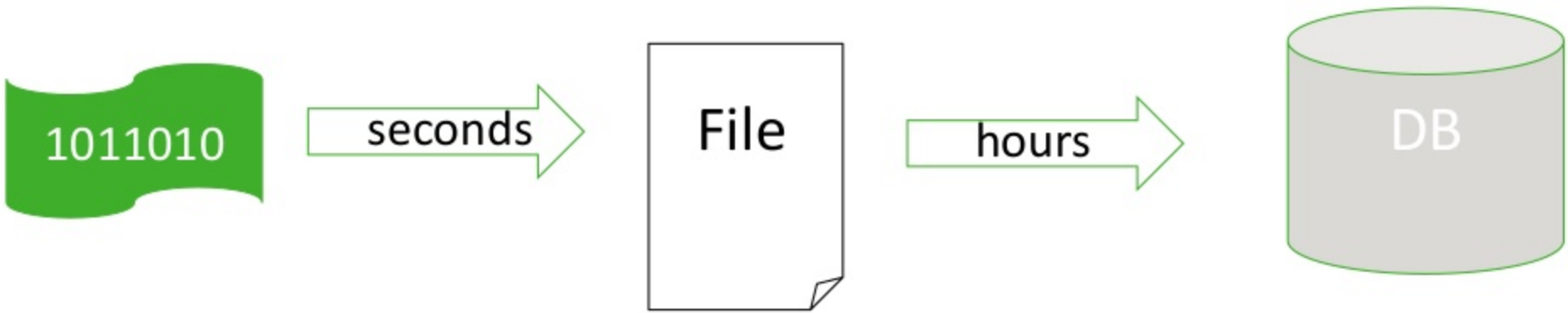
Using Spark Streaming and NiFi for the Next Generation of ETL in the Enterprise

Andrew Psaltis

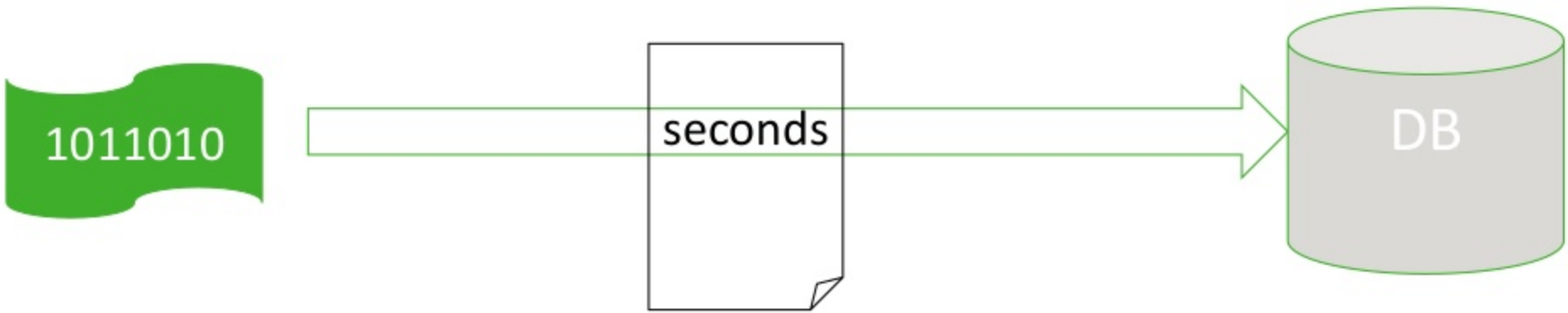
Regional CTO APAC

Traditional ETL



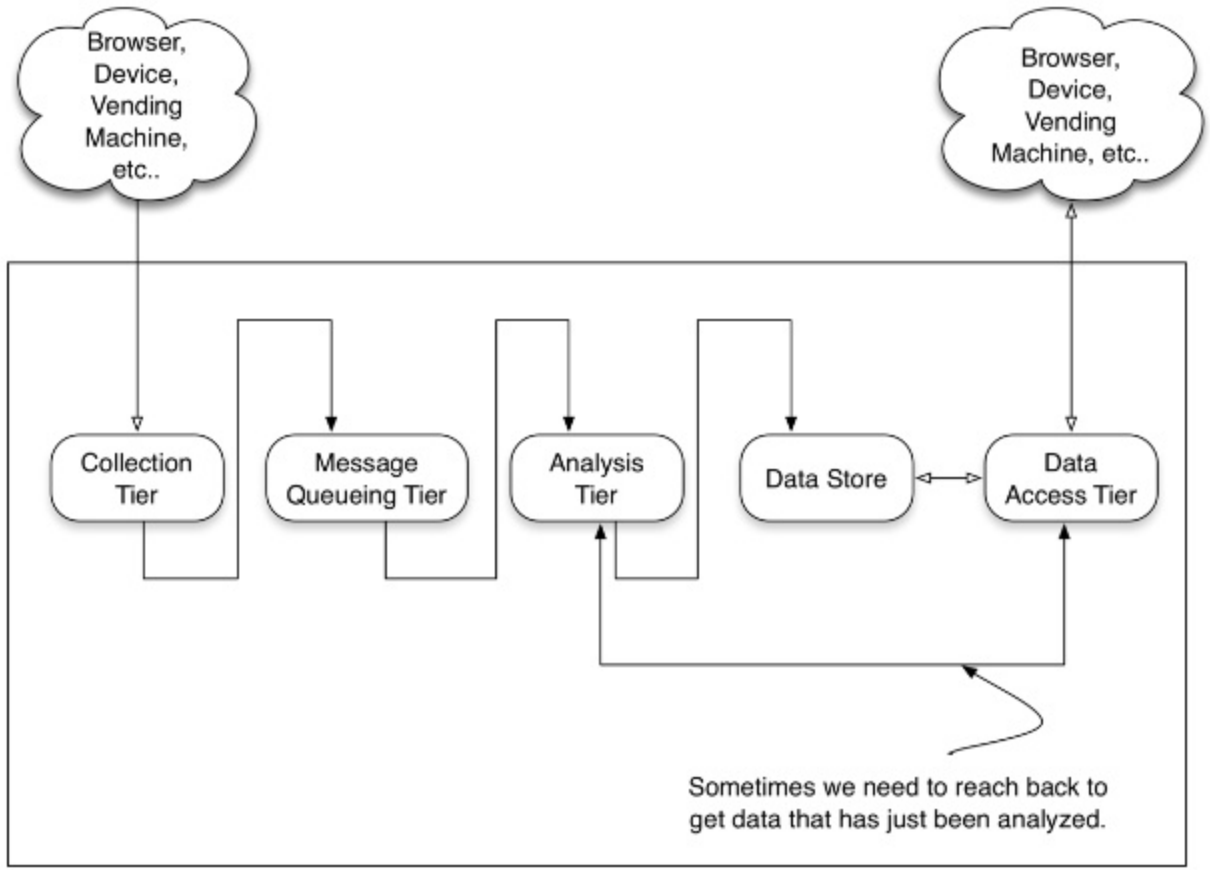
HORTONWORKS

Streaming ETL



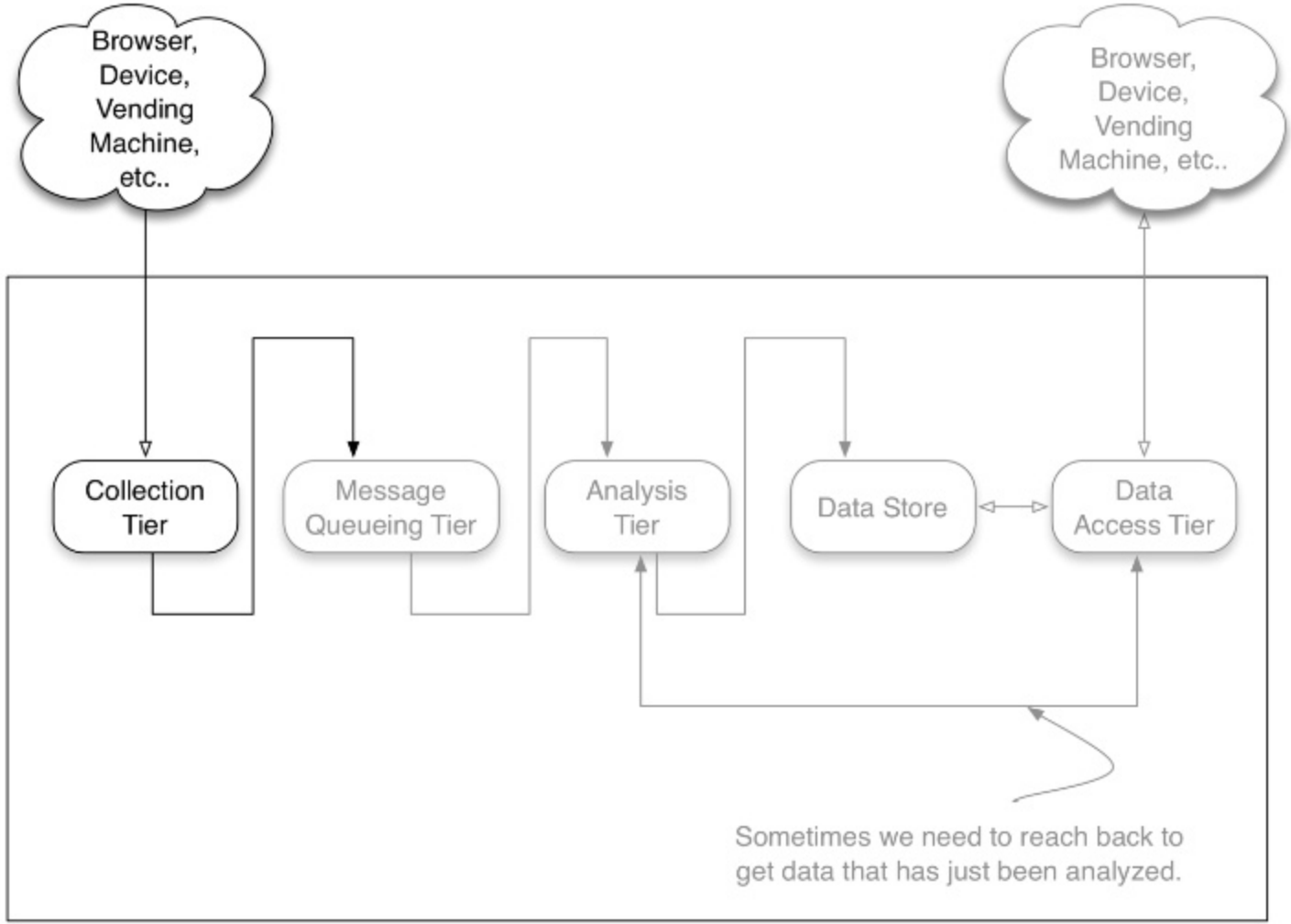
HORTONWORKS

Reference Streaming Architecture



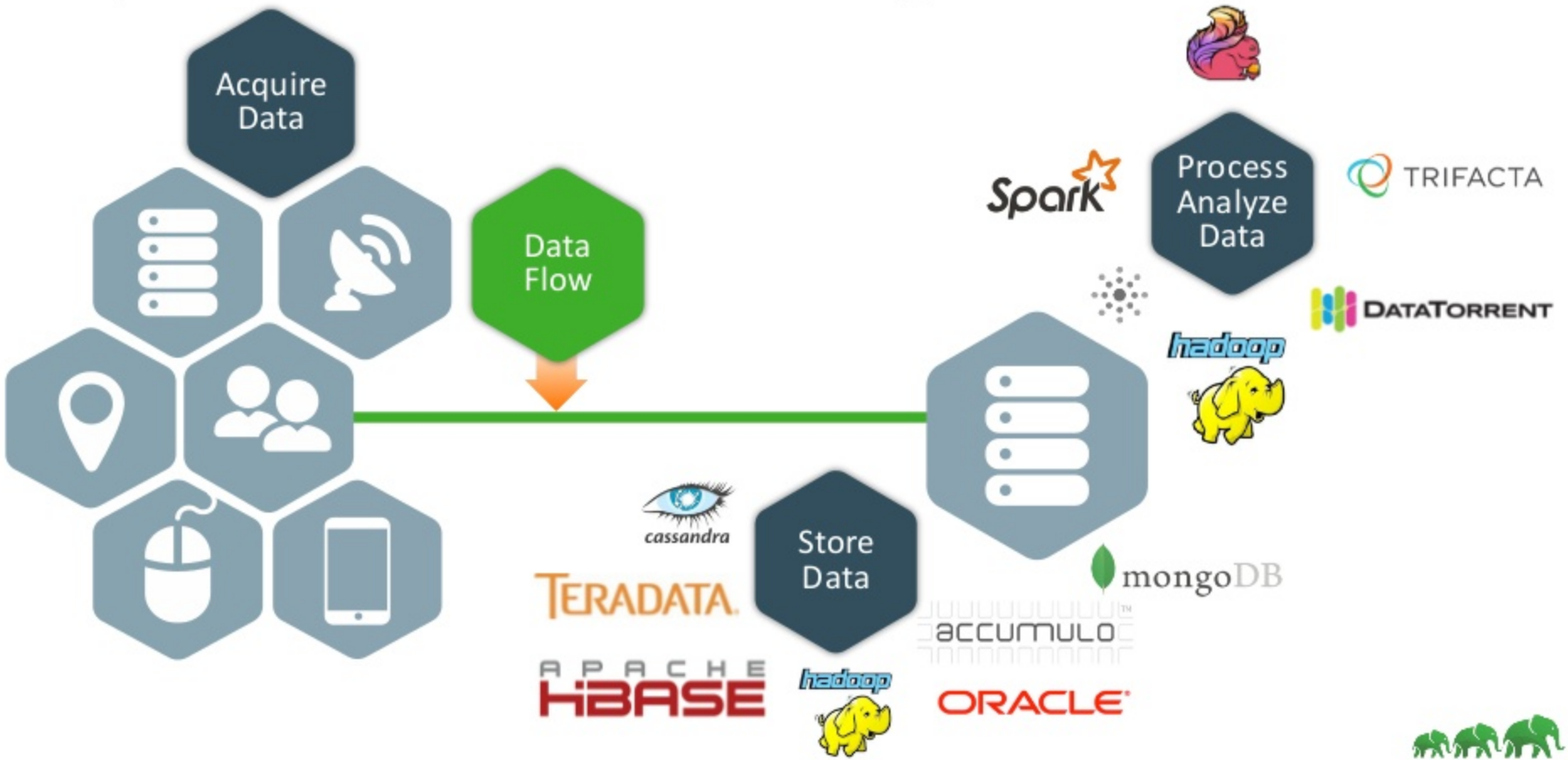
HORTONWORKS


Apache NiFi
HORTONWORKS

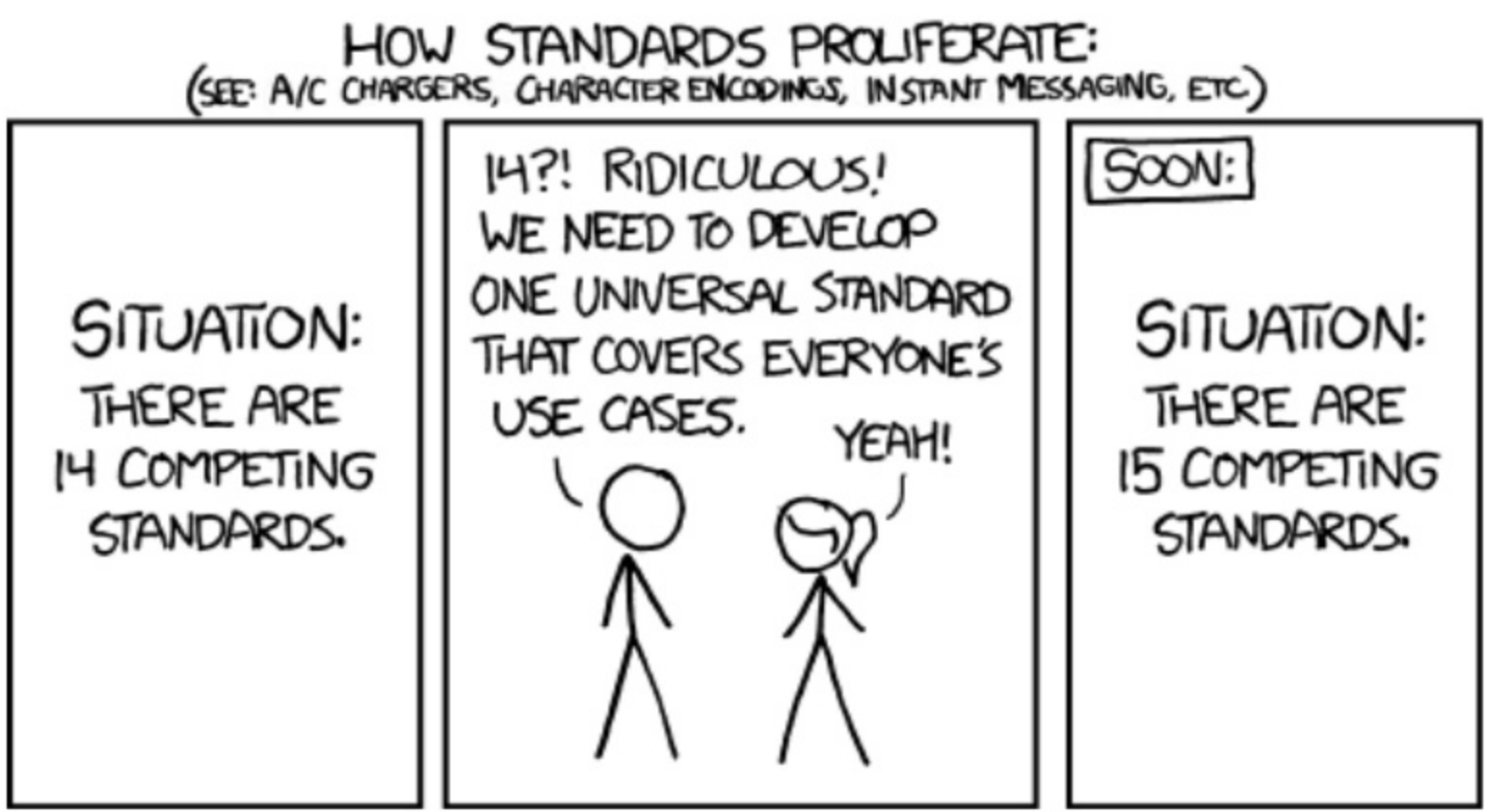


HORTONWORKS

Simplistic View of Dataflows: Easy, Definitive



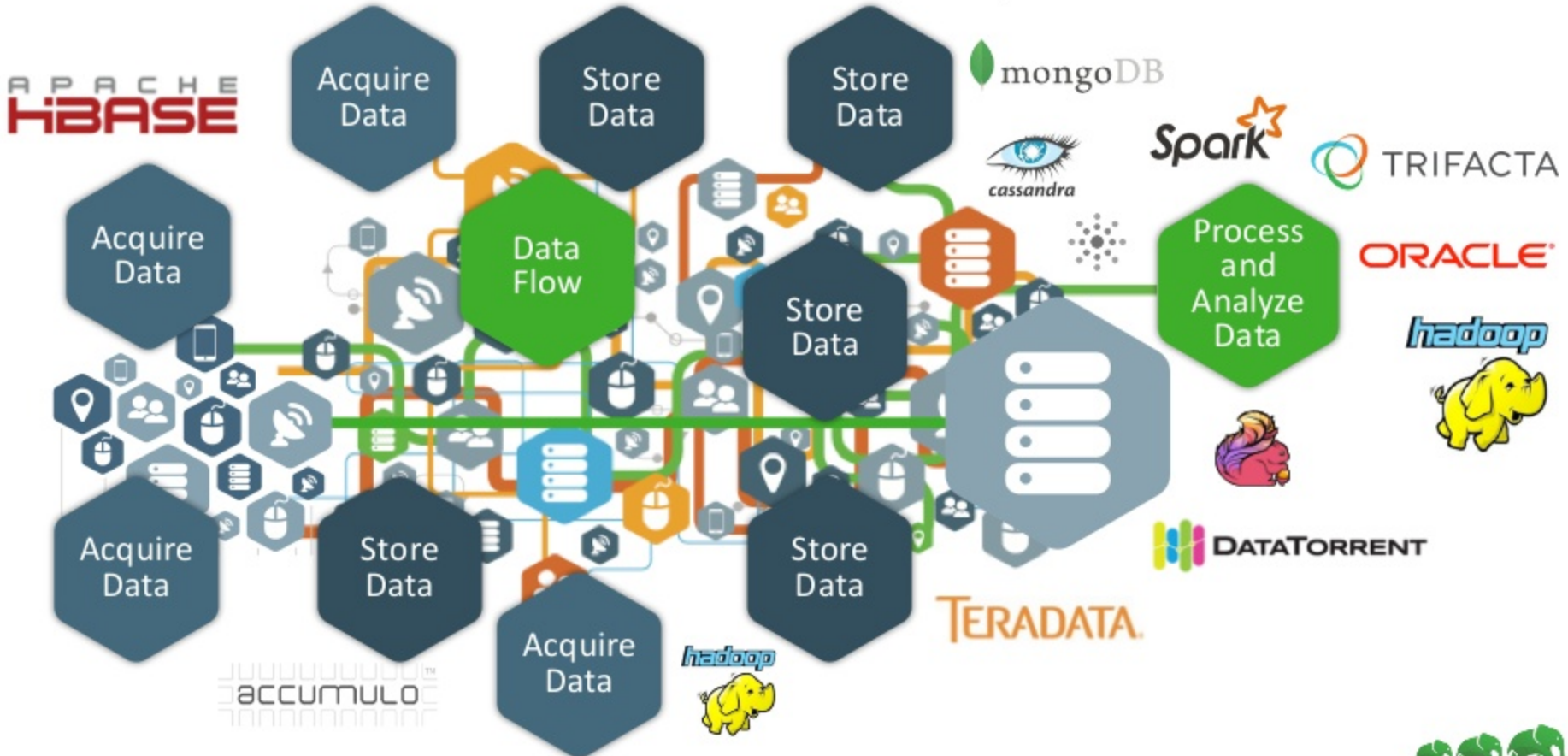
HORTONWORKS



Standards: http://xkcd.com/927/

HORTONWORKS

Realistic View of Dataflows: Complex, Convoluted



HORTONWORKS

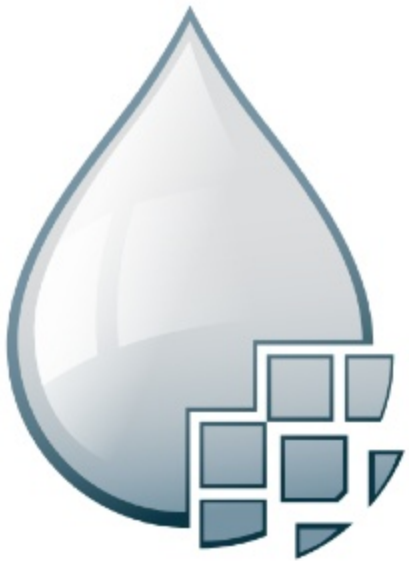
The National Security Agency Years

- **Created in 2006**
- **Improved over eight years**
  - Simple initial vision – Visio for real-time dataflow management
- **National Security Agency donated the codebase to the ASF in late 2014**

HORTONWORKS

Apache NiFi

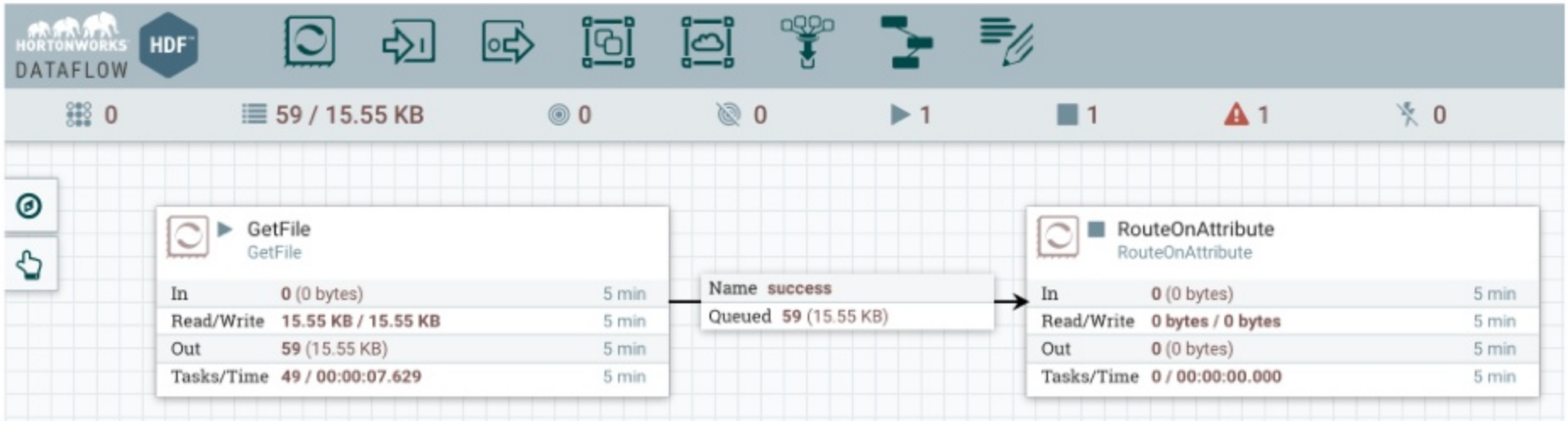
Key Features



- Guaranteed delivery
- Data buffering
  - Backpressure
  - Pressure release
- Prioritized queuing
- Flow specific QoS
  - Latency vs. throughput
  - Loss tolerance
- Data provenance
- Supports push and pull models
- Recovery/recording a rolling log of fine-grained history
- Visual command and control
- Flow templates
- Pluggable/multi-role security
- Designed for extension
- Clustering

HORTONWORKS

Visual Command and Control



- Drag and drop processors to build a flow
- Start, stop, and configure components in real time
- View errors and corresponding error messages
- View statistics and health of data flow
- Create templates of common processor & connections

HORTONWORKS

Provenance/Lineage

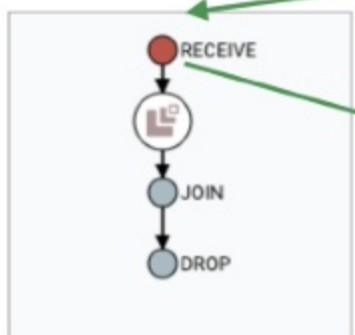
Displaying 13 of 104
Oldest event available: 11/15/2016 13:34:50 EST

Showing the most recent events.

ConsumeKafka | by component name

	Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type	
ⓘ	11/15/2016 13:35:03.8...	RECEIVE	379fc4f6-60e0-4151-9743-28...	44 bytes	ConsumeKafka	ConsumeKafka	⋔→
ⓘ	11/15/2016 13:35:02.7...	RECEIVE	78f8c38b-89fc-4d00-a8d8-51...	44 bytes	ConsumeKafka	ConsumeKafka	⋔→
ⓘ	11/15/2016 13:35:01.6...	RECEIVE	2bcd5124-bb78-489f-ad8a-7...	44 bytes	ConsumeKafka	ConsumeKafka	⋔→

- Tracks data at each point as it flows through the system
- Records, indexes, and makes events available for display
- Handles fan-in/fan-out, i.e. merging and splitting data
- View attributes and content at given points in time



Provenance Event

DETAILS | ATTRIBUTES | CONTENT

Attribute Values

filename
328717796819631
No value previously set

kafka.offset
44815
No value previously set

kafka.partition
6
No value previously set

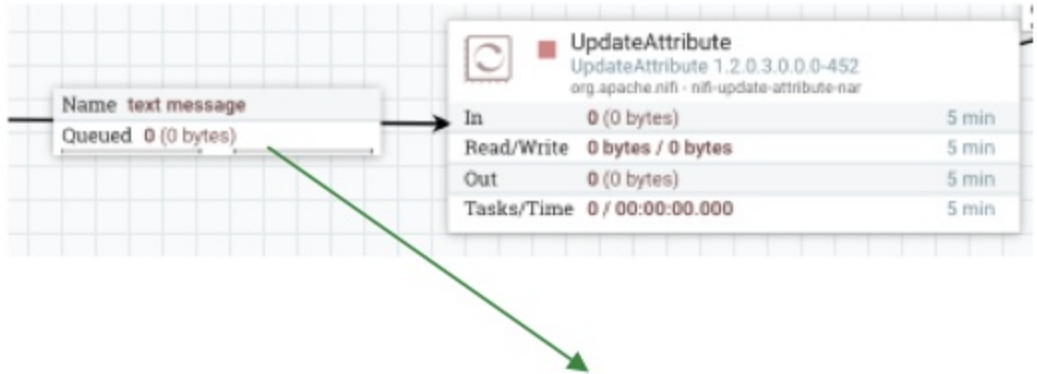
kafka.topic
nifi-testing
No value previously set

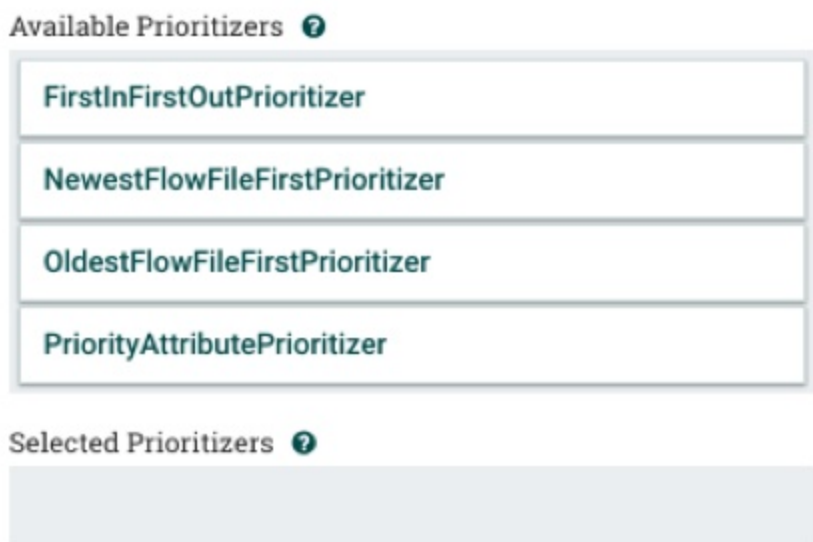
path
./
No value previously set

uuid

Prioritization

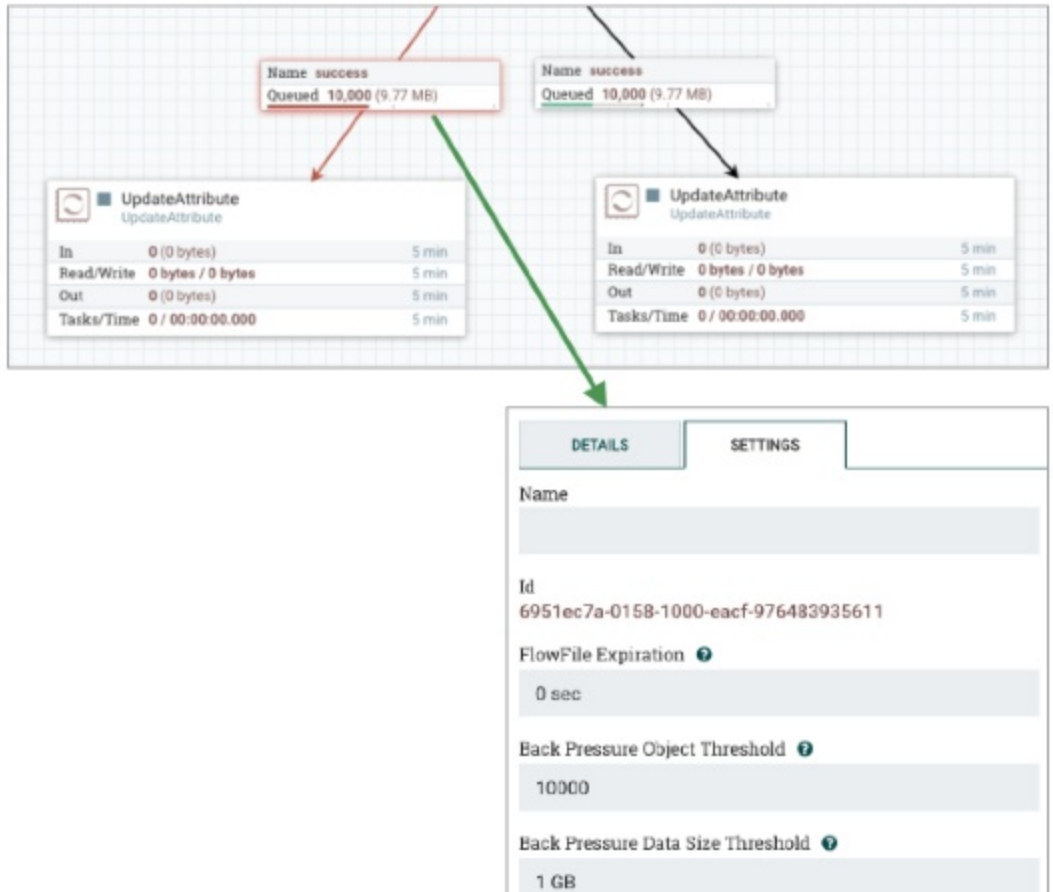
- Configure a prioritizer per connection
- Determine what is important for your data – time based, arrival order, importance of a data set
- Funnel many connections down to a single connection to prioritize across data sets
- Develop your own prioritizer if needed





Back-Pressure

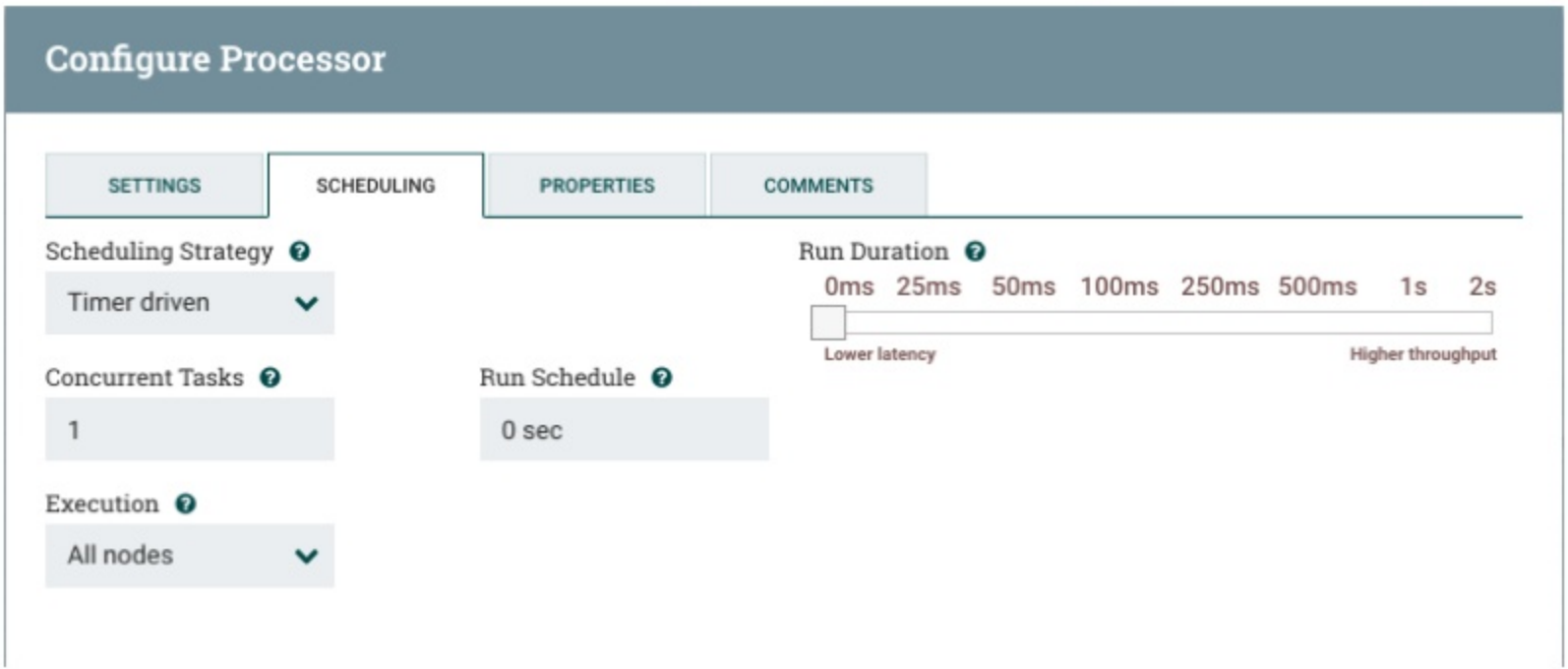
- Configure back-pressure per connection
- Based on number of FlowFiles or total size of FlowFiles
- Upstream processor no longer scheduled to run until below threshold



HORTONWORKS

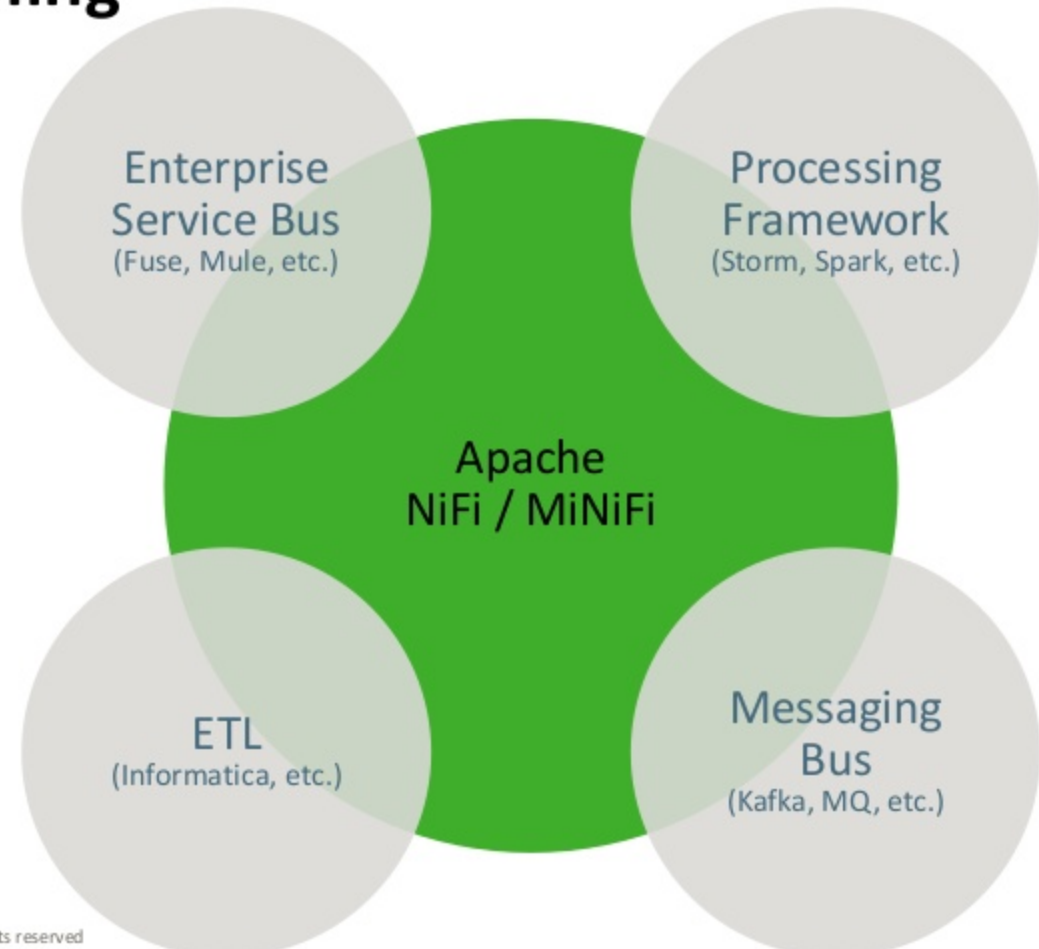
Latency vs. Throughput

Choose between lower latency, or higher throughput on each processor



HORTONWORKS

NiFi Positioning



HORTONWORKS

Apache NiFi / Processing Frameworks

NiFi

Simple event processing

- Primarily feed data into processing frameworks, can process data, with a focus on simple event processing
- Operate on a single piece of data, or in correlation with an enrichment dataset (enrichment, parsing, splitting, and transformations)
- Can scale out, but scale up better to take full advantage of hardware resources, run concurrent processing tasks/threads (processing terabytes of data per day on a single node)

Not another distributed processing framework, but to feed data into those

Processing Frameworks (Storm, Spark, etc.)

Complex and distributed processing

- Complex processing from multiple streams (JOIN operations)
- Analyzing data across time windows (rolling window aggregation, standard deviation, etc.)
- Scale out to thousands of nodes if needed

Not designed to collect data or manage data flow

HORTONWORKS

Apache NiFi / Messaging Bus Services

NiFi

Provide dataflow solution

- Centralized management, from edge to core
- Great traceability, event level data provenance starting when data is born
- Interactive command and control – real time operational visibility
- Dataflow management, including prioritization, back pressure, and edge intelligence
- Visual representation of global dataflow

Not a messaging bus, flow maintenance needed when you have frequent consumer side updates

Messaging Bus (Kafka, JMS, etc.)

Provide messaging bus service

- Low latency
- Great data durability
- Decentralized management (producers & consumers)
- Low broker maintenance for dynamic consumer side updates

Not designed to solve dataflow problems (prioritization, edge intelligence, etc.)

Traceability limited to in/out of topics, no lineage

Lack of global view of components/connectivities

HORTONWORKS

Apache NiFi / Integration, or Ingestion, Frameworks

NiFi

End user facing dataflow management tool

- Out of the box solution for dataflow management
- Interactive command and control in the core, design and deploy on the edge
- Flexible failure handling at each point of the flow
- Visual representation of global dataflow and connectivities
- Native cross data center communication
- Data provenance for traceability

Not a library to be embedded in other applications

Integration framework (Spring Integration, Camel, etc), ingestion framework (Flume, etc)

Developer facing integration tool with a focus on data ingestion

- A set of tools to orchestrate workflow
- A fixed design and deploy pattern
- Leverage messaging bus across disconnected networks

Developer facing, custom coding needed to optimize

Pre-built failure handling, lack of flexibility

No holistic view of global dataflow

No built-in data traceability

HORTONWORKS

Apache NiFi / ETL Tools

NiFi

NOT schema dependent

- Dataflow management for both structured and unstructured data, powered by separation of metadata and payload
- Schema is not required, but you can have schema
- Minimum modeling effort, just enough to manage dataflows
- Do the plumbing job, maximize developers' brainpower for creative work

Not designed to do heavy lifting transformation work for DB tables (JOIN datasets, etc.). You can create custom processors to do that, but long way to go to catch up with existing ETL tools from user experience perspective (GUI for data wrangling, cleansing, etc.)

ETL (Informatica, etc.)

Schema dependent

- Tailored for Databases/WH
- ETL operations based on schema/data modeling
- Highly efficient, optimized performance

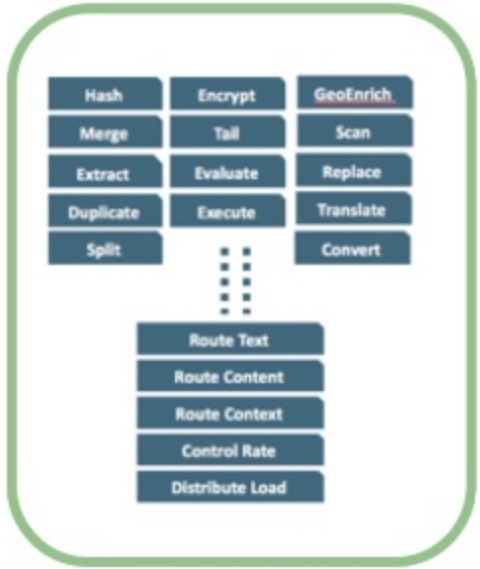
Must pre-prepare your data, time consuming to bui data modeling, and maintain schemas

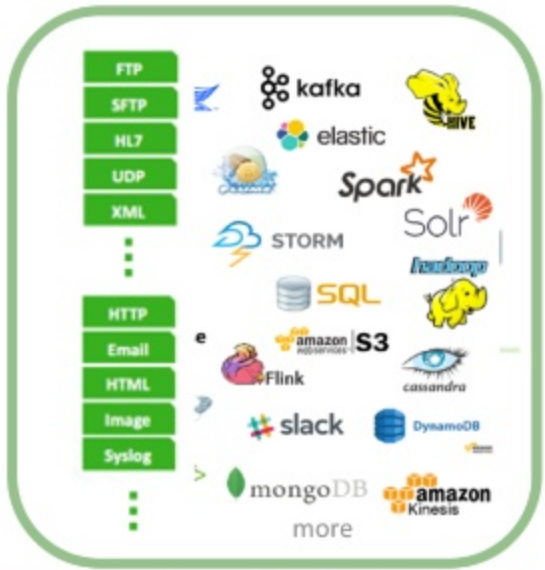
Not geared towards handling unstructured data, PD Audio, Video, etc.

Not designed to solve dataflow problems

HORTONWORKS

NiFi Big Picture Pattern: Diverse Flows from One Tool





"Swiss Army Knife of Data Movement"

Acquire Data
diverse sources

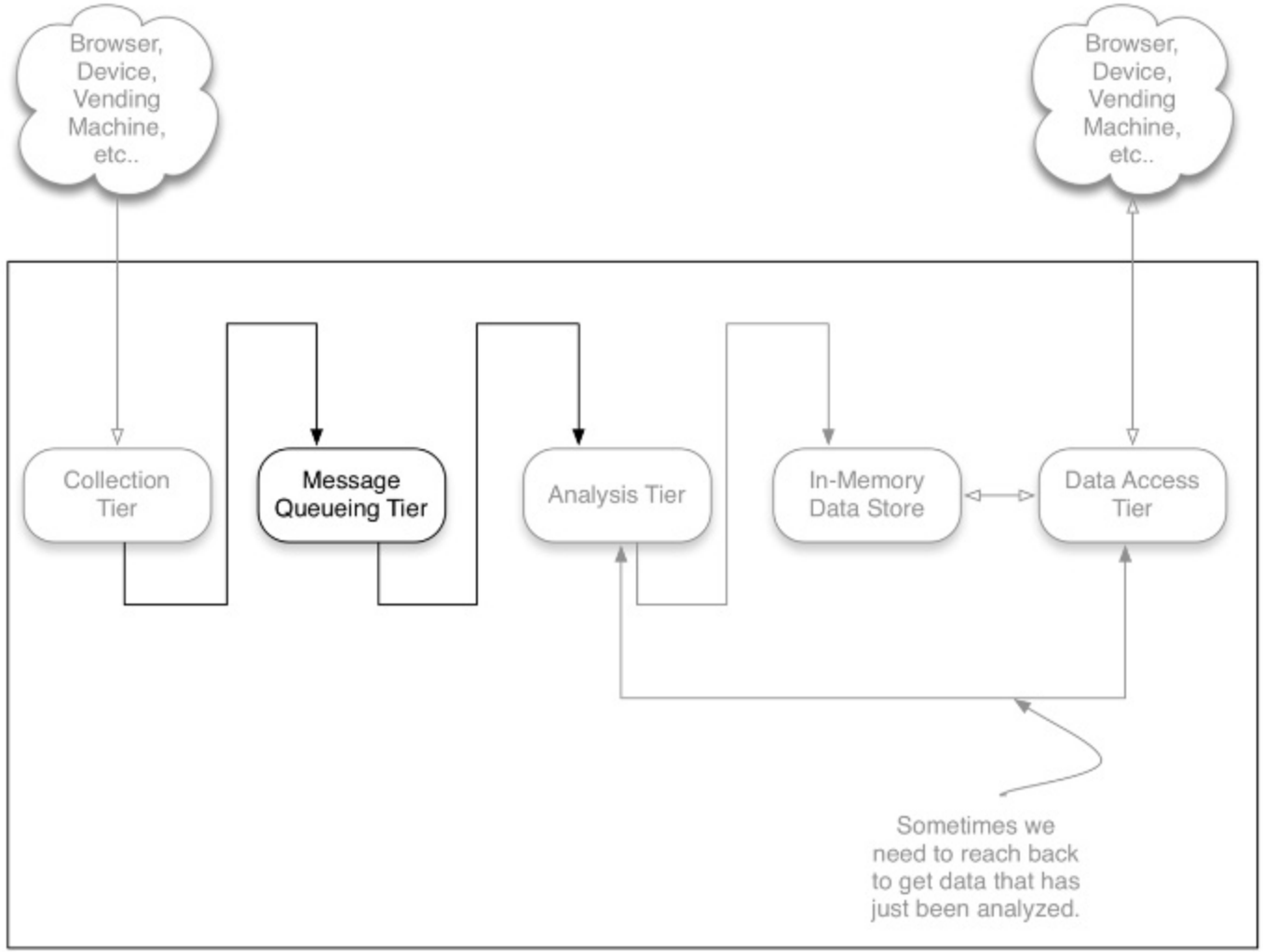
Process Data
parse
filter
transform
enrich
route
apply schema

Split / Merge

Deliver Data
diverse targets

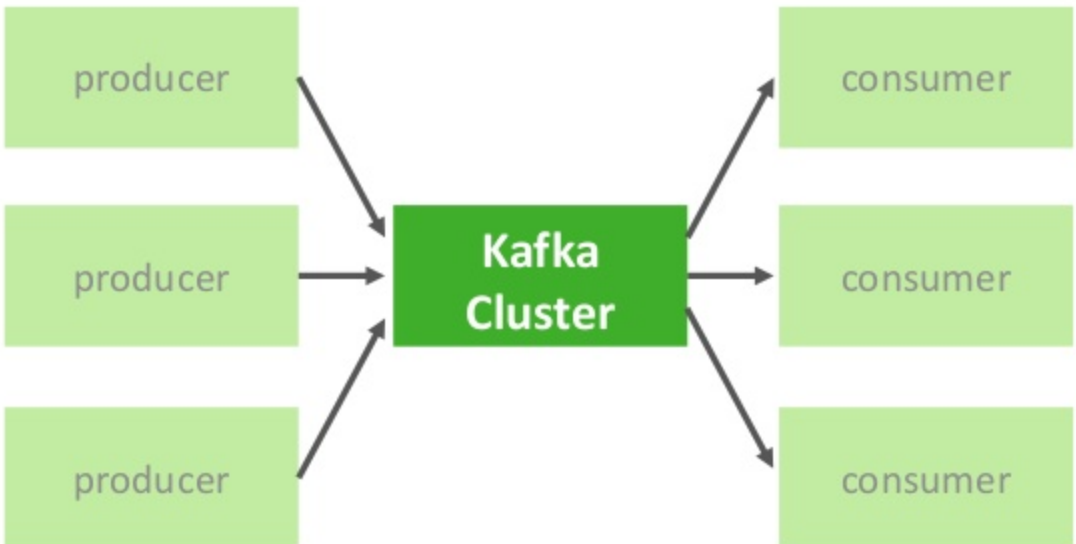
HORTONWORKS


Apache Kafka
HORTONWORKS



HORTONWORKS

What Is Apache Kafka?



- Distributed streaming platform that allows publishing and subscribing to streams of records
- Streams of records are organized into categories called topics
- Topics can be partitioned and/or replicated
- Records consist of a key, value, and timestamp

http://kafka.apache.org/intro

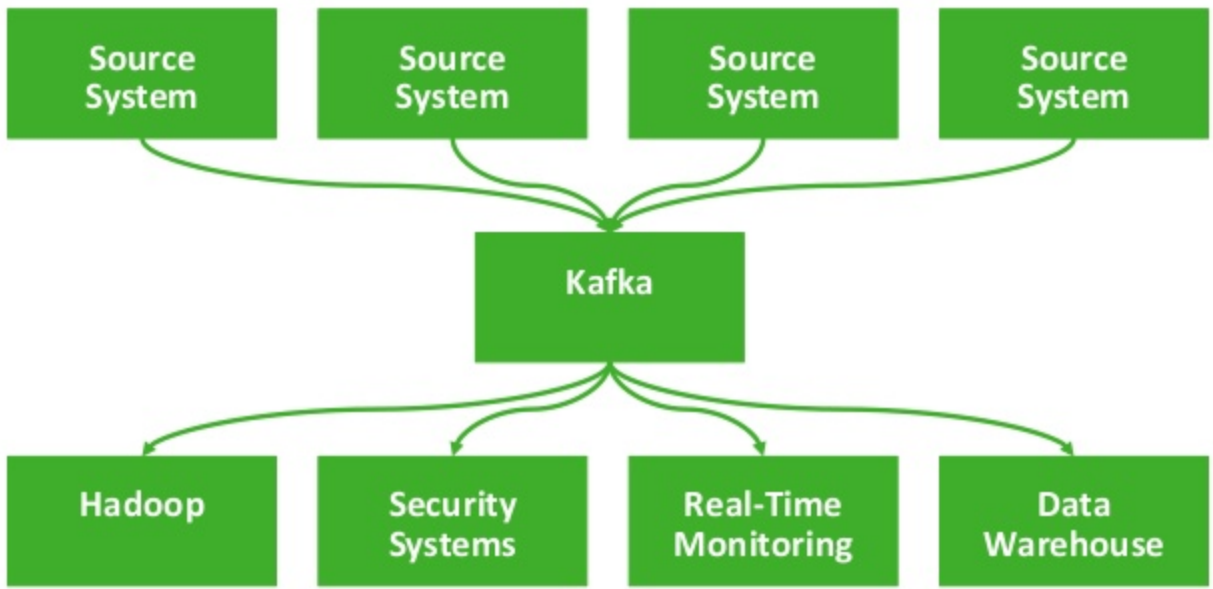
High throughput, distributed system. Designed to operate at large scale.

Why Kafka

Producers

Brokers

Consumers



Kafka decouples data pipelines

HORTONWORKS

Overview of Topics

- Topics are a **partitioned** ordered, immutable sequence of messages
- Messages are retained for a configurable amount of time (24 hours, 7 days, etc.)
- Each consumer retains its own offset in the partition

HORTONWORKS

Understanding Partitions

- Partitions are an ordered, immutable sequence of messages
- Each partition is **replicated** for fault tolerance
- A replicated partition has one broker that acts as the **leader** *and the rest as* **followers**

HORTONWORKS

Consuming Messages

- Messages are consumed in Kafka by a **consumer group**
- Each individual consumer is labeled with a group name
- Each message in a topic is sent to one consumer in the group

HORTONWORKS


Spark Structured
Streaming
HORTONWORKS

Structured Streaming

Turns stream processing into SQL

fast, scalable, fault-tolerant

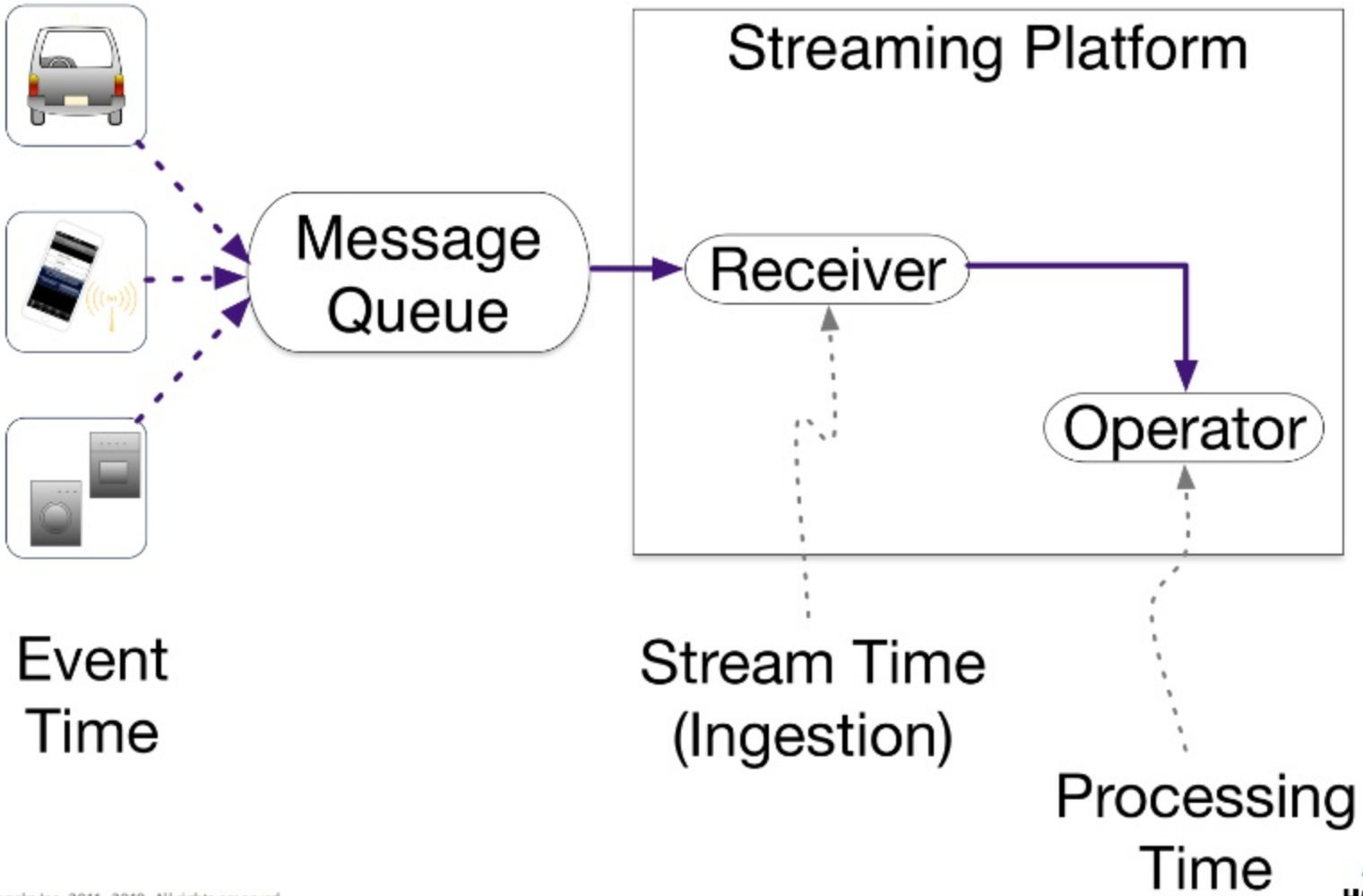
Unifies high level APIs with Spark

deal with complex data and complex workloads

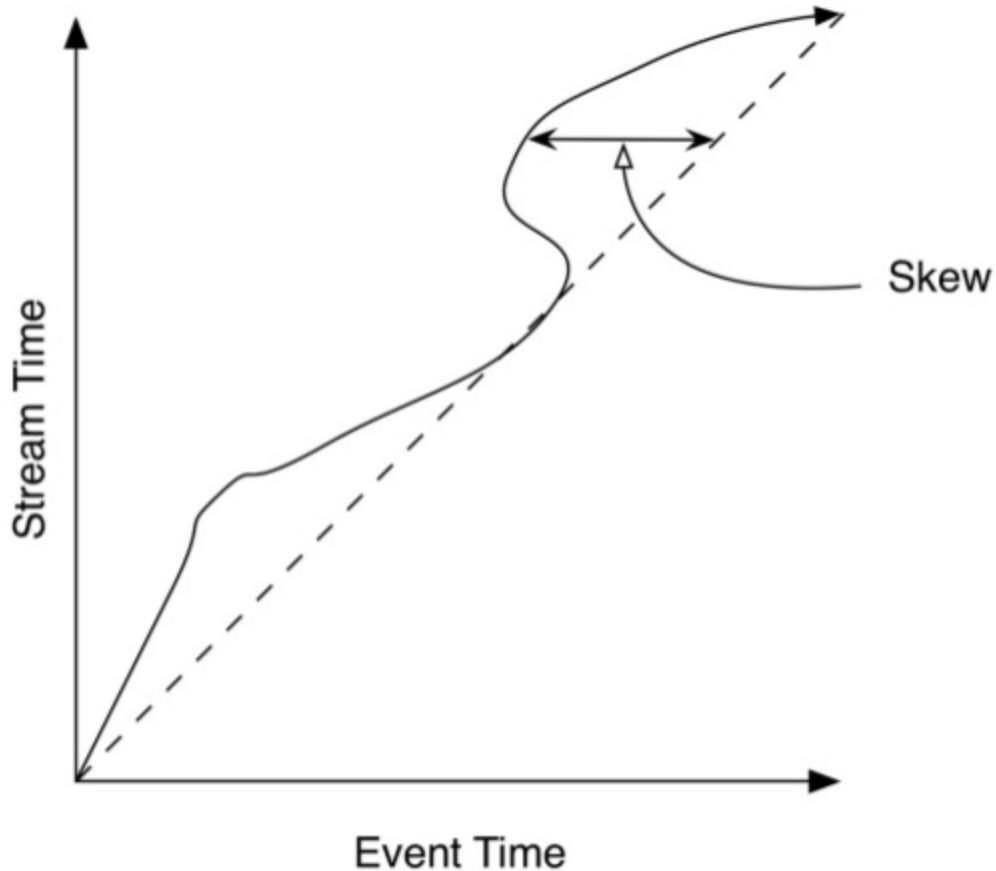
HORTONWORKS

Thinking about time

HORTONWORKS



Time Skew



HORTONWORKS

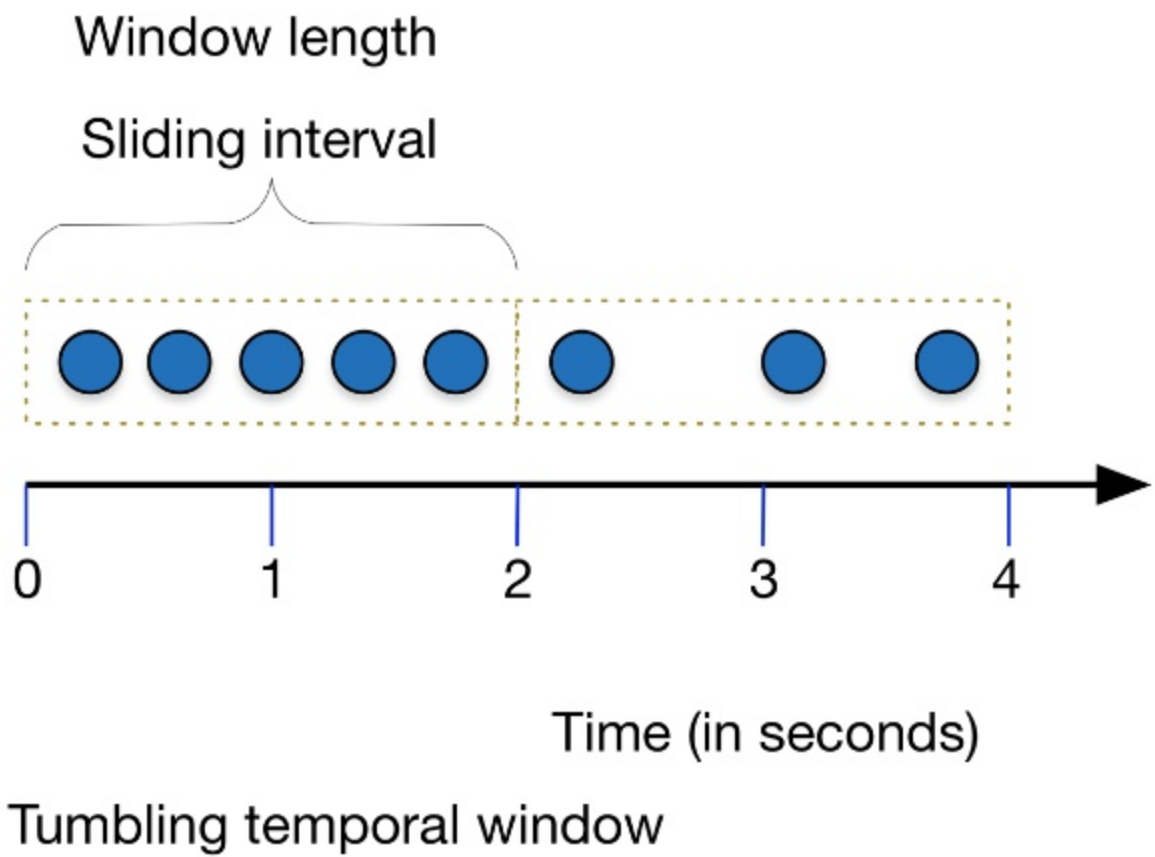
Windows

HORTONWORKS

Tumbling Windows

HORTONWORKS

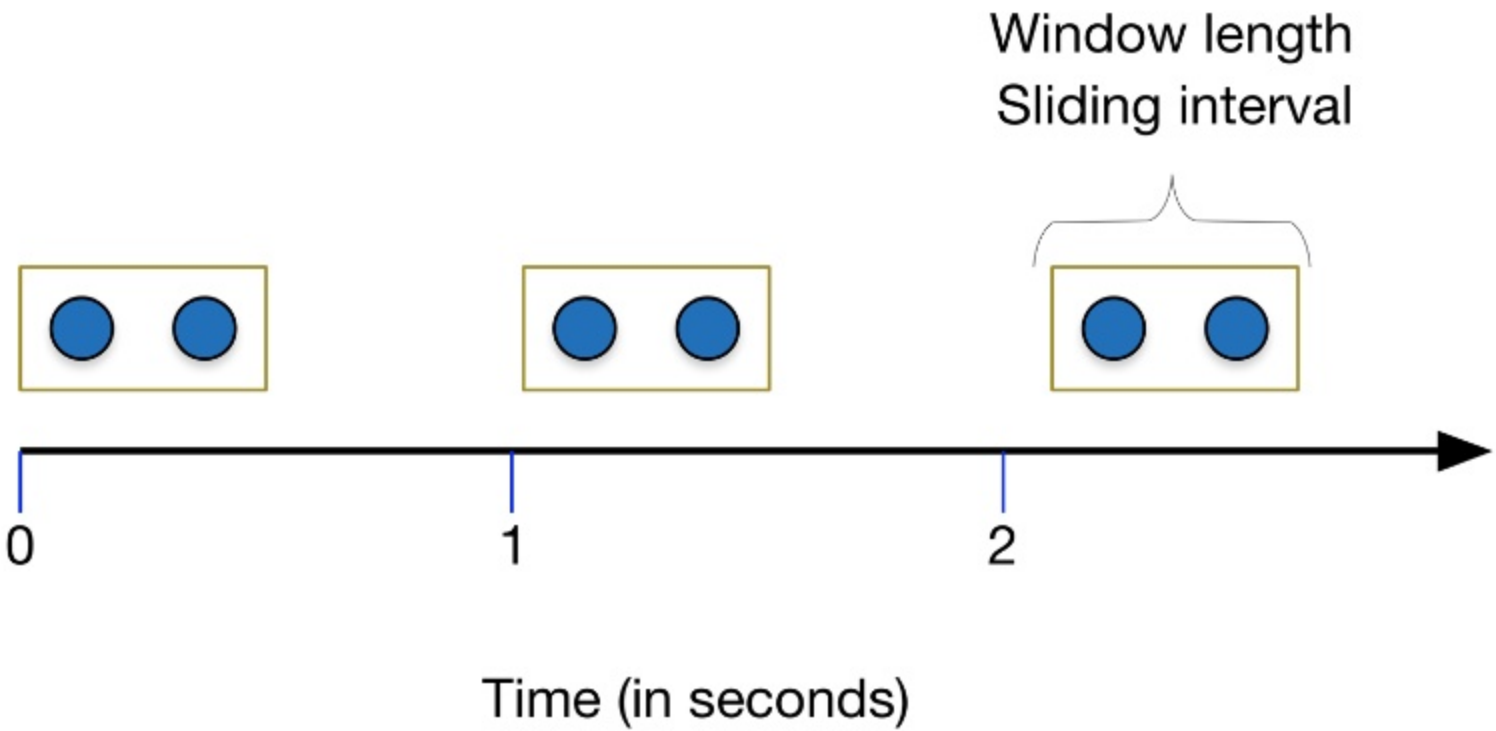
Tumbling Time Windowing



Tumbling temporal window

HORTONWORKS

Tumbling Count Windowing



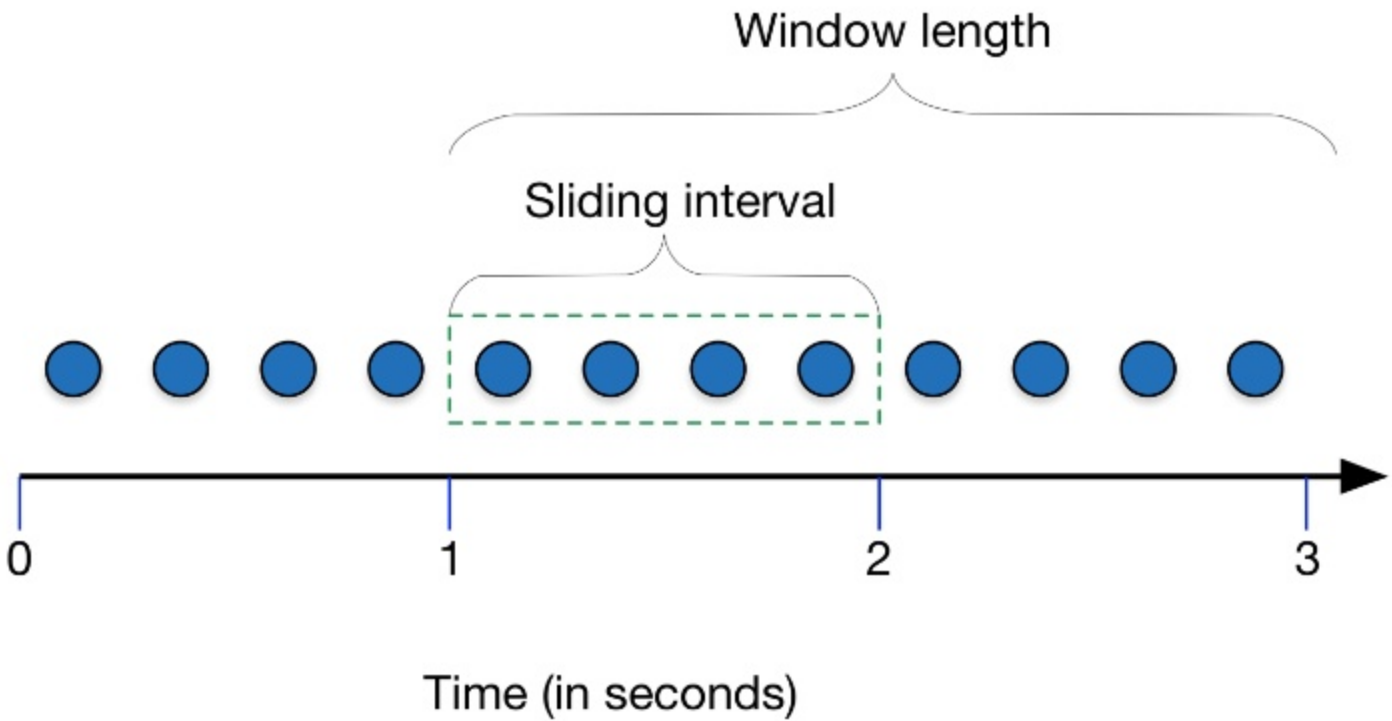
Tumbling count-window

HORTONWORKS

Sliding Windows

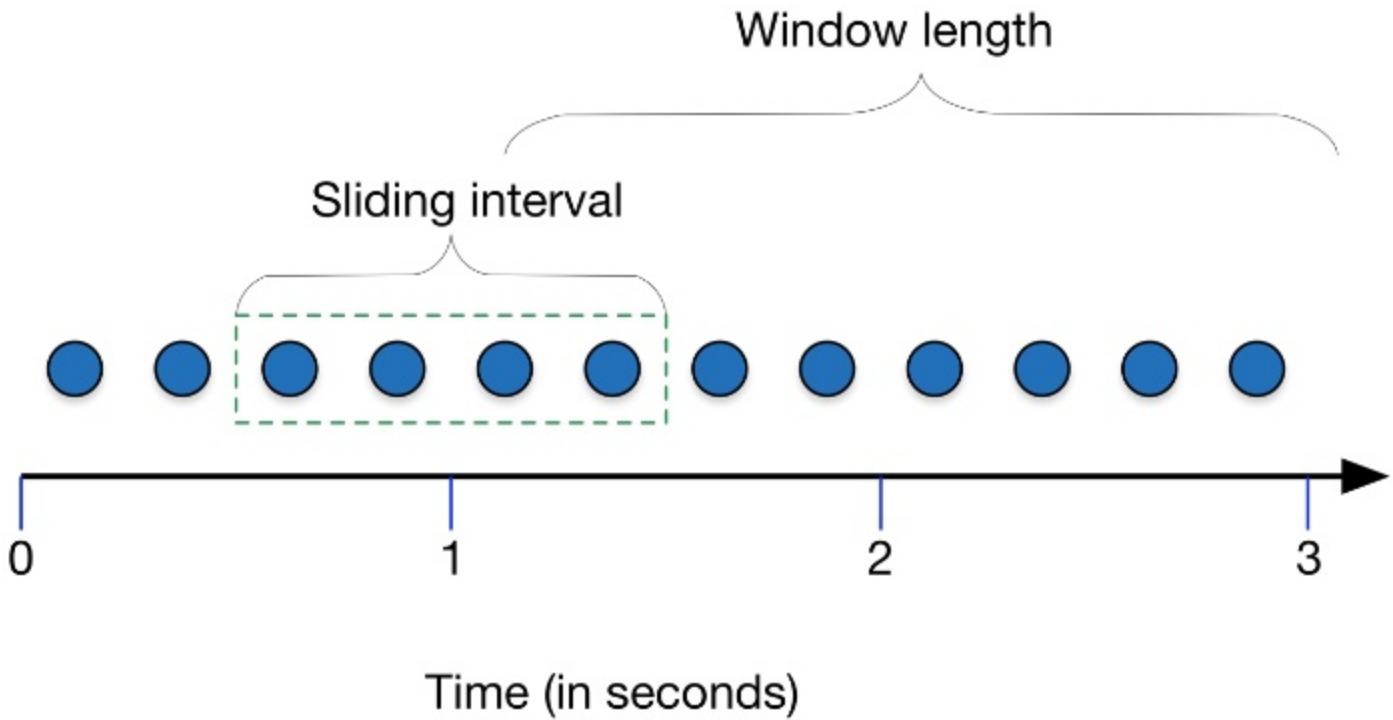
HORTONWORKS

Sliding Time Window



HORTONWORKS

Sliding Count Window



HORTONWORKS

Watermarking

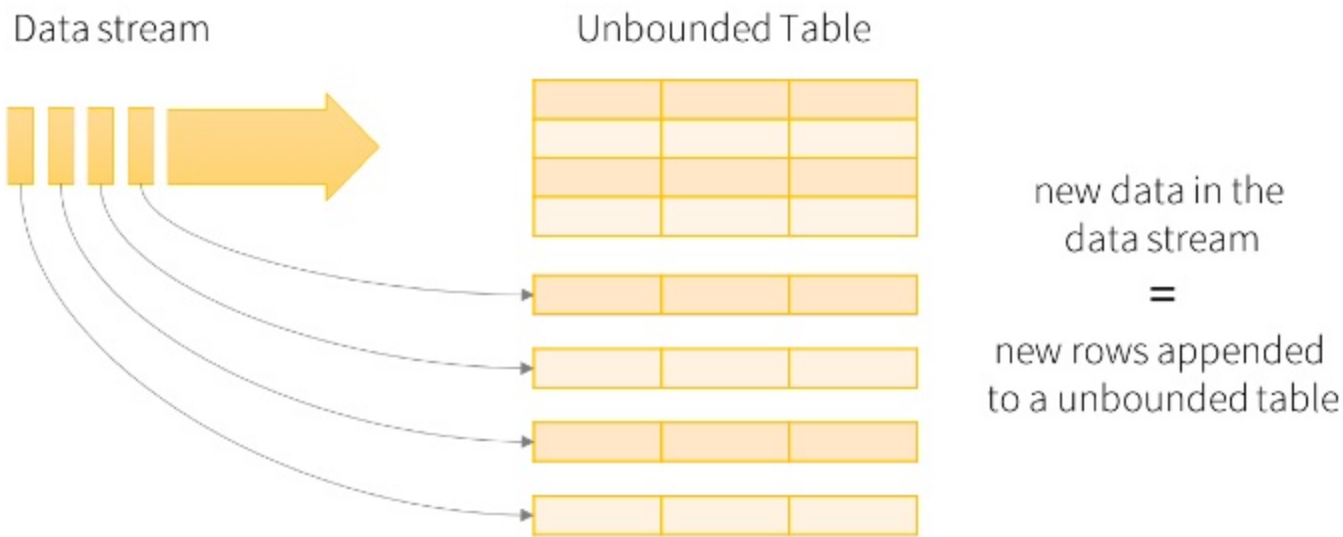
- The threshold of how late data is expected to be and when to drop old state
- Trails behind max event time seen by the Spark engine
- Easiest to think of the Watermark delay as the trailing gap

HORTONWORKS

Watermarking

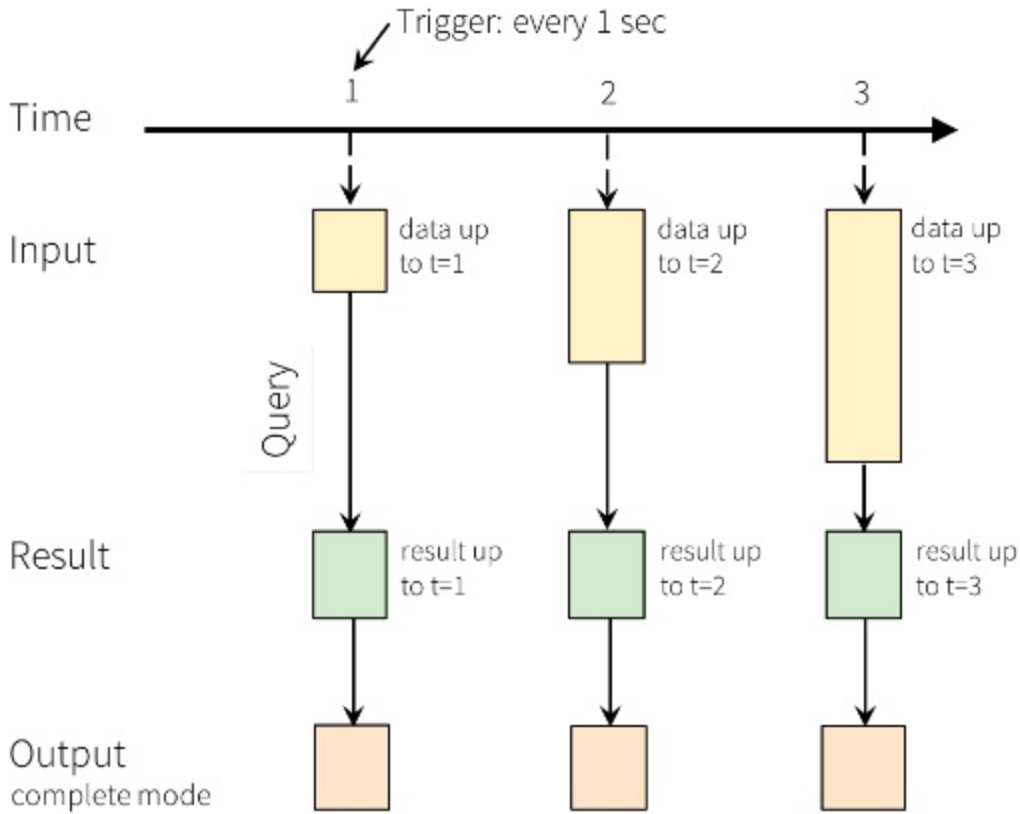
- Data that is newer than the watermark is late, but allowed to aggregate
- Data that is older than the watermark is dropped as it’s "too late"
- Any windows older than watermark are automatically deleted to limit state

HORTONWORKS



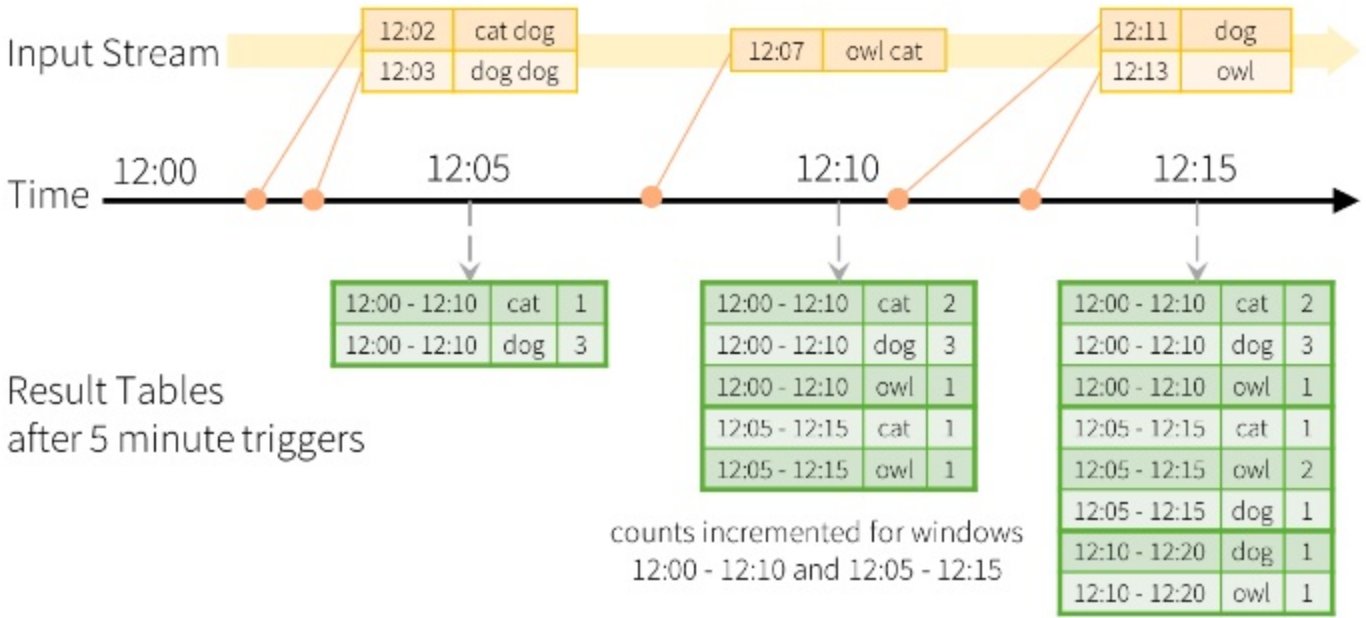
Data stream as an unbounded table

Source: http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



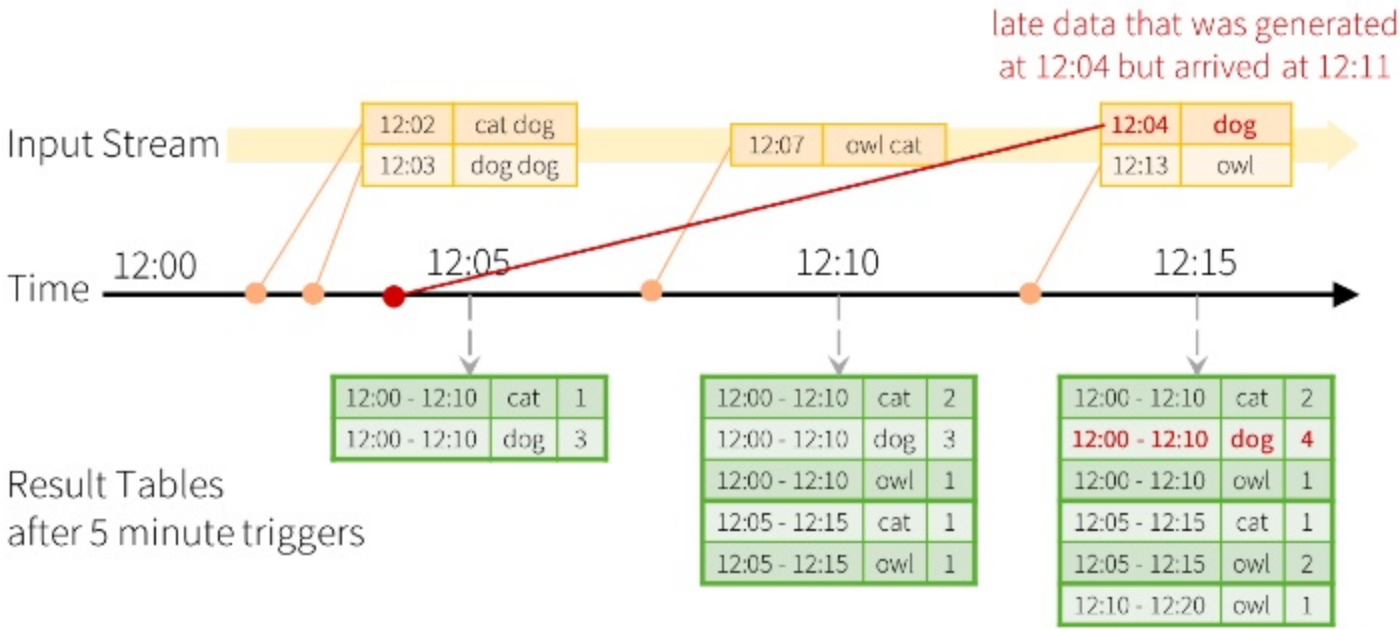
Programming Model for Structured Streaming

Source: http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



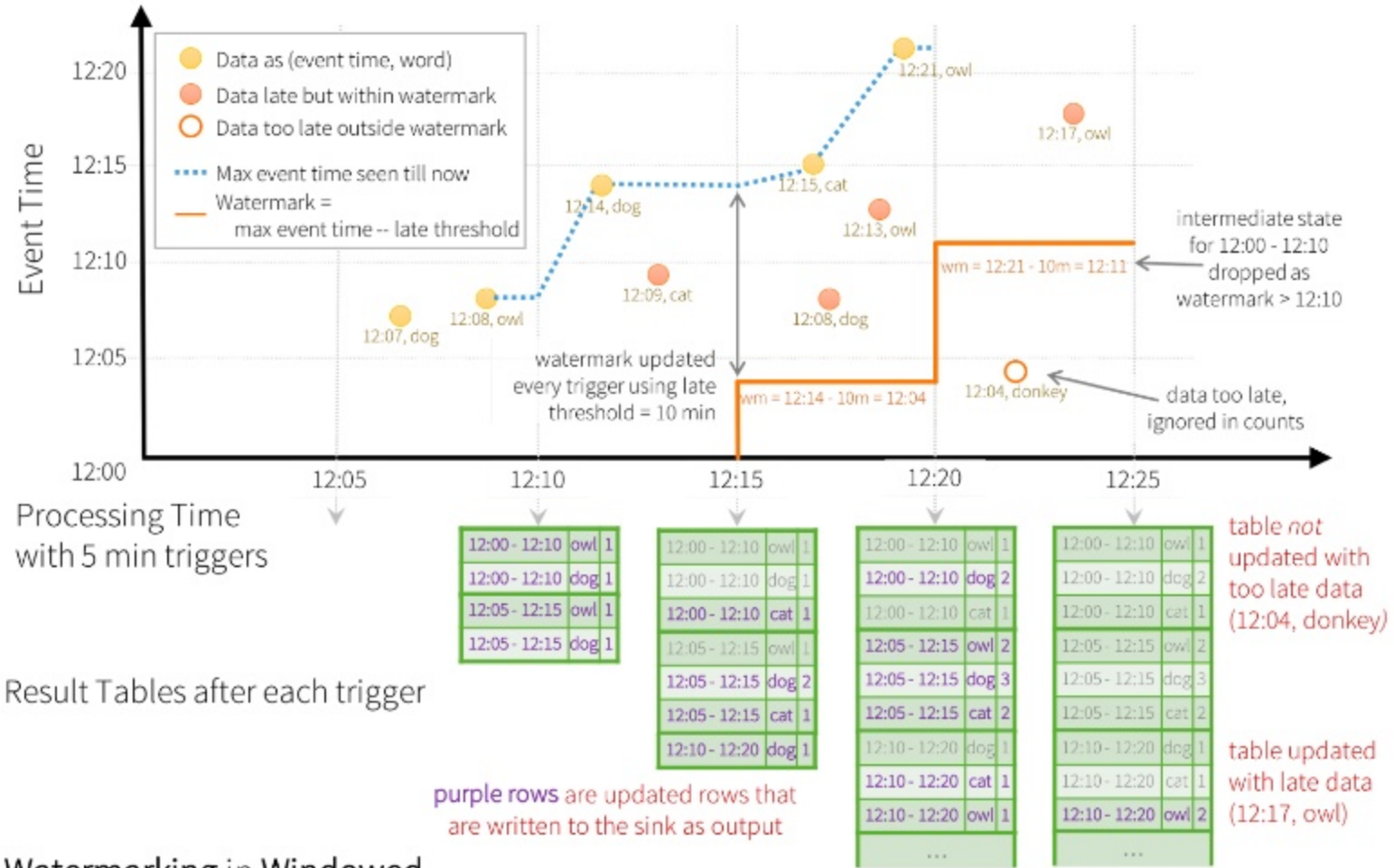
Windowed Grouped Aggregation
with 10 min windows, sliding every 5 mins

Source: http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



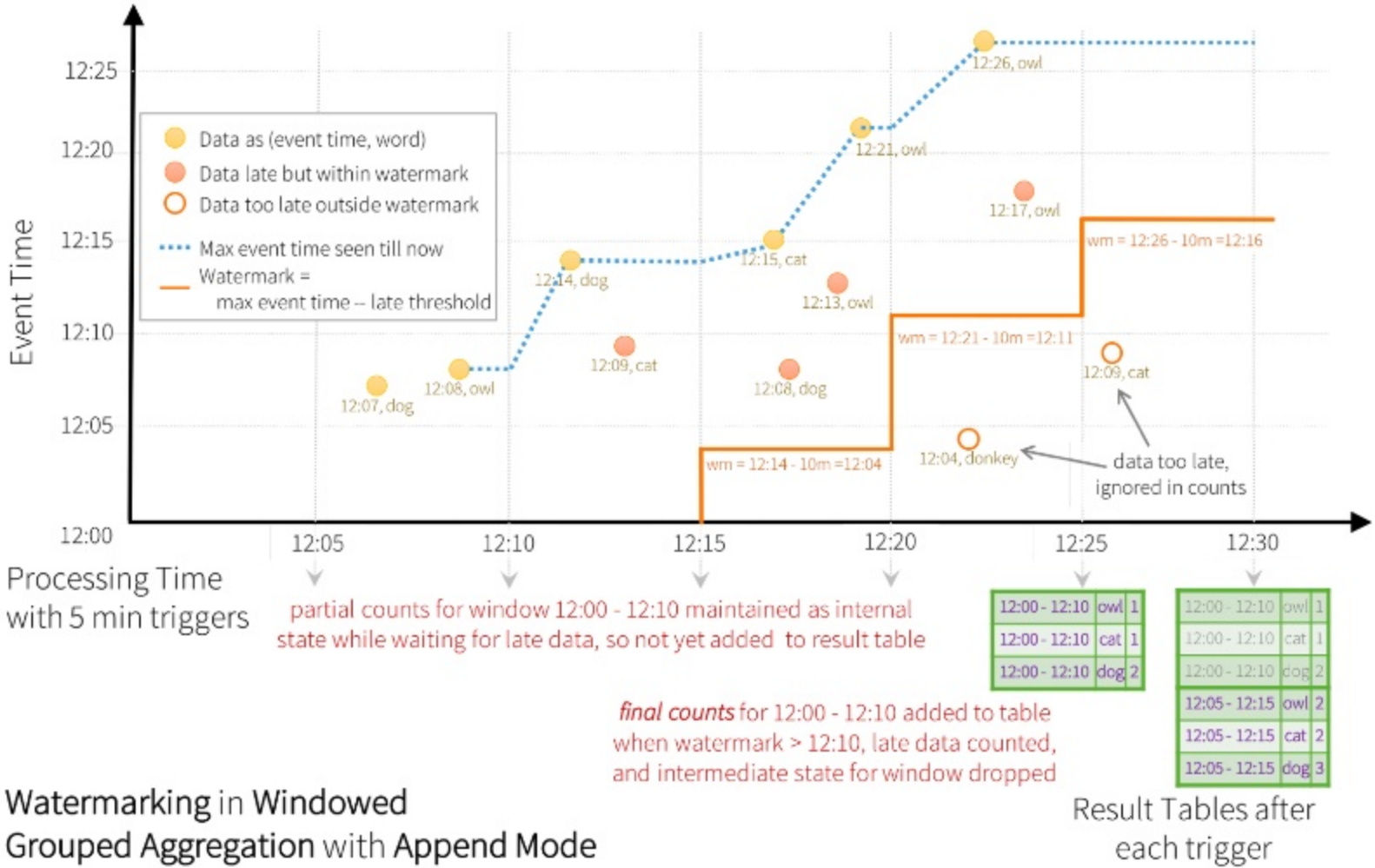
Late data handling in Windowed Grouped Aggregation

Source: http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



Watermarking in Windowed Grouped Aggregation with Update Mode

Source: http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



Watermarking in Windowed Grouped Aggregation with Append Mode

Source: http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

HORTONWORKS


The Do’s and Don’t’s
HORTONWORKS

The Good, the Bad, the Ugly

- Use NiFi for all ingestion and data preparation
- Be very cautious of forcing order in Kafka
- Chose your data store to match your query pattern

HORTONWORKS

The Good, the Bad, the Ugly

- **Monitor, monitor, monitor**
- Think about using schemas
- Plan for spike in traffic – this impacts Kafka portioning and consumers

HORTONWORKS


Questions?
HORTONWORKS


HORTONWORKS®
Thank you