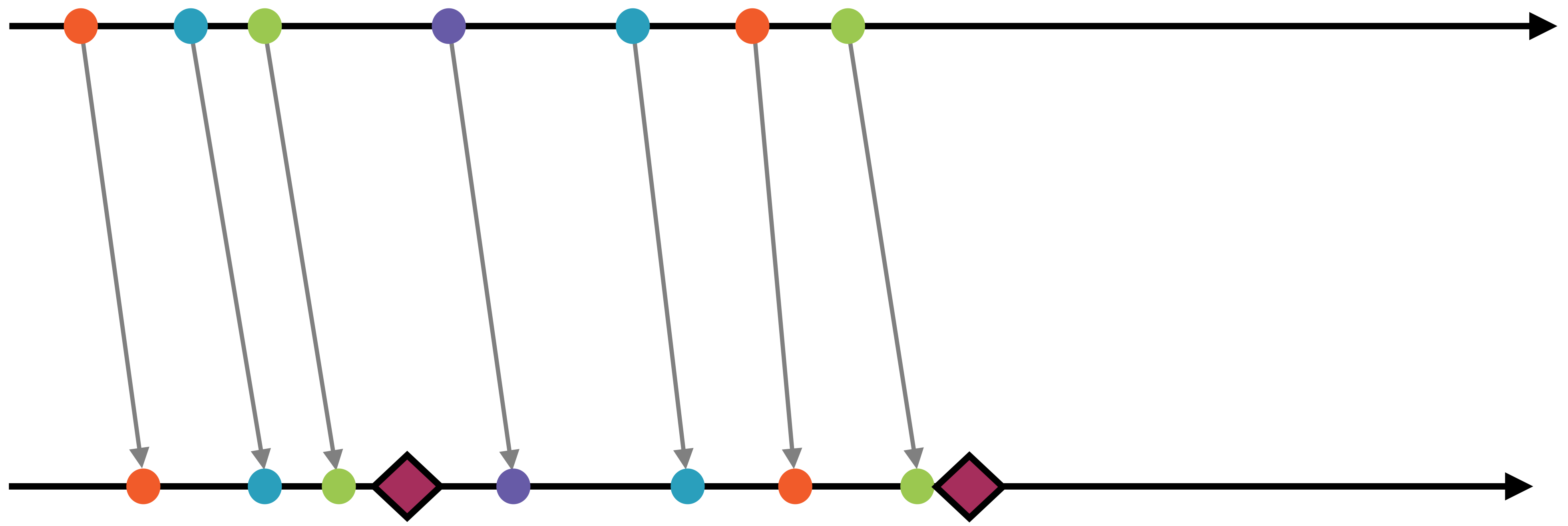
Processed continuously using long running jobs

Lower End-to-end Latencies

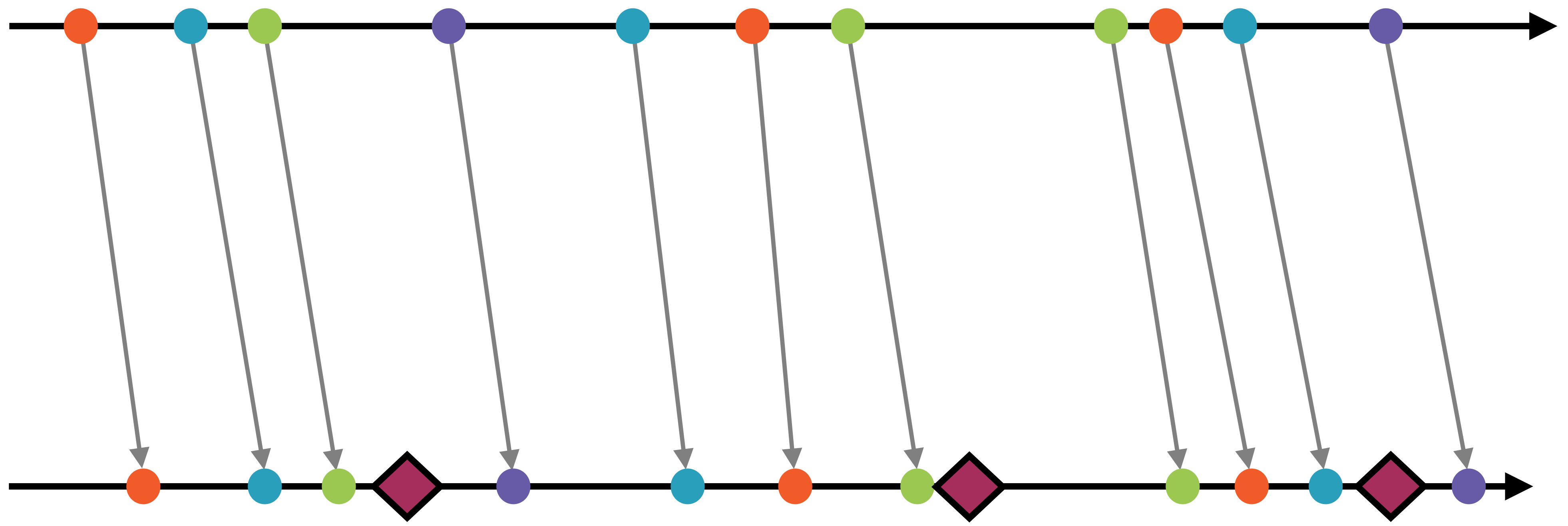


Epoch markers used to write progress of query out to checkpoint locations

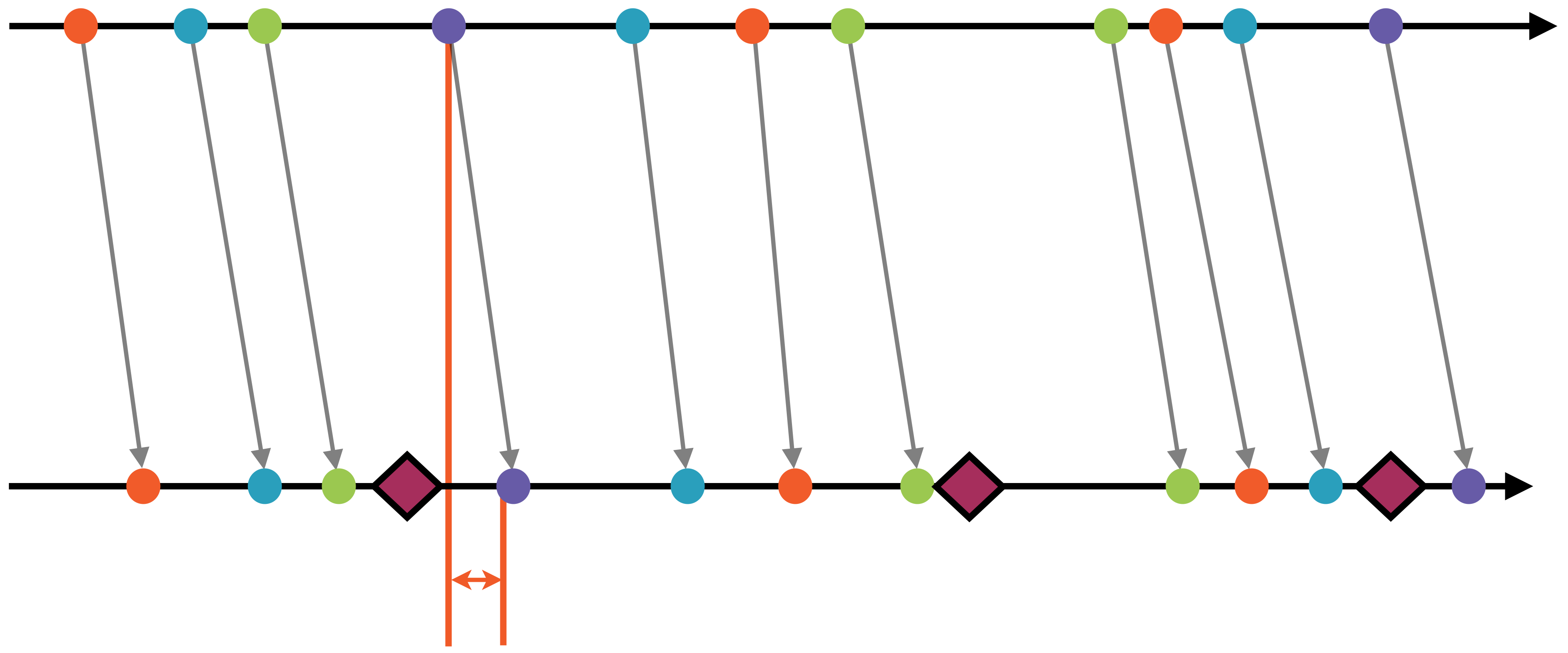
Lower End-to-end Latencies



Lower End-to-end Latencies



Lower End-to-end Latencies



End-to-end latencies for processing in milliseconds

Micro-batches vs. Continuous Processing

Micro-batches	Continuous Processing
Latency in hundreds of milliseconds	Latency in few milliseconds
Exactly-once guarantee	Only at-least-once guarantee
Spark Streaming launches many periodic tasks, each short-lived	Spark Streaming launches a few, long-running tasks
Basis of Structured Streaming since Spark 2.0	Introduced as experimental feature in Spark 2.3
All operations supported	Restrictions apply - e.g. aggregation functions currently unsupported

Demo

Continuous processing using a rate source

Summary

Stream processing models

Micro-batch execution and continuous processing

Considerations of latency, scaling, and recovery

At-least-once guarantees

Running Spark in continuous processing mode

Up Next:

Understanding Query Planning