Apache Gearpump

Lightweight Real-time Streaming Engine

About me

- Software Engineer at Intel Big Data Team
- Apache Gearpump committer, awesome-streaming
- Previously MapReduce NativeTask, storm-benchmark
- Shanghai Big Data Streaming Meetup

History of Gearpump

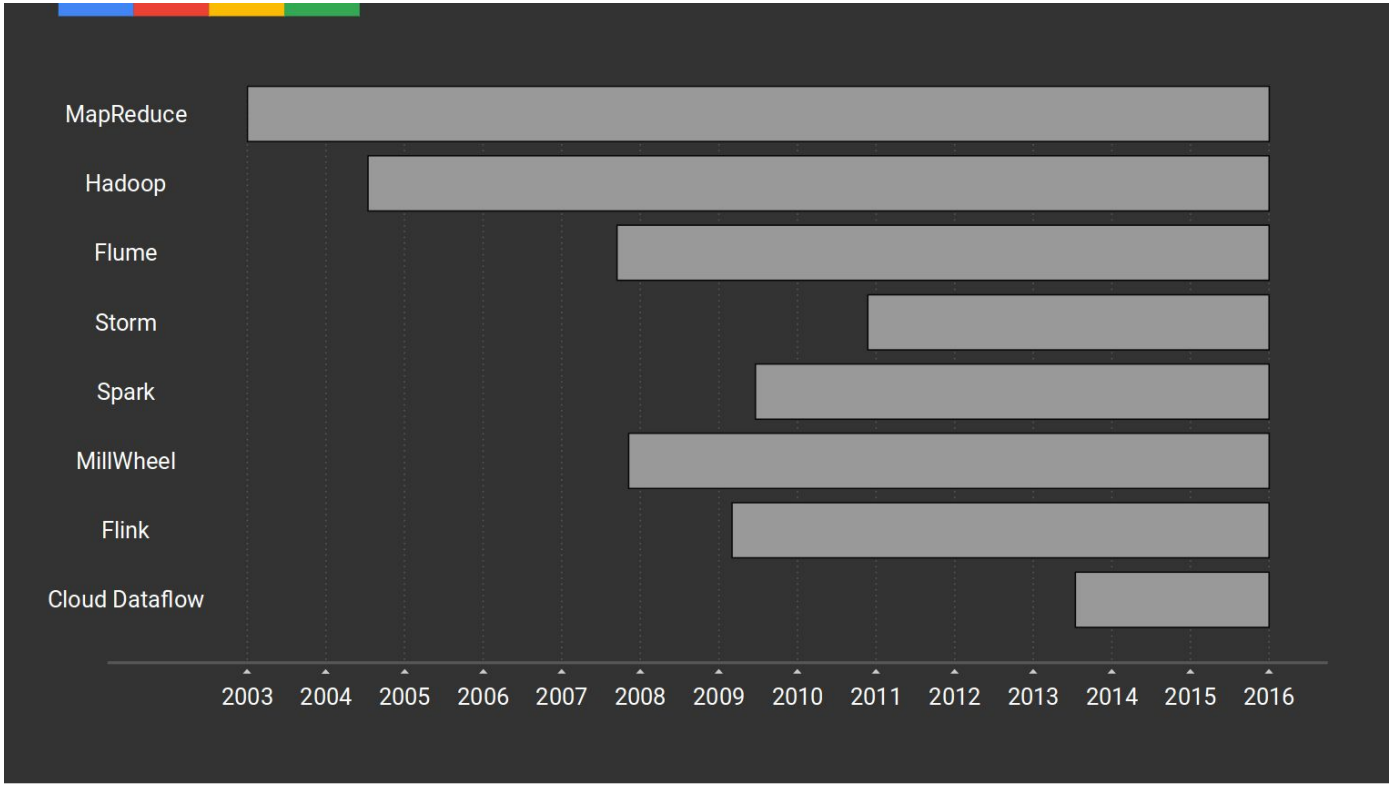
- Conceived at Intel in mid-2014
- Open source project on GitHub from start
- Entered Apache incubation on Mar.8th, 2016
- Current stable release 0.8.0

"The name Gearpump is a reference to the engineering term "Gear Pump", which is a super simple pump that consists of only two gears, but is very powerful at streaming water."

Yet Another Streaming Engine ?

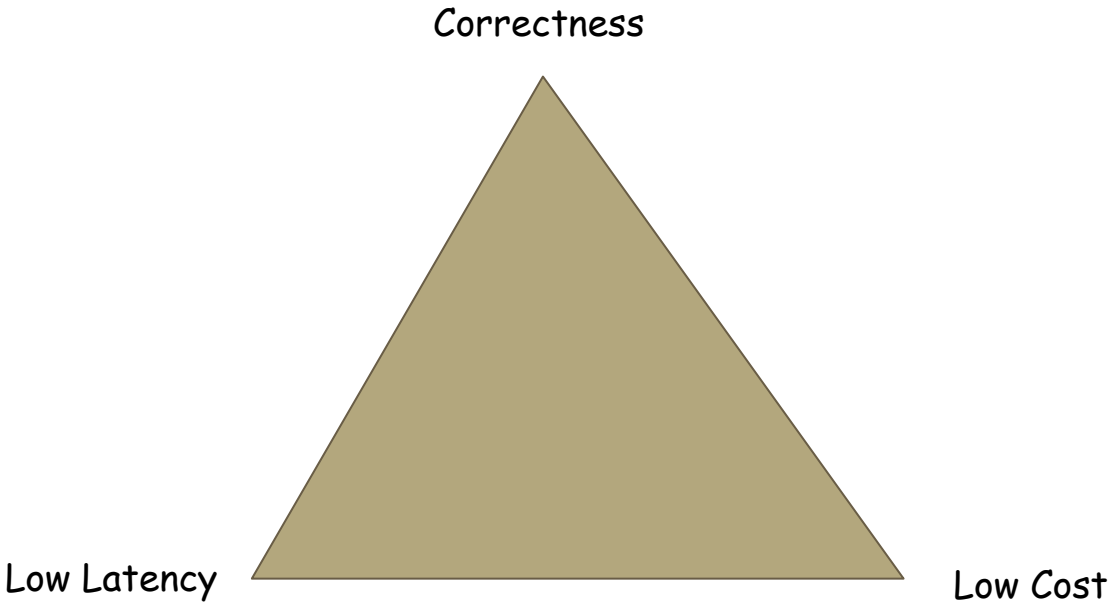
	Flume	NiFi	Gearpump	Apex	Kafka Streams	Spark Streaming	Storm	Storm + Trident	Samza	Flink	Ignite Streaming	Beam (*GC DataFlow)
Current version	1.6.0	0.6.1	incubating	3.3.0	0.9.0.1* (available in 0.10)	1.6.1	1.0.0	1.0.0	0.10.0	1.0.2	1.5.0	incubating
Category	DC/SEP	DC/SEP	SEP	DC/ESP	ESP	ESP	ESP/CEP	ESP/CEP	ESP	ESP/CEP	ESP/CEP	SDK
Event size	single	single	single	single	single	micro-batch	single	mini-batch	single	single	single	single
Available since [incubator since]	June 2012 [June 2011]	July 2015 [Nov 2014]	[Mar 2018]	Apr 2016 [Aug 2015]	Apr 2016 [July 2011]	Feb 2014 [2013]	Sep 2014 [Sep 2013]	Sep 2014 [Sep 2013]	Jan 2014 [July 2013]	Dec 2014 [Mar 2014]	Sep 2015 [Oct 2014]	[Feb 2016]
Contributors	26	67	19	53	160	938	207	207	48	159	56	80
Main backers	Apple Cloudera	Hortonworks	Intel Lightbend	Data Torrent	Confluent	AMPLab Databricks	Backtype Twitter	Backtype Twitter	LinkedIn	dataArtisans	GridGain	Google
Delivery guarantees	at least once	at least once	exactly once at least once (with non-fault-tolerant sources)	exactly once	at least once	exactly once at least once (with non-fault-tolerant sources)	at least once	exactly once	at least once	exactly once	at least once	exactly once*
State management	transactional updates	local and distributed snapshots	checkpoints	checkpoints	local and distributed snapshots	checkpoints	record acknowledgements	record acknowledgements	local snapshots distributed snapshots [fault-tolerant]	distributed snapshots	checkpoints	transactional updates*
Fault tolerance	yes (with file channel only)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes*
Out-of-order processing	no	no	yes	no	yes	no	yes	yes	yes (but not within a single partition)	yes	yes	yes*
Event prioritization	no	yes	programmable	programmable	programmable	programmable	programmable	programmable	yes	programmable	programmable	programmable
Windowing	no	no	time-based	time-based	time-based	time-based	time-based count-based	time-based count-based	time-based	time-based count-based	time-based count-based	time-based
Back-pressure	no	yes	yes	yes	N/A	yes	yes	yes	yes	yes	yes	yes*
Primary abstraction	Event	FlowFile	Message	Tuple	KafkaStream	DStream	Tuple	TridentTuple	Message	DataStream	IgniteDataStreamer	PCollection
Data flow	agent	flow (process group)	streaming application	streaming application	process topology	application	topology	topology	job	streaming dataflow	job	pipeline
Latency	low	configurable	very low	very low	very low	medium	very low	medium	low	low (configurable)	very low	low*
Resource management	native	native	YARN	YARN	Any process manager (e.g. YARN, Mesos, Chef, Puppet, Salt, Kubernetes, ...)	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN	YARN Mesos	integrated*
Auto-scaling	no	no	no	yes	yes	yes	no	no	no	no	no	yes*
In-flight modifications	no	yes	yes	yes	yes	no	yes (for resources)	yes (for resources)	no	no	no	no
API	declarative	compositional	declarative	declarative	declarative	declarative	compositional	compositional	compositional	declarative	declarative	declarative
Primarily written in	Java	Java	Scala	Java	Java	Scala	Clojure	Java	Scala	Java	Java	Java
API languages	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	Scala Java Clojure Python Ruby	Java Python Scala	Java	Java Scala Python	Java .NET C++	Java*
Notable users	Meebo Sharethrough SimpleGeo	N/A	Intel Levi's Honeywell	Capital One GE Predix PubMatic	N/A	Kelkoo Localytics AsiaInfo Openable Fairmdata Guavus	Yahoo! Spotify Groupon Flipboard The Weather Channel Alibaba Baidu Yelp WebMD	Klout GumGum CrowdFlower	LinkedIn Netflix Intuit Uber	King Otto Group	GridGain	N/A

https://databaseline.wordpress.com/2016/03/12/an-overview-of-apache-streaming-technologies/



(source: The Evolution of Massive-Scale Data Processing, slide 4)

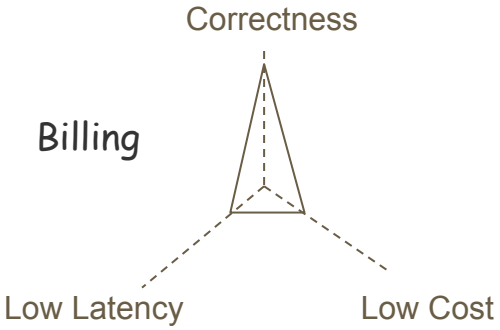
Data Processing Tradeoffs



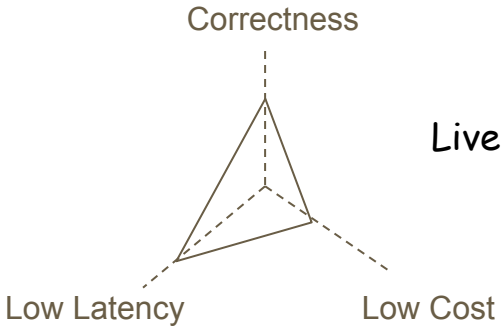
(source: The Beam Model, slide 10)

Use case: charge advertisers

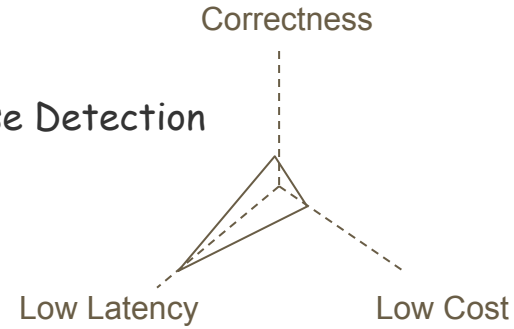
Billing



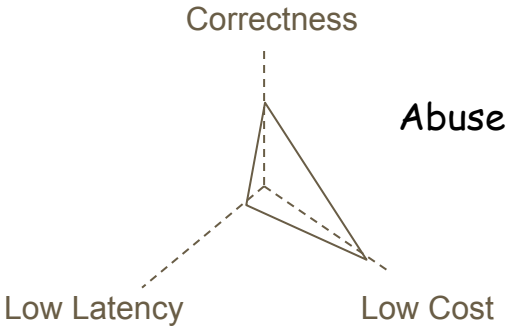
Live Cost Estimate



Abuse Detection



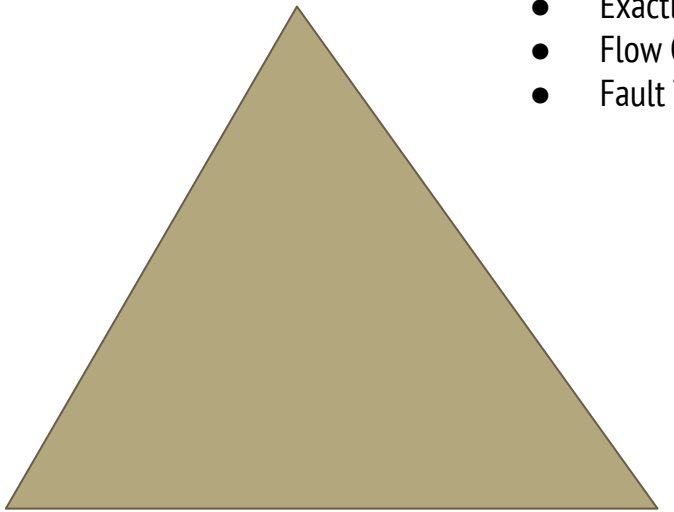
Abuse Detection Backfill



Gearpump

Correctness

- Out of order processing
- Exactly Once
- Flow Control
- Fault Tolerance



- Native Streaming
- Message Driven

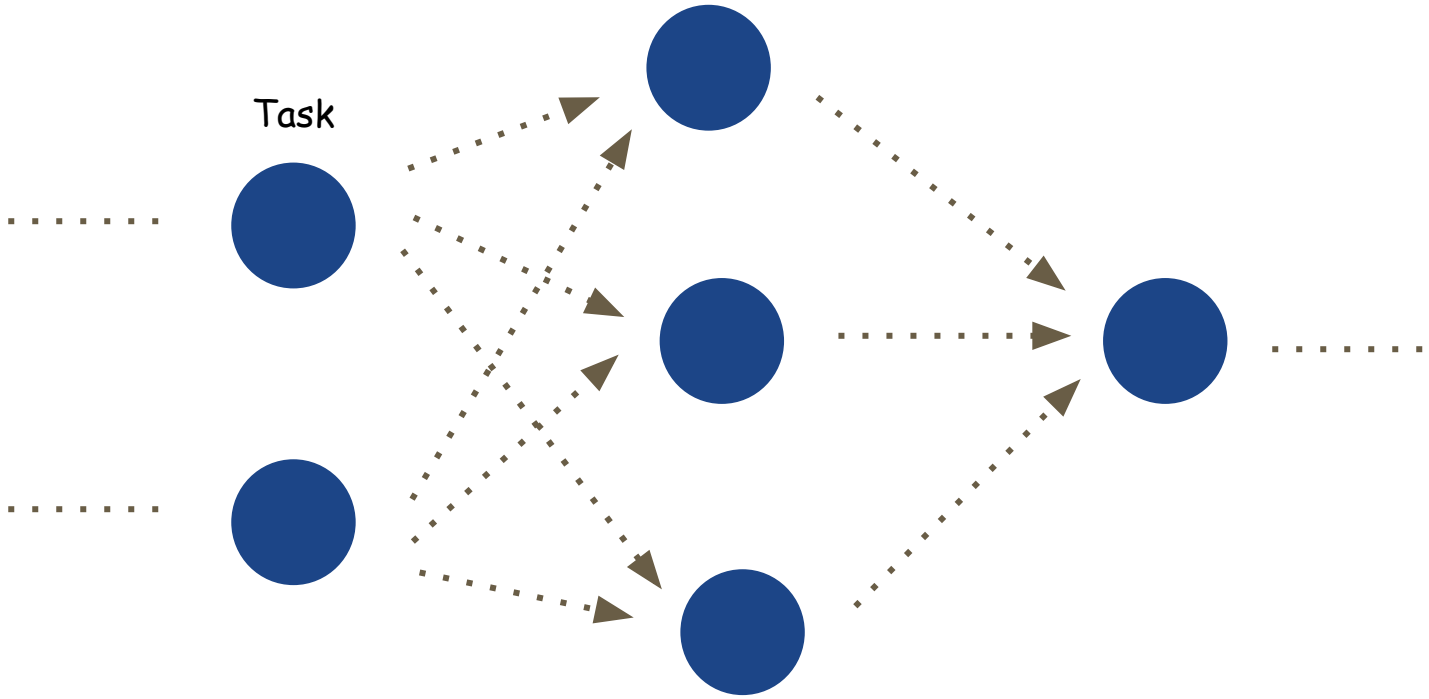
Low Latency

- Message Driven
- Dynamic DAG

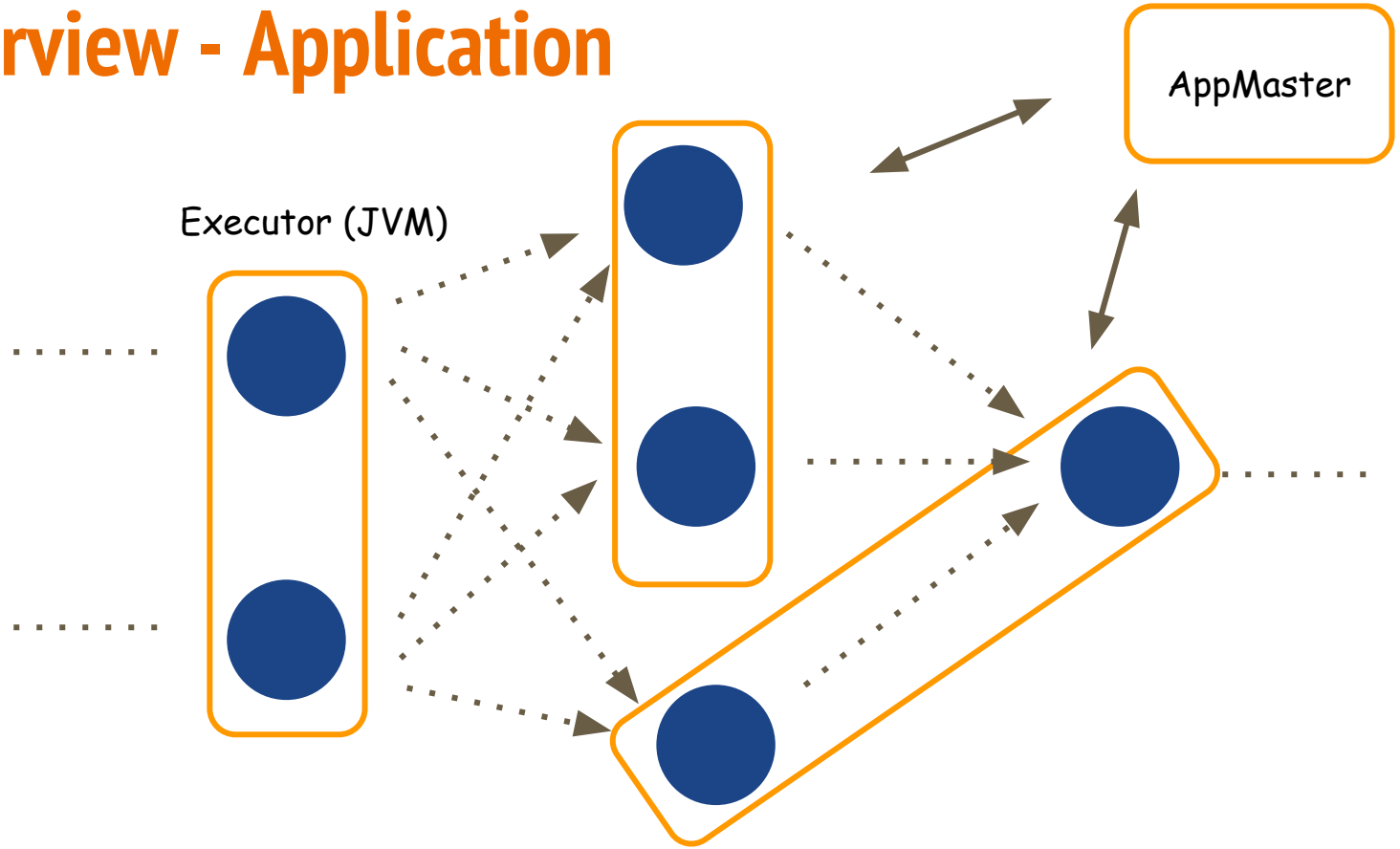
Low Cost

(source: The Beam Model, slide 10)

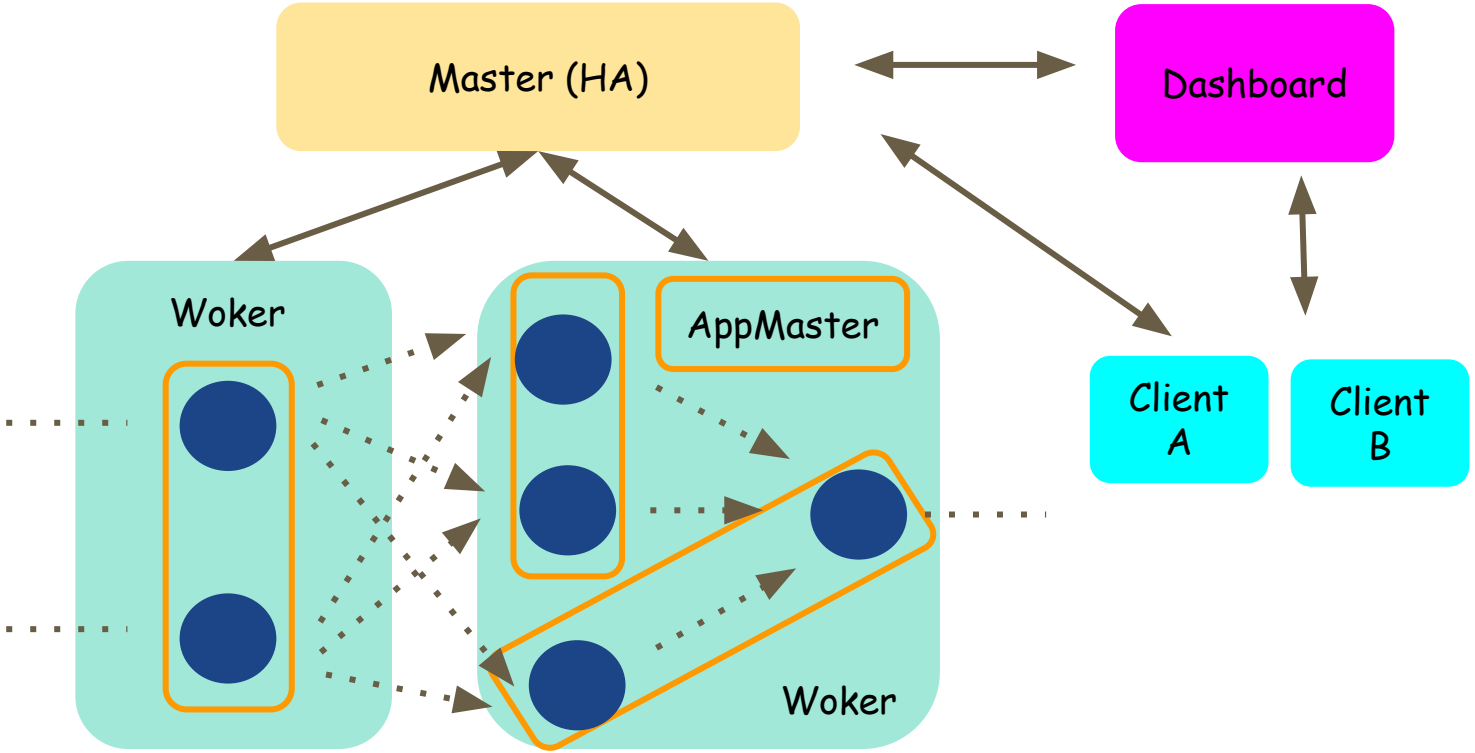
Overview - DAG



Overview - Application



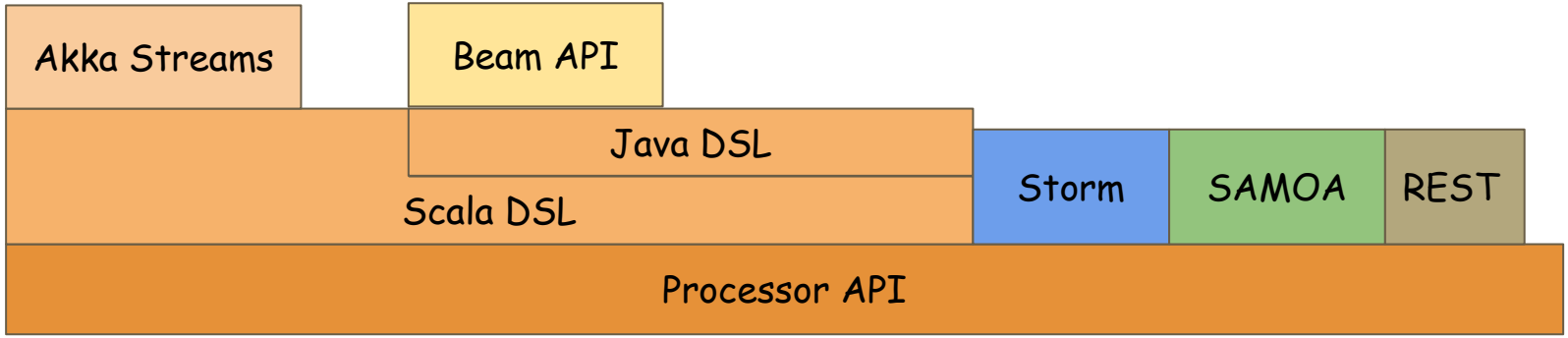
Overview - Cluster



Overview - Deployment

- Local mode
- Standalone mode
- YARN mode

Overview - API

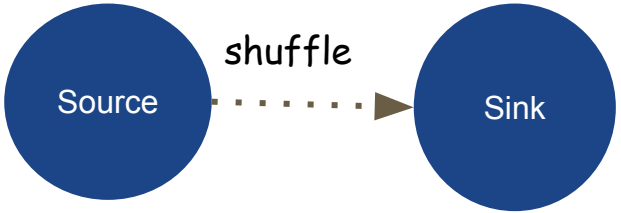


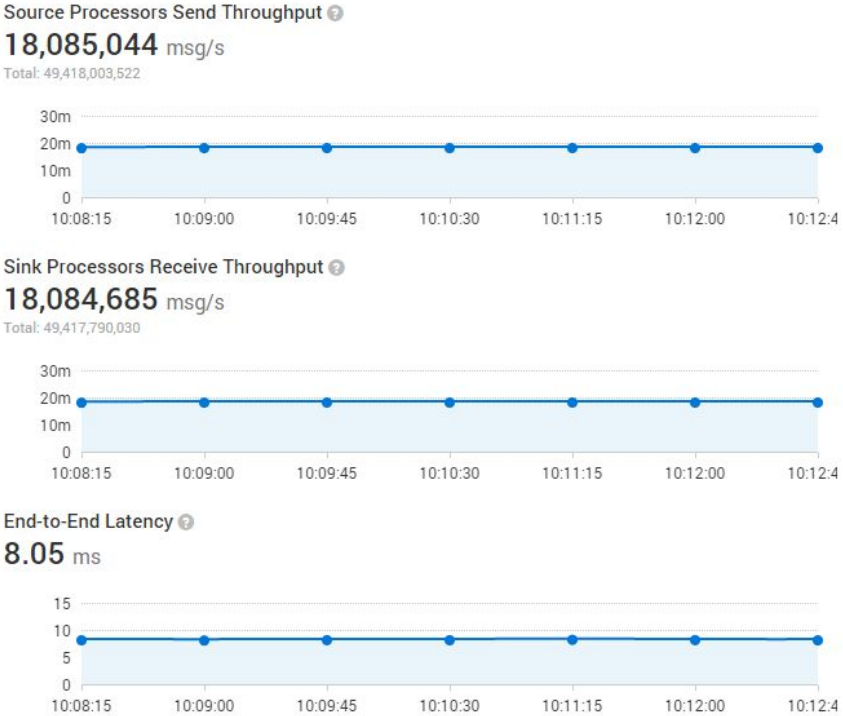
Storm compatibility

- Binary compatibility
- Dynamic DAG
- Support Storm 0.9 and 0.10

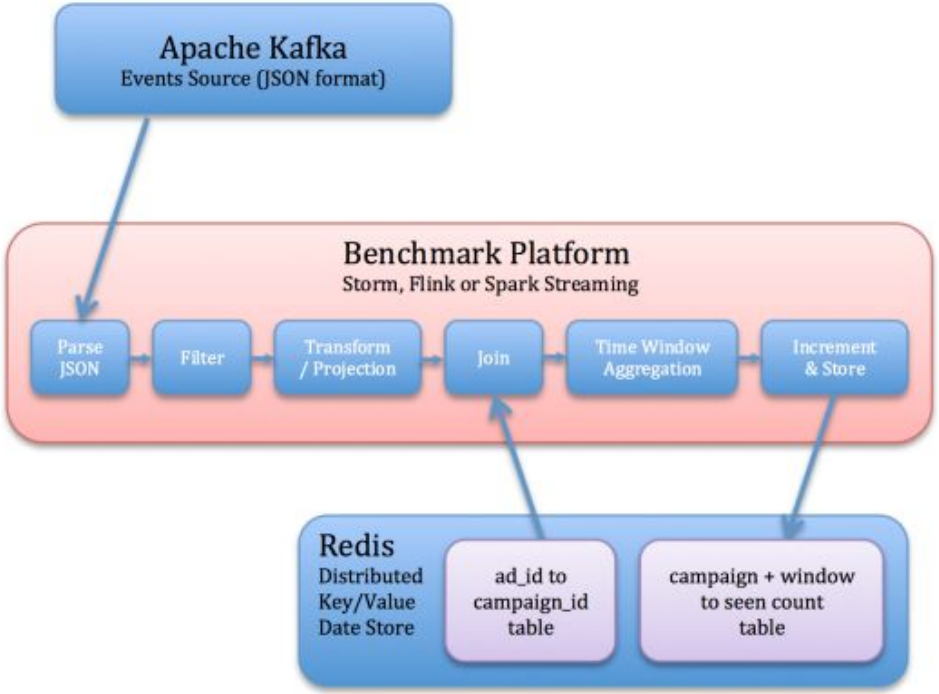
Overview - Performance

- 100 byte message
- 48-core, 256 GB memory, four node cluster



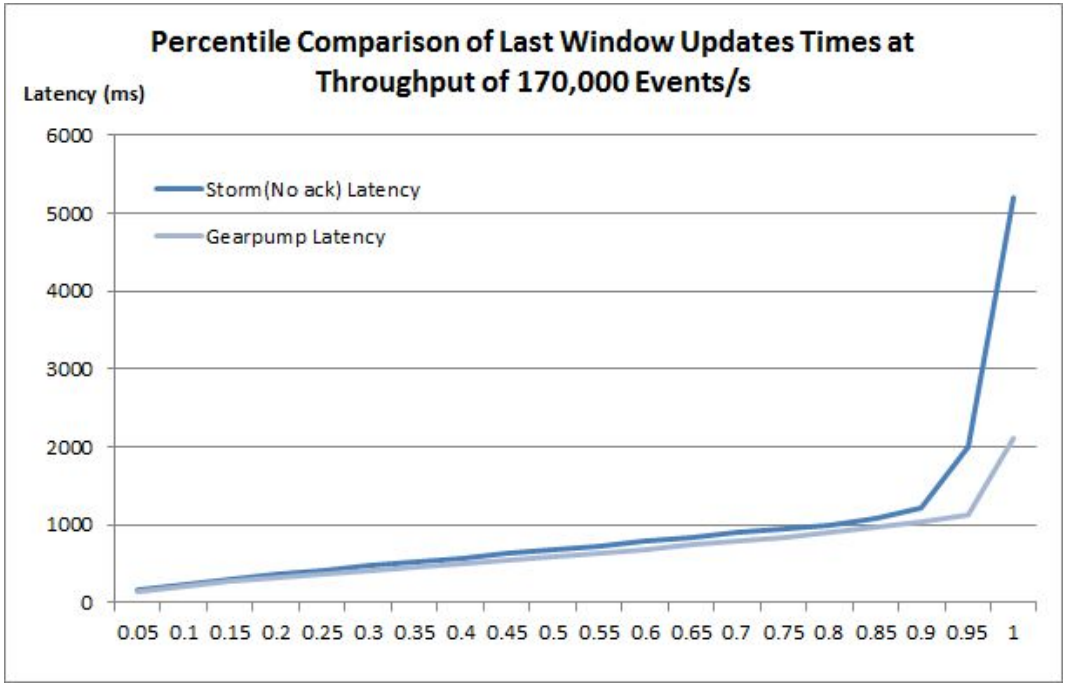


Yahoo Streaming Benchmarks



(source: benchmarking streaming computation engines at yahoo)

Yahoo Streaming Benchmarks



https://github.com/yahoo/streaming-benchmarks/pull/10

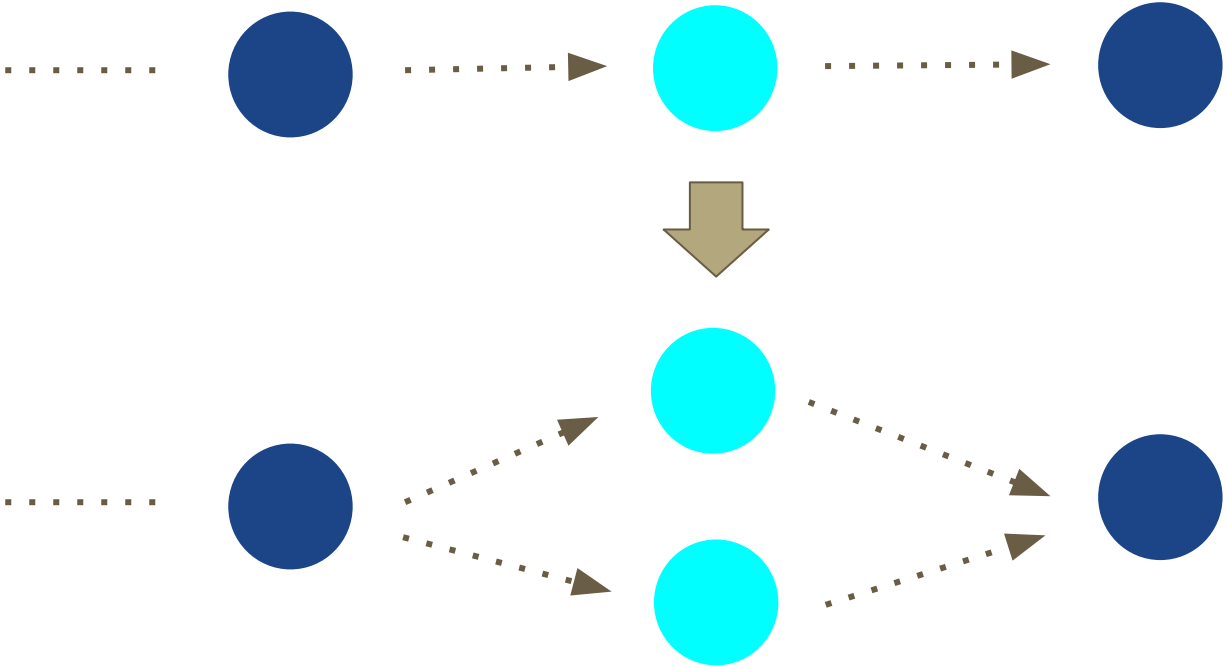
(source: https://www.flickr.com/photos/mike_lao/2588723972)

Demo

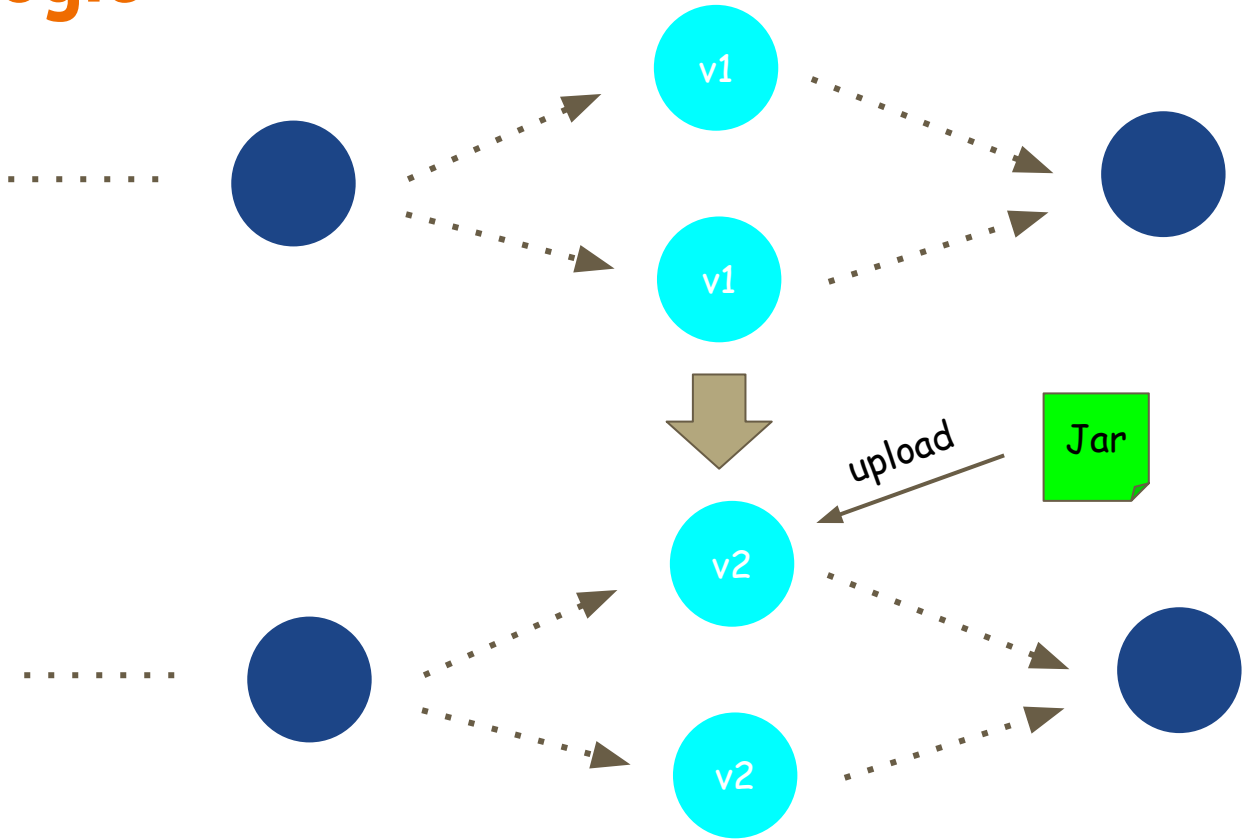
Dynamic DAG

Runtime DAG modification without restarting applications

Change parallelism

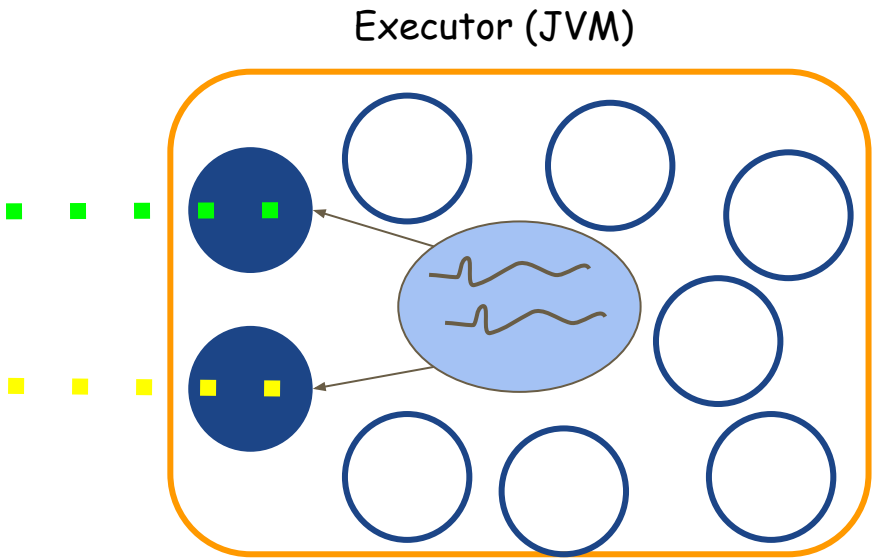


Change logic



Message Driven Processing

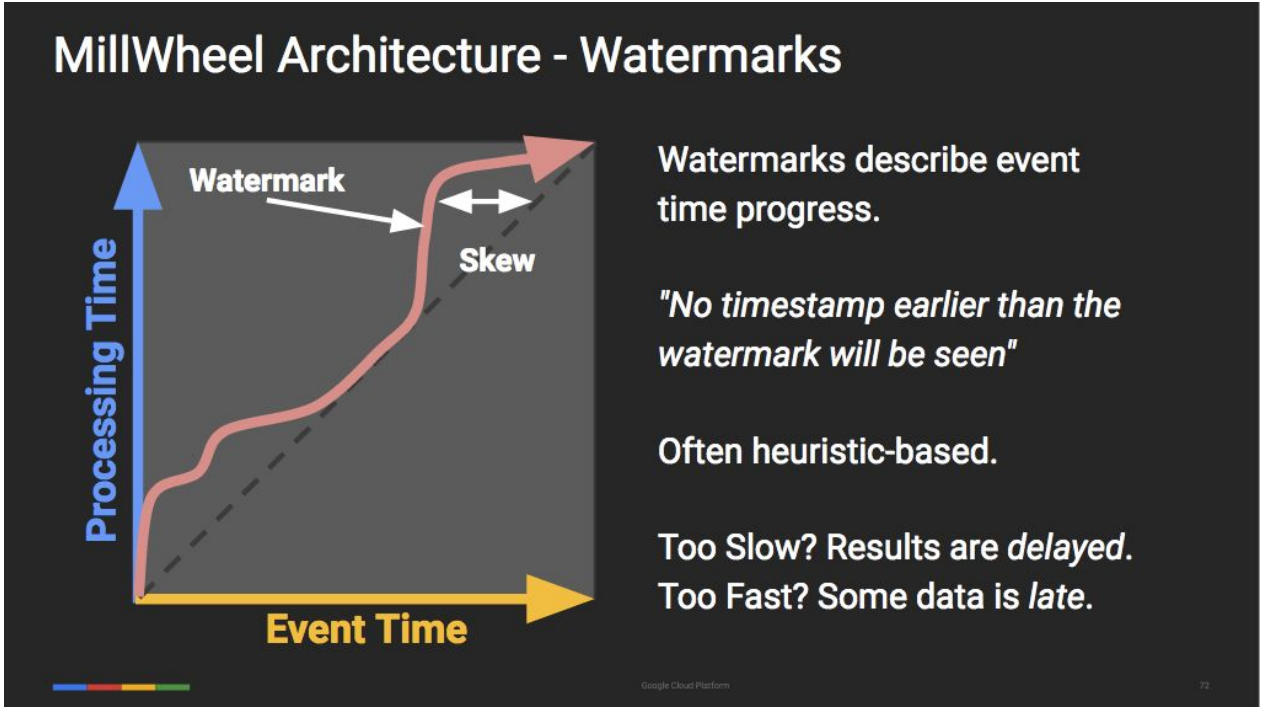
Message driven processing



- Task is thread safe
- Task is only taking up CPU on incoming messages
- Scale up to 10000 task on single four-core machine[1]

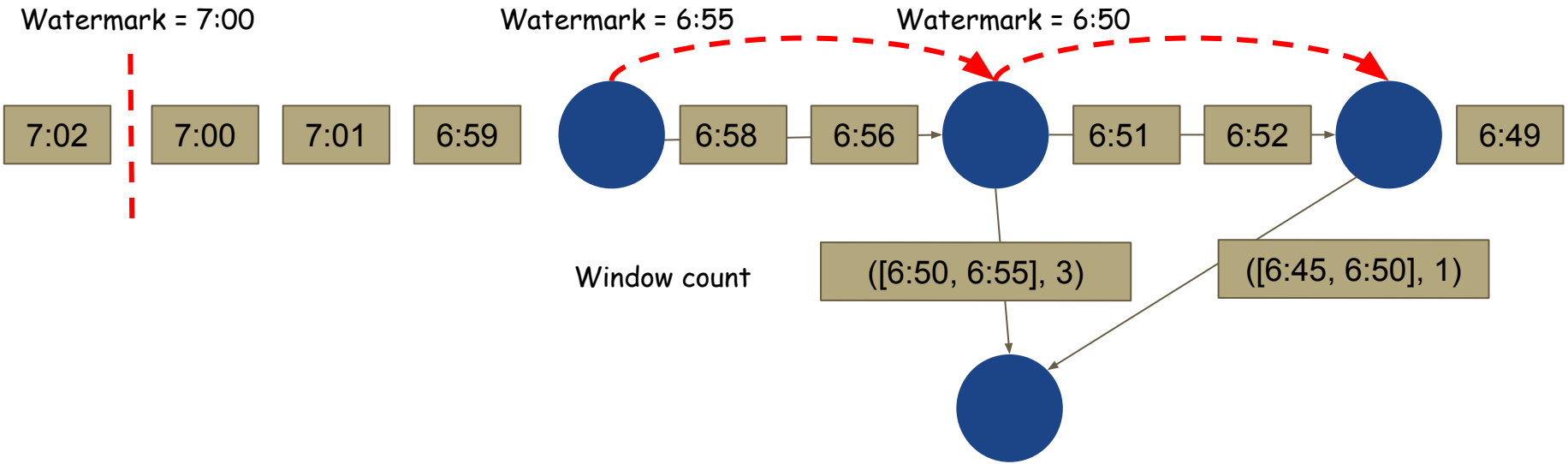
1. Gearpump Task is actually Akka Actor and it is reported ~2.5 million actors per GB of heap by Akka

Out of order processing



(source: The Evolution of Massive-Scale Data Processing, slide 72)

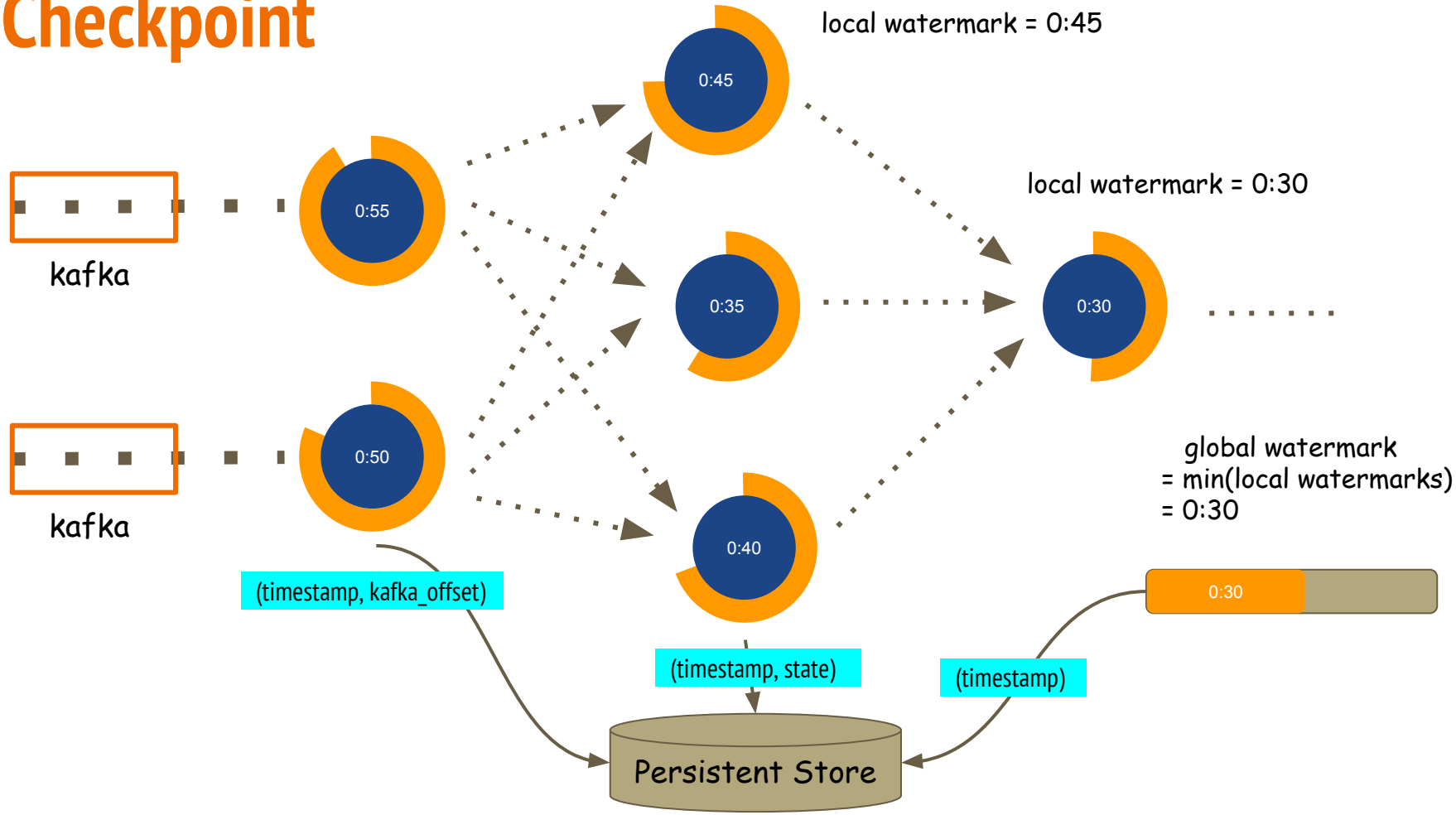
Event time based window count



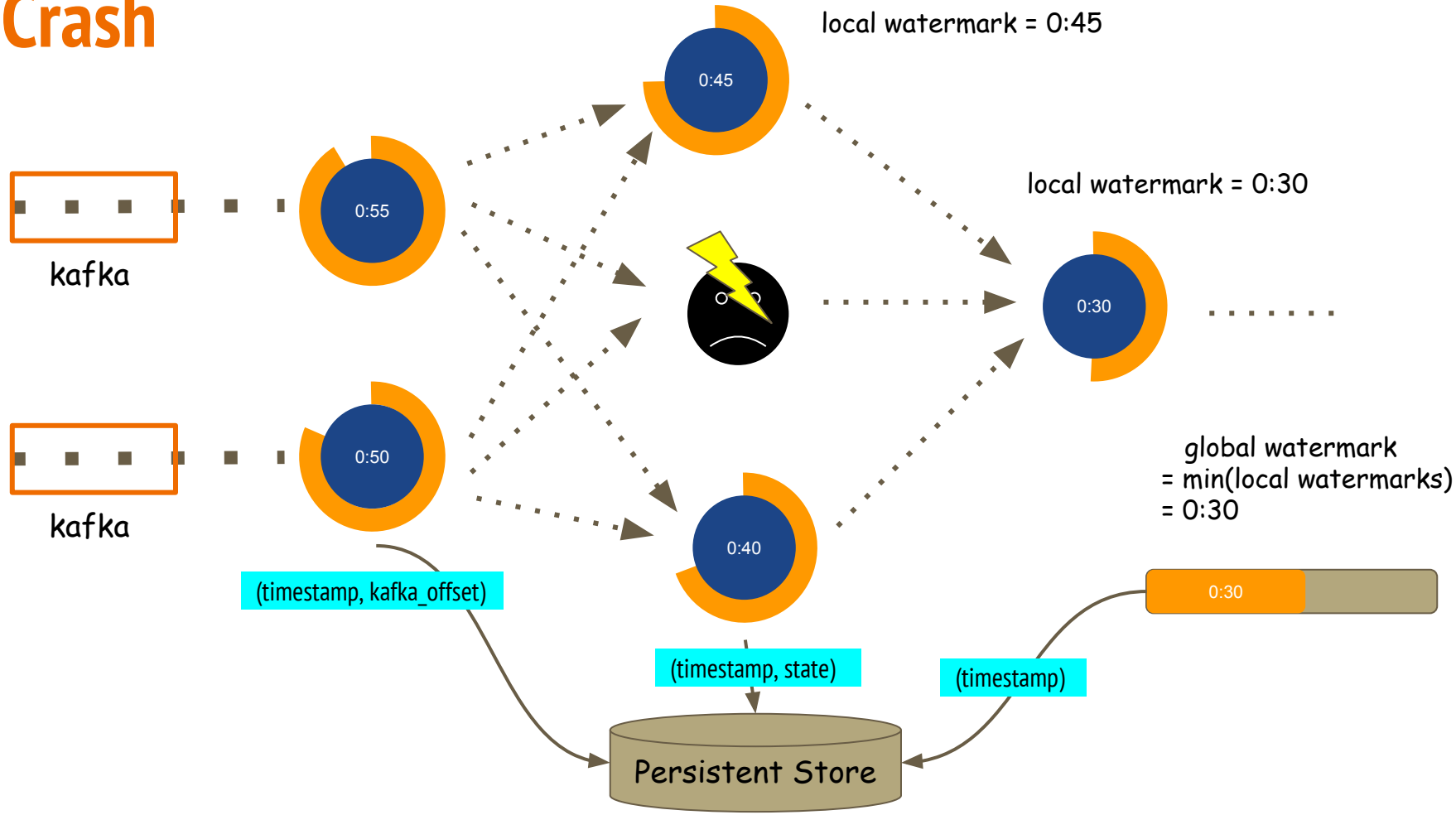
Exactly Once

No lost or duplicate updates to state

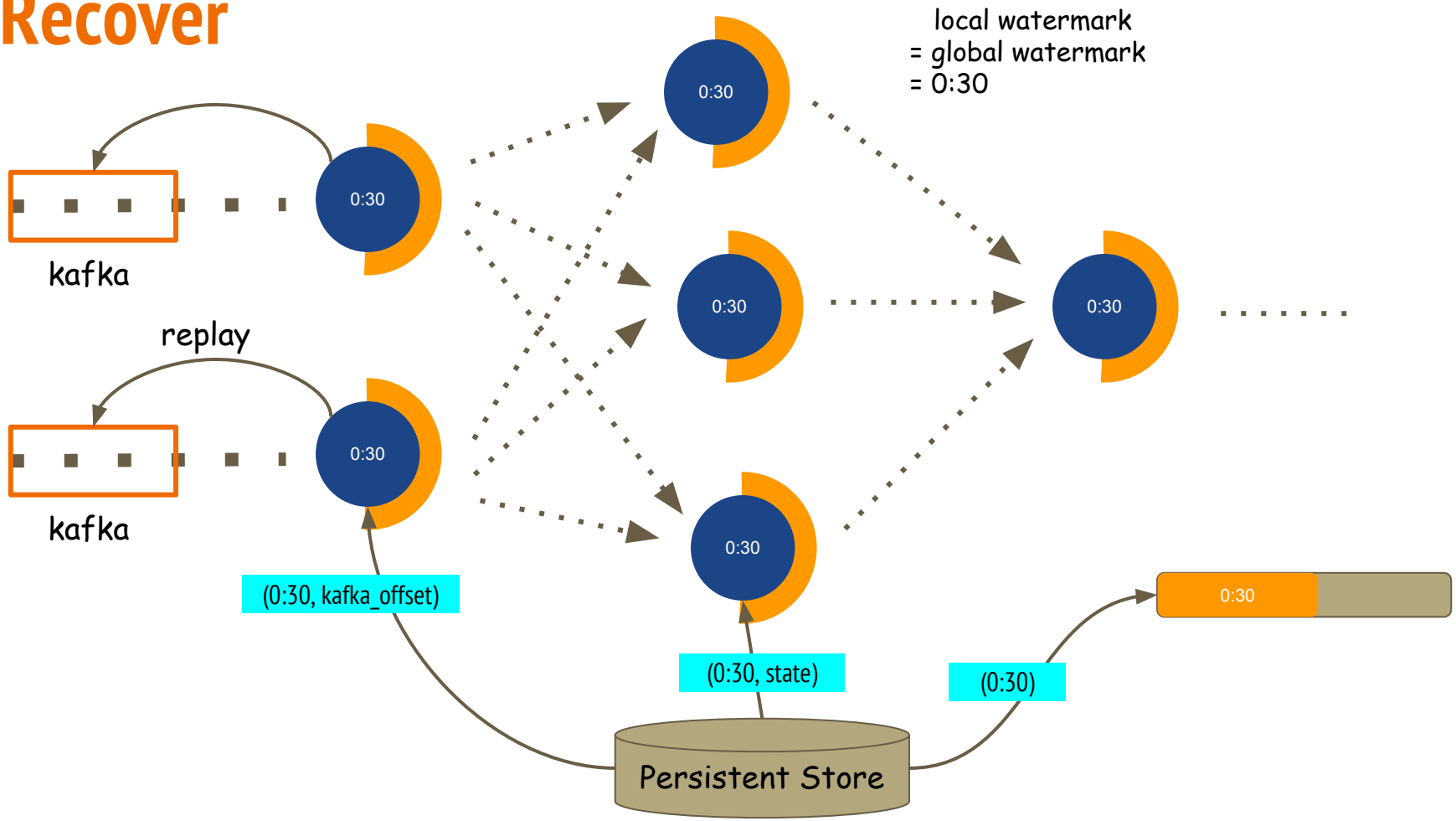
Checkpoint


local watermark = 0:45
0:45
local watermark = 0:30
0:55
kafka
0:35
0:30
0:50
kafka
global watermark
= min(local watermarks)
= 0:30
0:40
(timestamp, kafka_offset)
0:30
(timestamp, state)
(timestamp)
Persistent Store

Crash


local watermark = 0:45
0:45
local watermark = 0:30
0:55
kafka
0:30
0:50
kafka
global watermark
= min(local watermarks)
= 0:30
0:40
(timestamp, kafka_offset)
0:30
(timestamp, state)
(timestamp)
Persistent Store

Recover


local watermark
= global watermark
= 0:30
0:30
kafka
replay
kafka
(0:30, kafka_offset)
(0:30, state)
(0:30)
Persistent Store

Example

checkpoint

crash

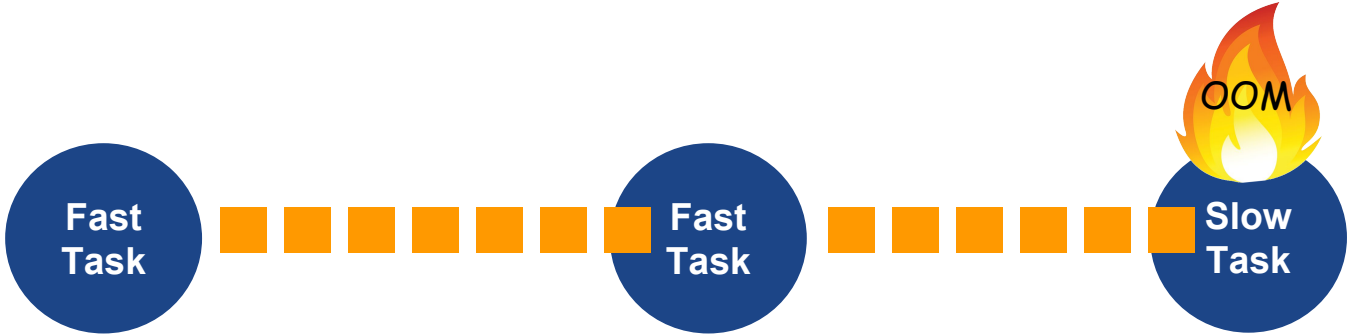
global watermark	(local watermark, kafka offset)	(local watermark, state)
0:00	(0:10, 1)	(0:00, 0)
0:10	(0:20, 2)	(0:10, 1)
0:20	(0:30, 3)	(0:20, 2)
0:30	(0:40, 4)	(0:30, 3)

Example

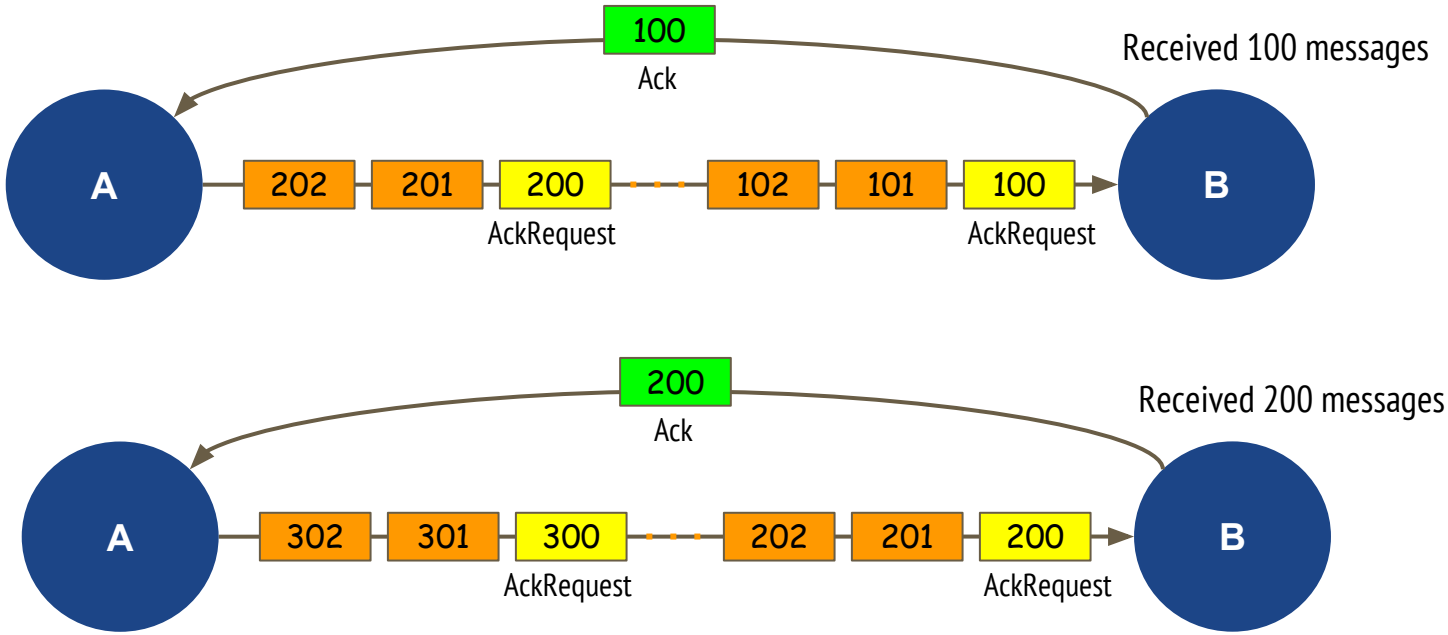
	global watermark	(local watermark, kafka offset)	(local watermark, state)
recover	0:30	(0:30, 3)	(0:30, 3)
checkpoint	0:30	(0:40, 4)	(0:30, 3)
⋮	0:40	(0:50, 5)	(0:40, 4)
▼	0:50	(1:00, 6)	(0:50, 5)

Flow Control

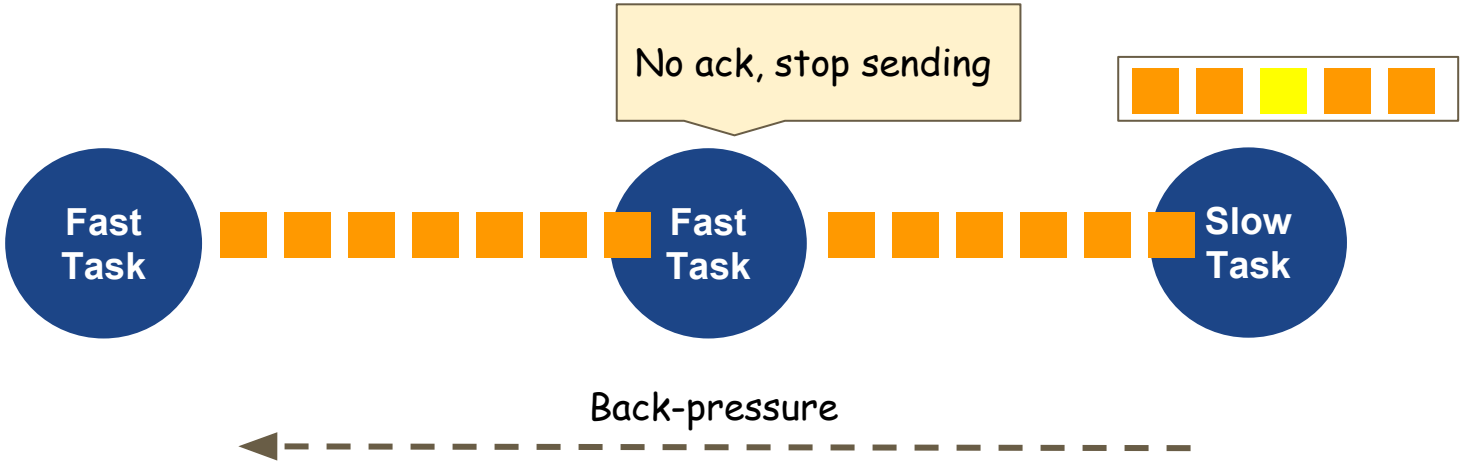
Without flow control



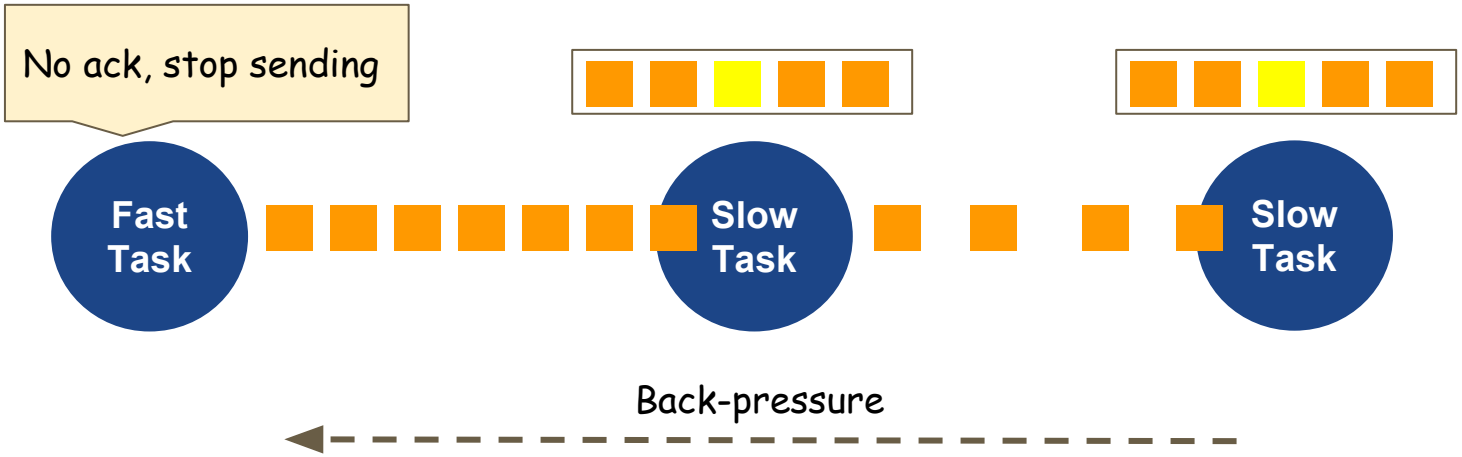
Performant message track



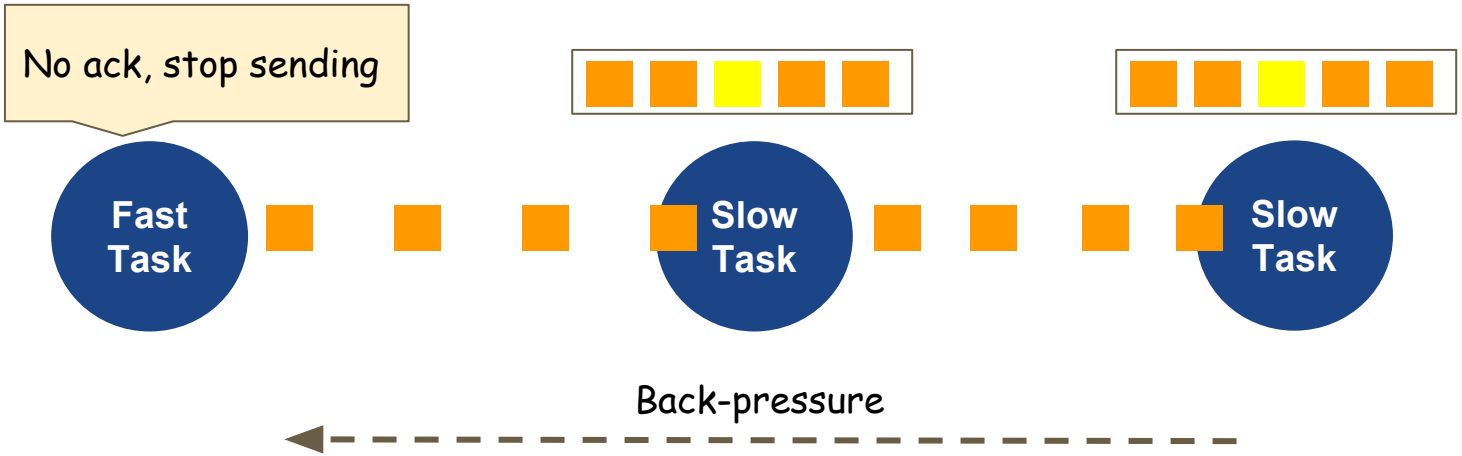
Flow control



Flow control



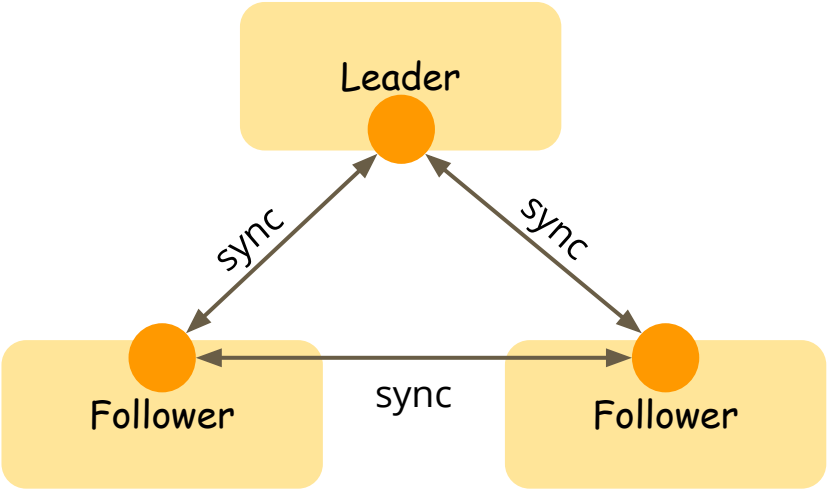
Flow control

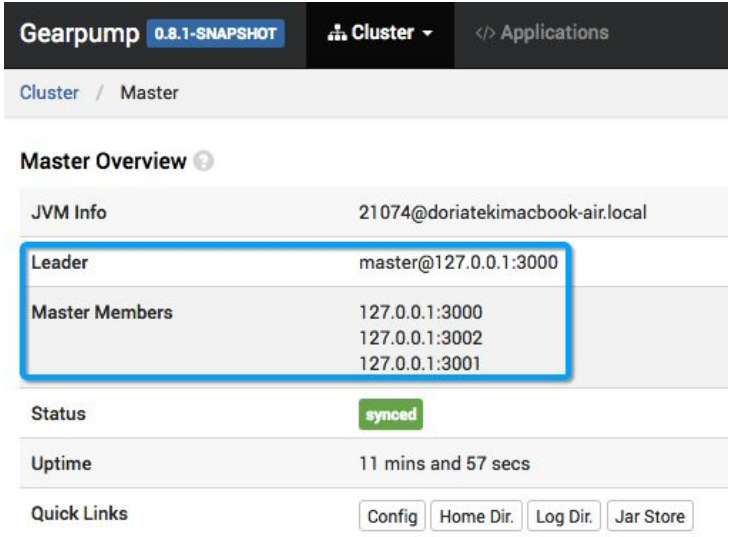


Fault Tolerance

Master HA

- Conflict-free Replicated Data Type (CRDT) for state consistency





Resource isolation

- Linux CGroup
- Configurable CPU resource per executor (JVM)
- Configurable executor number per application

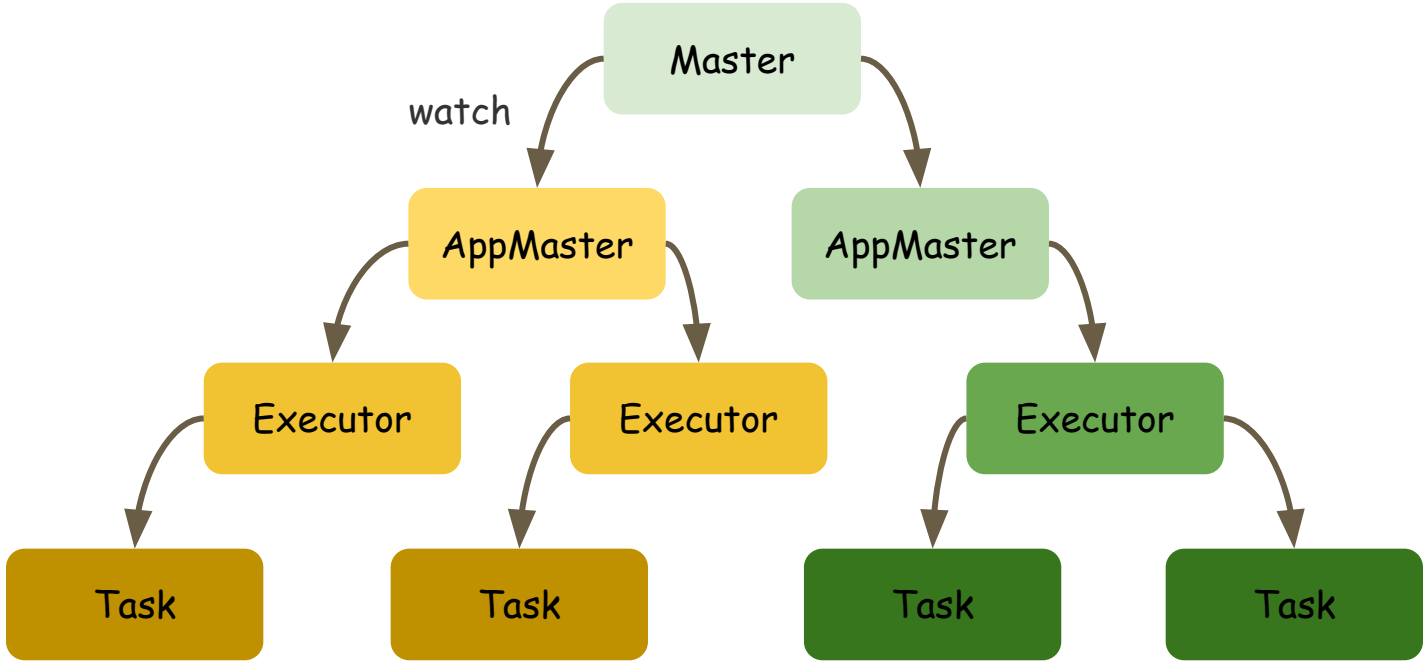
References

1. An Introduction to the Beam Model
2. The Evolution of Massive-Scale Data Processing
3. gearpump.apache.org
4. akka.io
5. http://www.slideshare.net/SeanZhong/strata-singapore-gearpumpreal-time-dagprocessing-with-akka-at-scale

Q & A

Backup slides

Supervisor hierarchy



Supervisor hierarchy