

CC5212-1

PROCESAMIENTO MASIVO DE DATOS

OTOÑO 2020

Lecture 6

Streaming: Kafka

Aidan Hogan

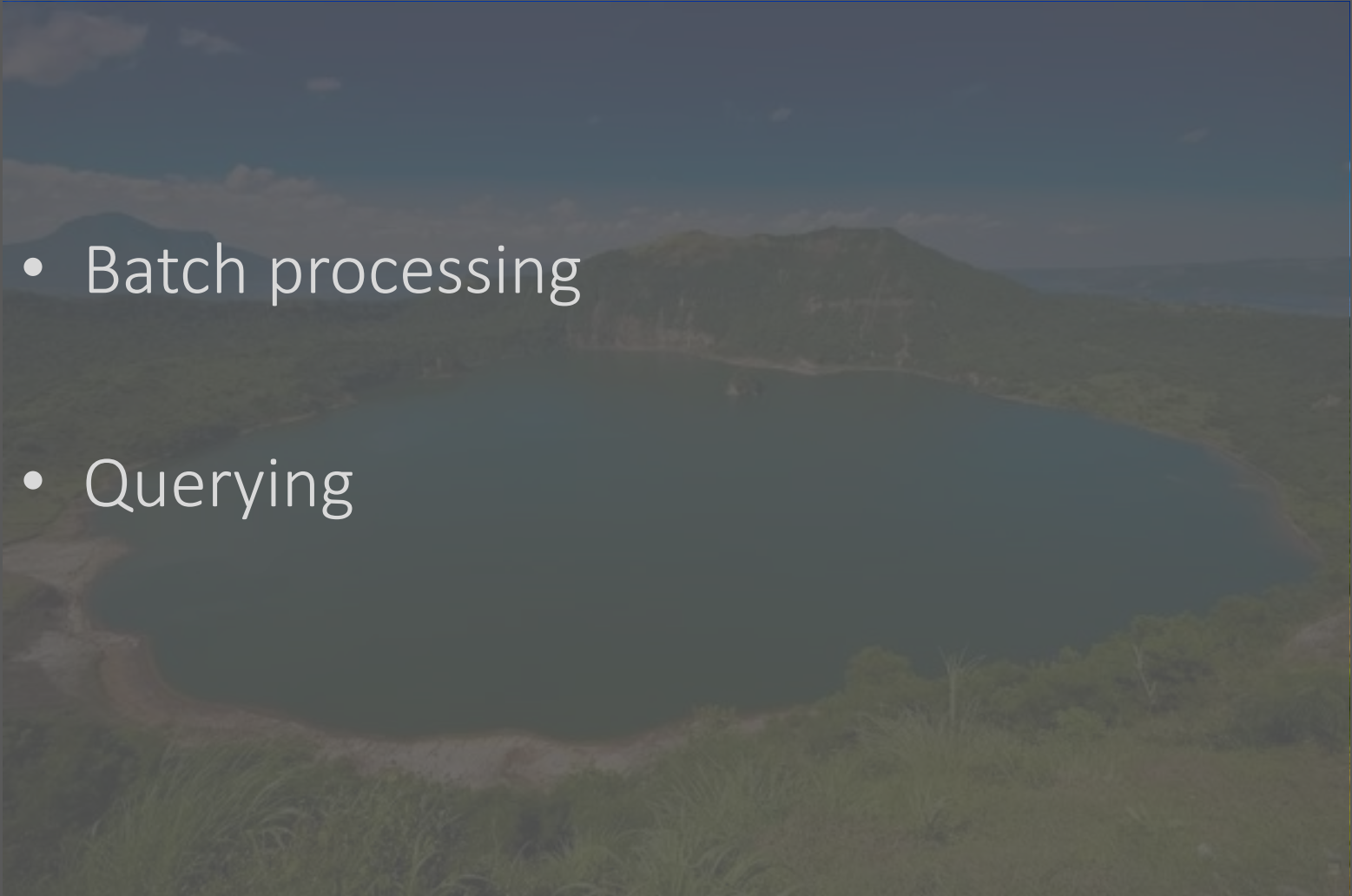
aidhog@gmail.com

Files



Files

- Batch processing
- Querying

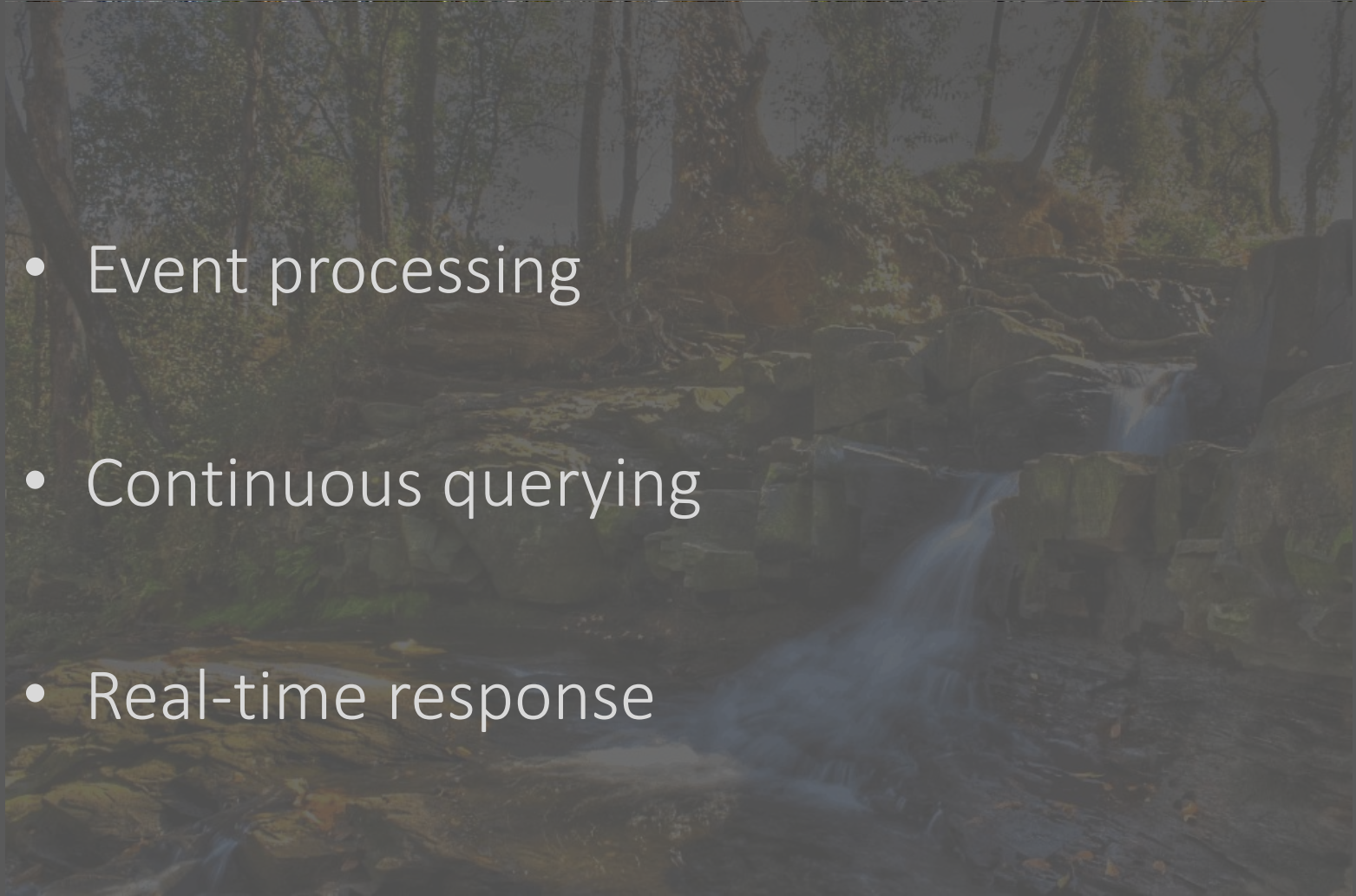


Streams



Streams

- Event processing
- Continuous querying
- Real-time response



Applications: Social Media Analytics



Applications: Social Media Analytics

- Event processing
 - Kitten video goes viral
 - Burst of tweets about earthquakes
- Continuous querying
 - Track sentiment for company's products
 - Monitor popular users tweeting about me
- Real-time response
 - Put Emergency Services on alert
 - Schedule Quality Control (QC) review


Applications: Log Monitoring

```
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Cache-Control' => 'no-cache' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Connection' => 'keep-alive' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Pragma' => 'no-cache' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Accept' => 'application/json, text/javascript, */*; q=0.01' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Accept-Encoding' => 'gzip, deflate' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Accept-Language' => 'en-gb,en;q=0.5' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Host' => 'vx-garcia.wcn.co.uk' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Referer' => 'http://vx-garcia.wcn.co.uk/vx/lang-en-GB/config-jail/channel-1
d27dad84/wid-4/ats/recruiter/profile/edit' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'User-Agent' => 'Mozilla/5.0 (X11; Linux x86_64; rv:17.0) Gecko/17.0 Firefox
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Content-Length' => '134' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Content-Type' => 'application/x-www-form-urlencoded; charset=UTF-8' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'Cookie' => 'utma=1.893613000.1353586388.1356003545.1356012074.105; utmz
[utmccn=(direct)|utmcmd=(none); utma=164338489.1362821095.1354014424.1354641679.1355835790.3; utmaz=164338489.1355835790.3.3.utmcsr=google|utmccn=(organ
provided); wcn_ats_session=000097a096228f7b2bdfa454194513879815e69871128c680440ee76ee108d9d39d6c44ddd261dd4f1a; utmc=1; su_user=0; utmb=1.89.9.135601659
0ec1126768805e55e2137f801ea5d9ad42d9f35dc1f31f70a568b822e61ace6f080f6' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 158] 'X-Requested-With' => 'XMLHttpRequest' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 4] [WCN.Role.PathMunge 173] In parse_url() - user requested access to 'vx-garcia.wcn.co.uk', 'lang-en-GB/config-jail
-1/xf-cbf4d27dad84/wid-4/ats/recruiter/profile/map_update/1'
[Thu Dec 20 15:41:01 2012] [notice] Apache/2.2.16 (Debian) mod_perl/2.0.4 Perl/v5.10.1 configured -- resuming normal operations
[INFO] [47152f7f] [2012/12/20 15:41:01 48] [WCN.Role.PathMunge 747] c->set_system: 51 (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.DBIC 388] Going to set system database to system '51', jail '1'
[INFO] [47152f7f] [2012/12/20 15:41:01 36] [WCN.Role.PathMunge 718] Setting brand to be '2'
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PathMunge 791] Set current language to 'en-GB'
[DEBUG] [47152f7f] [2012/12/20 15:41:01 2] [WCN.Role.PathMunge 319] Cookie for 'recruiter' => '97a096228f7b2bdfa454194513879815e6987112
[DEBUG] [47152f7f] [2012/12/20 15:41:01 6] [WCN.Role.PerformanceLogger 175] *** Request Params (4) ***
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 189] '_vxXSRF_Token' => '4ddc5de6bed3d3760b59c430cbeae48a6f0c8e95' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 189] 'code version' => '1355995679' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 189] 'submitted_via_ajax' => 'true' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 189] 'datafield_53274_1_1[]' => '1798' (1)
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 193] *** Uploads (0) ***
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 215] PERFORMANCE: prepare took 0.11599937057495 secs
[DEBUG] [47152f7f] [2012/12/20 15:41:01 1] [WCN.Role.Session 142] User's session id is '97a096228f7b2bdfa454194513879815e6987112', on server '0'
[DEBUG] [47152f7f] [2012/12/20 15:41:01 17] [WCN.AccessControl 233] Going to _cache_role_profile_rules for recruiter '1', role profile '20'
[DEBUG] [47152f7f] [2012/12/20 15:41:01 16] [WCN.Controller.Root 64] ** Enter root auto
[INFO] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Controller.Root 65] ** User requested access to 'ats/recruiter/profile/map_update/1' from '192.168.146.46'
[DEBUG] [47152f7f] [2012/12/20 15:41:01 10] [WCN.Role.Session 347] Loading session flash
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Controller.Root 219] ** About to return 1 from root auto
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Controller.ATS 61] ** Enter ATS auto
[DEBUG] [47152f7f] [2012/12/20 15:41:01 1] [WCN.Controller.ATS 145] User 'WCN::DBIC::User::Recruiter=HASH(0x7f9b16810610)' (id: 1) logged in
[DEBUG] [47152f7f] [2012/12/20 15:41:01 1] [WCN.Controller.ATS 950] Validate ATS access rights for recruiter '1' to path 'ats/recruiter/profile/map_update/1'
[WARN] [47152f7f] [2012/12/20 15:41:01 37] [WCN.Controller.ATS 988] User '' does not have access to path 'ats/recruiter/profile/map_update' - ACCESS DENIED
[ERROR] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.Controller.BadRequest 104] 403 FORBIDDEN
[DEBUG] [47152f7f] [2012/12/20 15:41:01 1] [WCN.Controller.Root 324] **** enter root controller's end() method
[INFO] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Controller.Root 339] Have already set status (403) and set a body, so will not render any templates
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Controller.Root 376] Set Content-Length header => '1' bytes
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 231] PERFORMANCE: dispatch finished at 0.20909595489502 secs (took: 0.093096017837524
[DEBUG] [47152f7f] [2012/12/20 15:41:01 4] [WCN.Role.PerformanceLogger 249] PERFORMANCE: finalize finished at 0.213540077209473 secs (took: 0.00444412231445
profile/map_update/1'
[DEBUG] [47152f7f] [2012/12/20 15:41:01 0] [WCN.Role.PerformanceLogger 266] PERFORMANCE: SQL query count: 'SELECT' => 15, 'SET' => 4
[DEBUG] [47152f7f] [2012/12/20 15:41:01 2] [WCN.Role.PerformanceLogger 274] PERFORMANCE: SAN calls: '0' total: '0' avg: '0'
```


Applications: Finance



Applications: Finance

- Event processing
 - Company goes public
 - Stock drops sharply
 - Continuous querying
 - Track stocks with gains of 10% in a day
 - Create alerts for major buy/sell transactions
 - Real-time response
 - BUY BUY BUY
 - SELL SELL SELL
- 
- The background of the slide features a complex financial data visualization. It includes a line graph with multiple data series, a bar chart at the bottom, and a grid of numerical values. The overall aesthetic is dark blue and grey, typical of a professional financial dashboard.

Applications: Astronomy

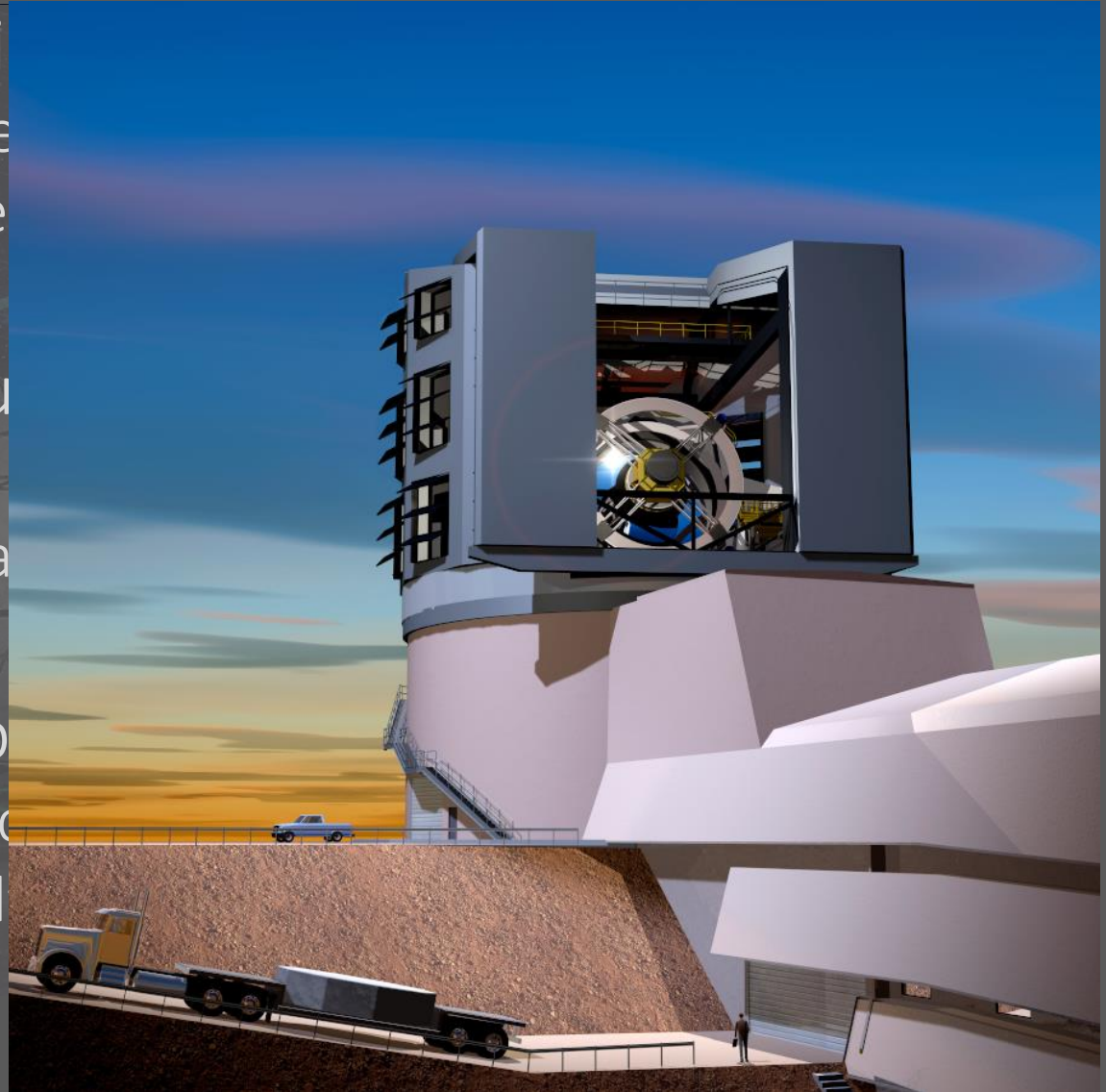


Applications: Astronomy

- Event processing
 - The telescope moves
 - A light source flashes
- Continuous querying
 - Find possible supernovae
 - Track object across the sky
- Real-time response
 - Refocus telescope on important object
 - Lower data filter thresholds

Applications: Astronomy

- Event processing
 - The telescope
 - A light source
- Continuous query
 - Find possible
 - Track object a
- Real-time response
 - Refocus telescope
 - Lower data fil



Streams: Internet of Things













- Event processing
 - A light turns on
 - It starts to rain
- Continuous querying
 - Tell me when temperature reaches 30°
 - Update position of vehicle
- Real-time response
 - Turn off air conditioning
 - Take another route

DISTRIBUTED STREAMING PLATFORM

Available Frameworks



Available Frameworks

												
	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 2016)	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	838	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	IgniteDataStream	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 Scala	Java	Scala	Java	Java	Java
API languages	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	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 Localitys AsialInfo Opentable Fairdata 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.bitbucket.io/an-overview-of-apache-streaming-technologies/>

Application: Emergency Response



chile natural disasters



All **Images** Videos News Maps More

Settings Tools

View saved SafeSearch

tsunami

natural hazard

earthquake

volcano

landslide

mudslide

school

photography

building

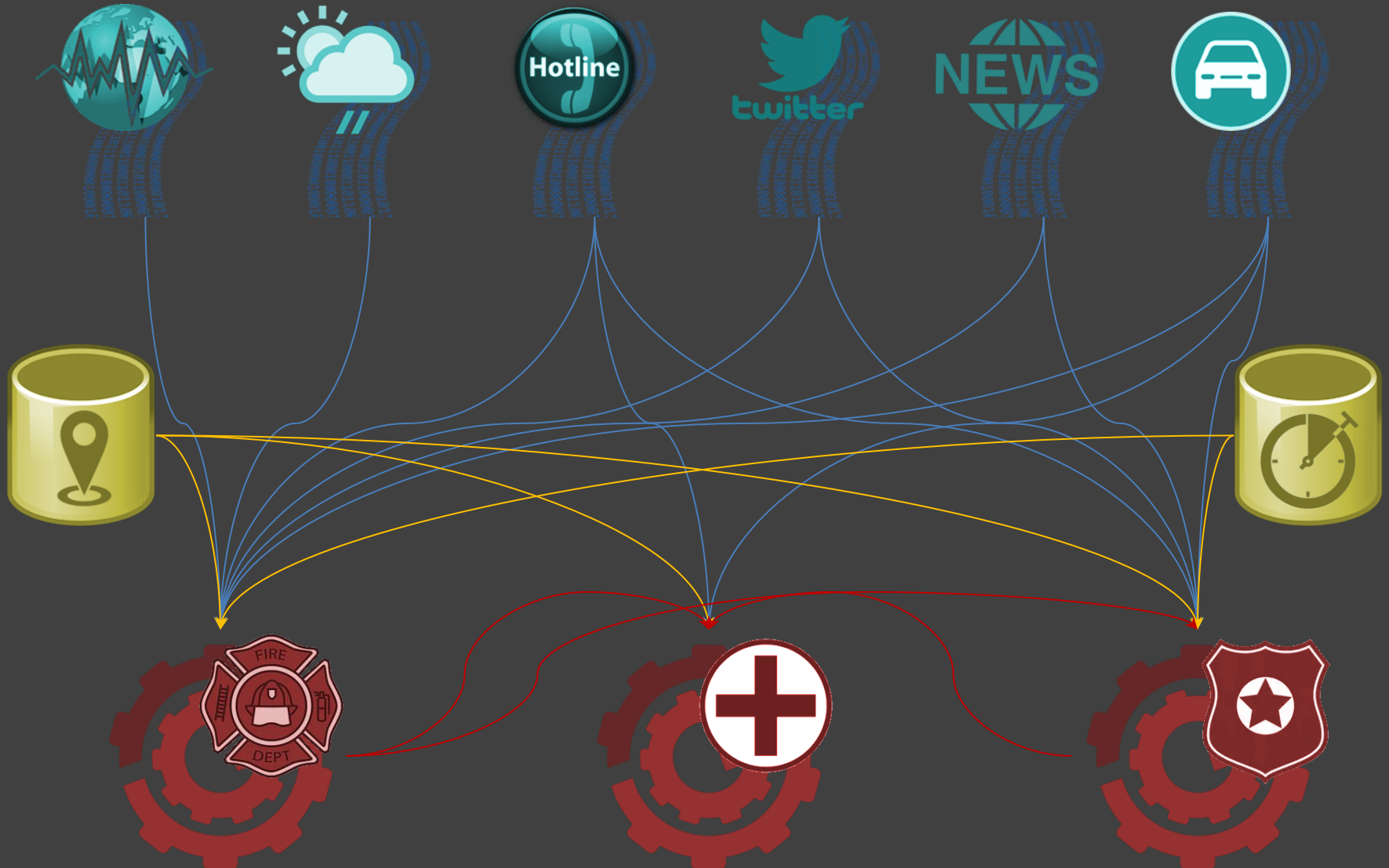
population

historic

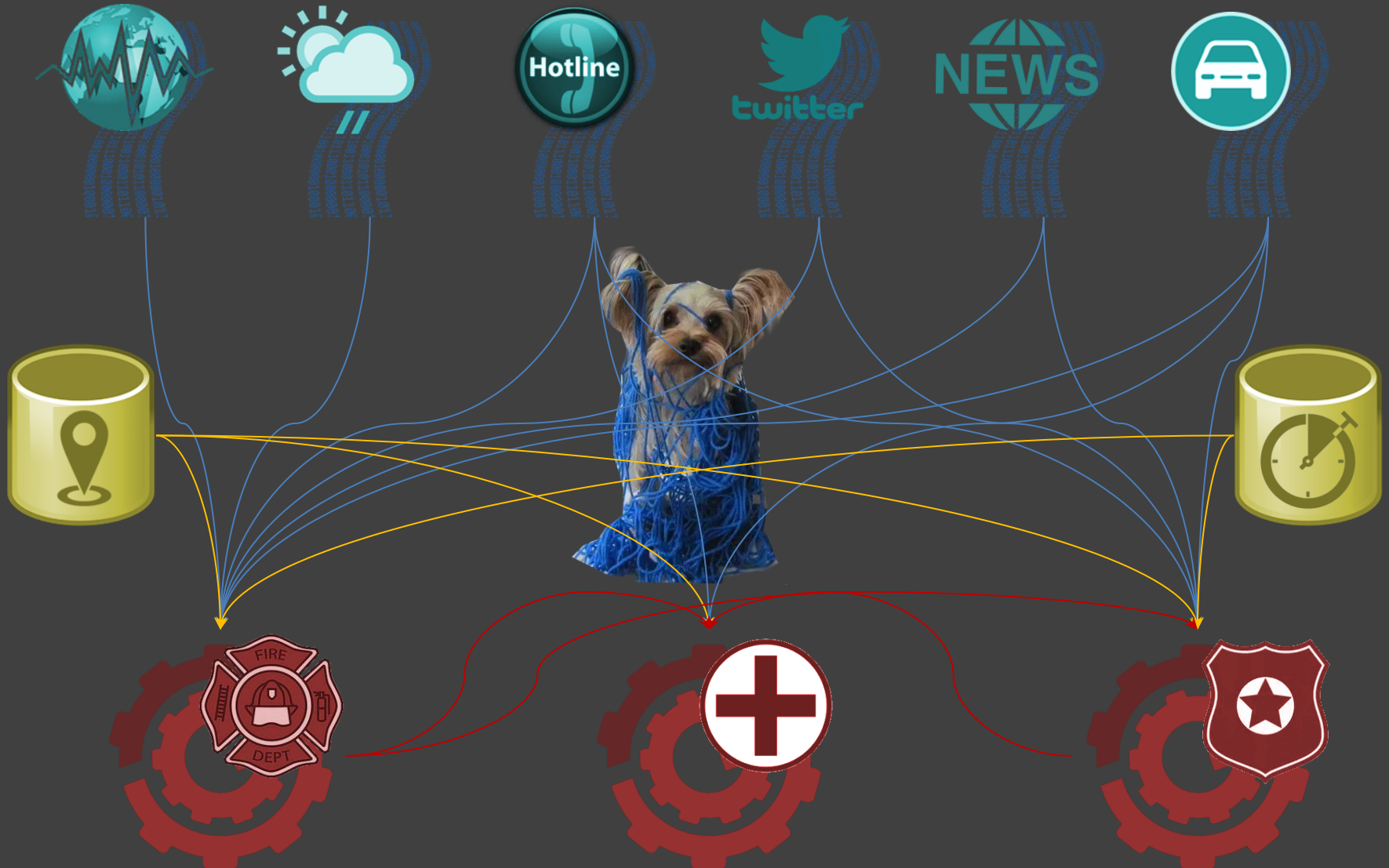
feature



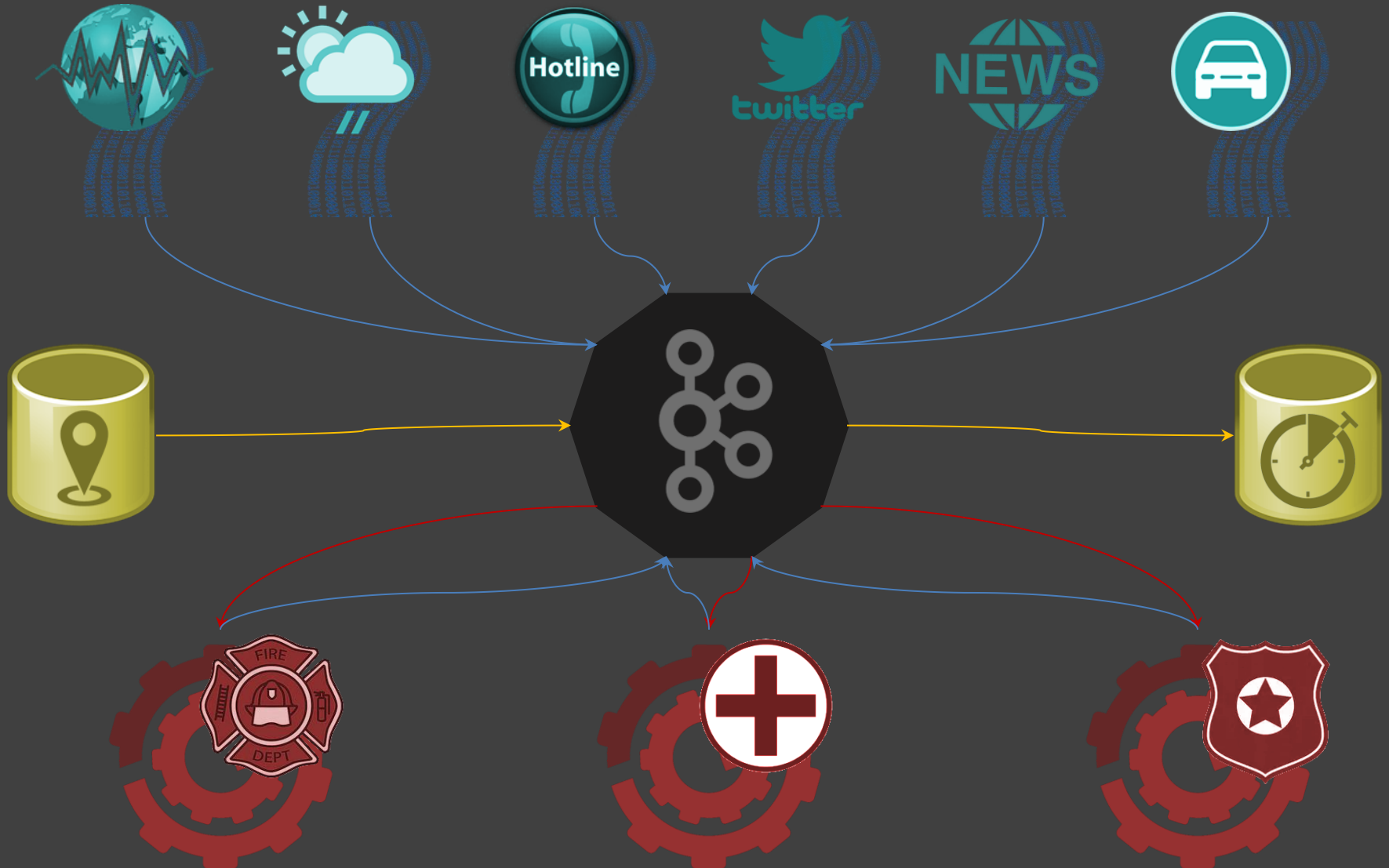
Real-Time Emergency Response



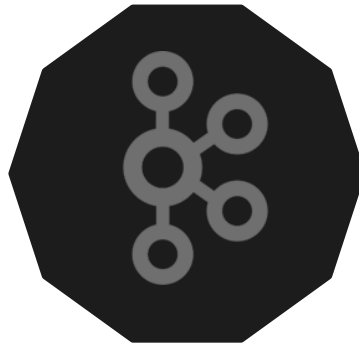
Real-Time Emergency Response



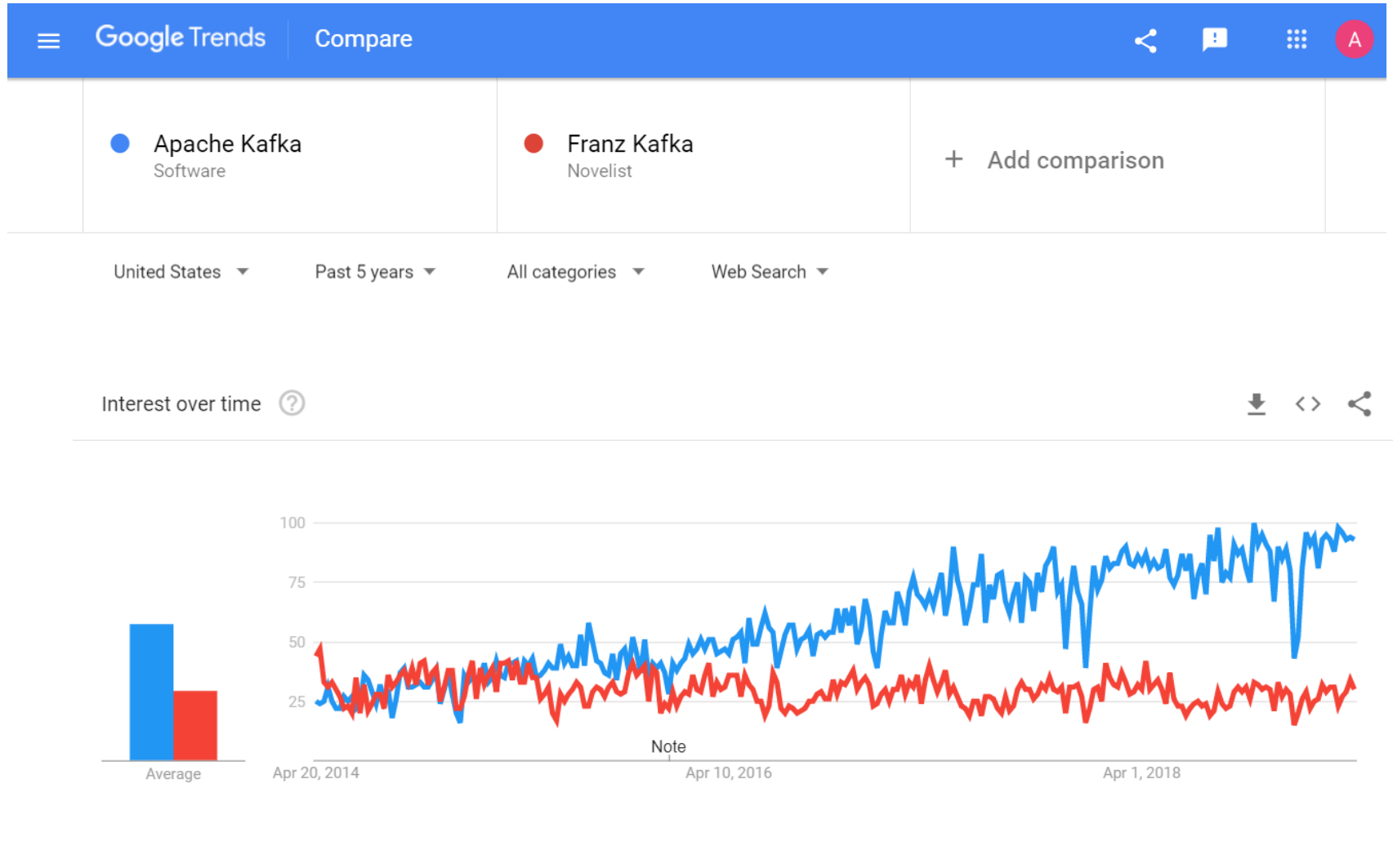
Real-Time Emergency Response



APACHE KAFKA




Apache Kafka vs. Franz Kafka



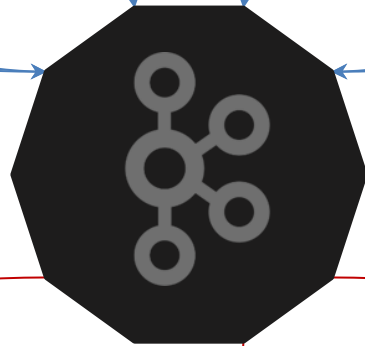
Apache Kafka



- Open Source
- Scala / Java
- Originated in 

Kafka Overview

Producers (Push)



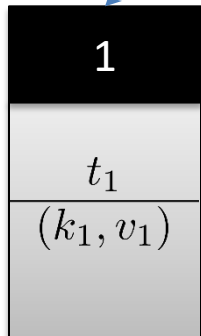
Consumers (Pull)



KAFKA: DATA MODEL

Kafka Record

Producers



Consumers

Kafka Record

Producers

- Records represent "events"




- Records are immutable

- Contain id (offset), timestamp, key and value
 - Timestamp assigned by application or Kafka

Consumers

Kafka Ledger

Producers



1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

Consumers

Kafka Ledger

Producers

- Producers may only append to ledger



Consumers

Kafka Log

Producers

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

Consumers

Kafka Log

Producers

- Producers may only append to log

1 2 3 4 5 6 7 8 ...

t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

- Consumers can read from anywhere*

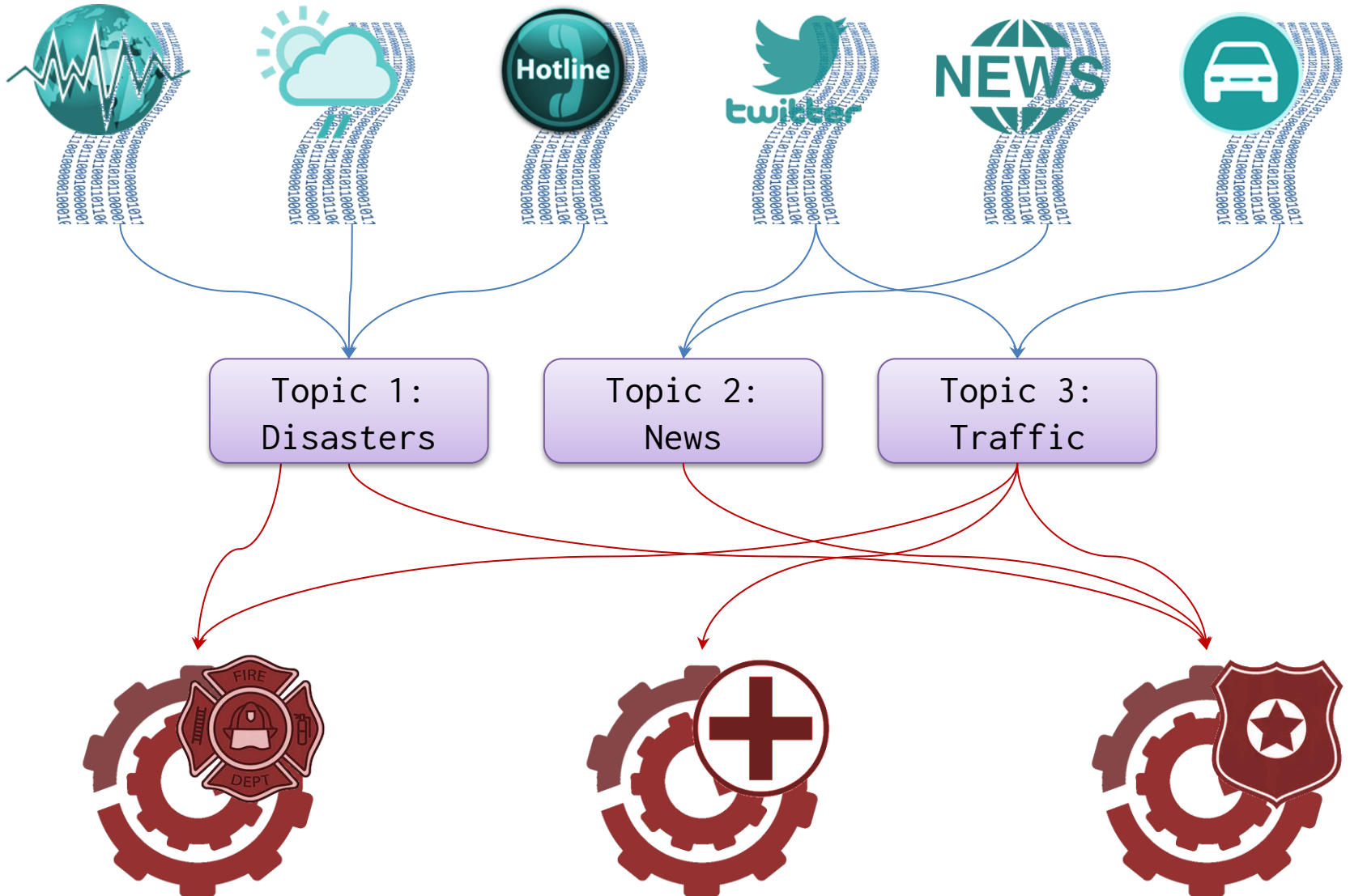
* kind of

Consumers

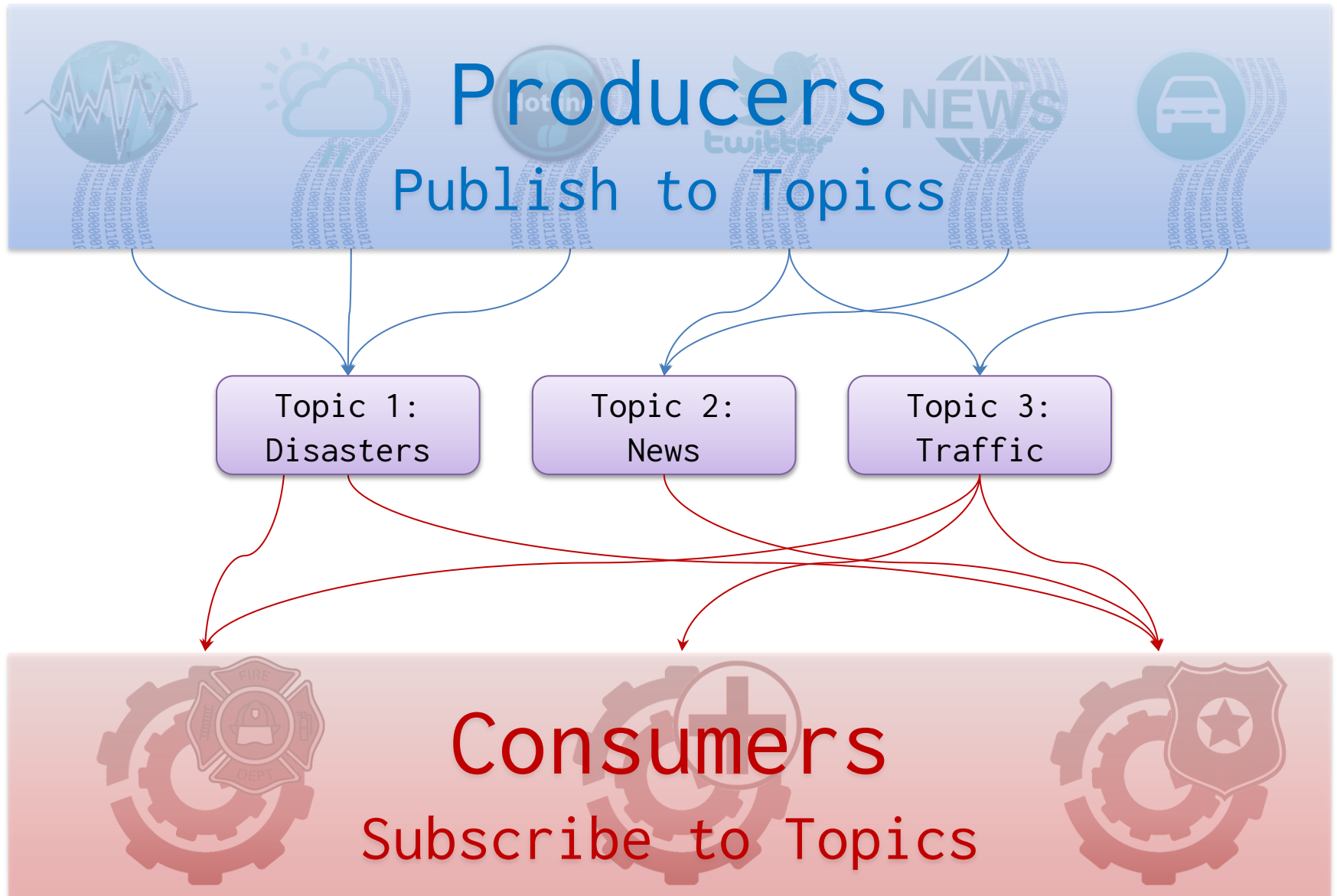


KAFKA: TOPICS

Kafka Topics




Kafka Topics



Topic: Default Partitioning by Key


Partition 1

1	2	3	4	5	6	...
t_1^1	t_2^1	t_3^1	t_4^1	t_5^1	t_6^1	
(k_1^1, v_1^1)	(k_2^1, v_2^1)	(k_3^1, v_3^1)	(k_4^1, v_4^1)	(k_5^1, v_5^1)	(k_6^1, v_6^1)	



Partition 2

1	2	3	4	5	6	7	8	...
t_1^2	t_2^2	t_3^2	t_4^2	t_5^2	t_6^2	t_7^2	t_8^2	
(k_1^2, v_1^2)	(k_2^2, v_2^2)	(k_3^2, v_3^2)	(k_4^2, v_4^2)	(k_5^2, v_5^2)	(k_6^2, v_6^2)	(k_7^2, v_7^2)	(k_8^2, v_8^2)	



Partition 3

1	2	3	4	...
t_1^3	t_2^3	t_3^3	t_4^3	
(k_1^3, v_1^3)	(k_2^3, v_2^3)	(k_3^3, v_3^3)	(k_4^3, v_4^3)	



Topic

Partition 1

- Topics are persistent (on disk)

t_1^1					
(k_1^1, v_1^1)	(k_2^1, v_2^1)	(k_3^1, v_3^1)	(k_4^1, v_4^1)	(k_5^1, v_5^1)	(k_6^1, v_6^1)

– Configurable retention policy

- Keep everything
- Delete once consumed
- Keep for a period of time
- Use fixed amount of space

Partition 2

1					6
t_1^2	t_2^2	t_3^2	t_4^2	t_5^2	t_6^2
(k_1^2, v_1^2)	(k_2^2, v_2^2)	(k_3^2, v_3^2)	(k_4^2, v_4^2)	(k_5^2, v_5^2)	(k_6^2, v_6^2)

Partition 3

1	2	3	4
t_1^3	t_2^3	t_3^3	t_4^3
(k_1^3, v_1^3)	(k_2^3, v_2^3)	(k_3^3, v_3^3)	(k_4^3, v_4^3)



Ordering

Partition 1

1	2	3	4	5	6
t_1^1	t_2^1	t_3^1	t_4^1	t_5^1	t_6^1
(k_1^1, v_1^1)	(k_2^1, v_2^1)	(k_3^1, v_3^1)	(k_4^1, v_4^1)	(k_5^1, v_5^1)	(k_6^1, v_6^1)



Partition 2

1	2	3	4	5	6	7	8
t_1^2	t_2^2	t_3^2	t_4^2	t_5^2	t_6^2	t_7^2	t_8^2
(k_1^2, v_1^2)	(k_2^2, v_2^2)	(k_3^2, v_3^2)	(k_4^2, v_4^2)	(k_5^2, v_5^2)	(k_6^2, v_6^2)	(k_7^2, v_7^2)	(k_8^2, v_8^2)



Partition 3

1	2	3	4
t_1^3	t_2^3	t_3^3	t_4^3
(k_1^3, v_1^3)	(k_2^3, v_2^3)	(k_3^3, v_3^3)	(k_4^3, v_4^3)



Ordering

Partition 1

- Ordering (offset) guaranteed per partition

t_1^1					t_6^1
(k_1^1, v_1^1)	(k_2^1, v_2^1)	(k_3^1, v_3^1)	(k_4^1, v_4^1)	(k_5^1, v_5^1)	(k_6^1, v_6^1)

- Not across partitions!
- For ordering across partitions, use timestamp

Partition 2

1	2	3	4	5	6	7	8
t_1^2	t_2^2	t_3^2	t_4^2	t_5^2	t_6^2	t_7^2	t_8^2
(k_1^2, v_1^2)	(k_2^2, v_2^2)	(k_3^2, v_3^2)	(k_4^2, v_4^2)	(k_5^2, v_5^2)	(k_6^2, v_6^2)	(k_7^2, v_7^2)	(k_8^2, v_8^2)

Partition 3

1	2	3	4
t_1^3	t_2^3	t_3^3	t_4^3
(k_1^3, v_1^3)	(k_2^3, v_2^3)	(k_3^3, v_3^3)	(k_4^3, v_4^3)



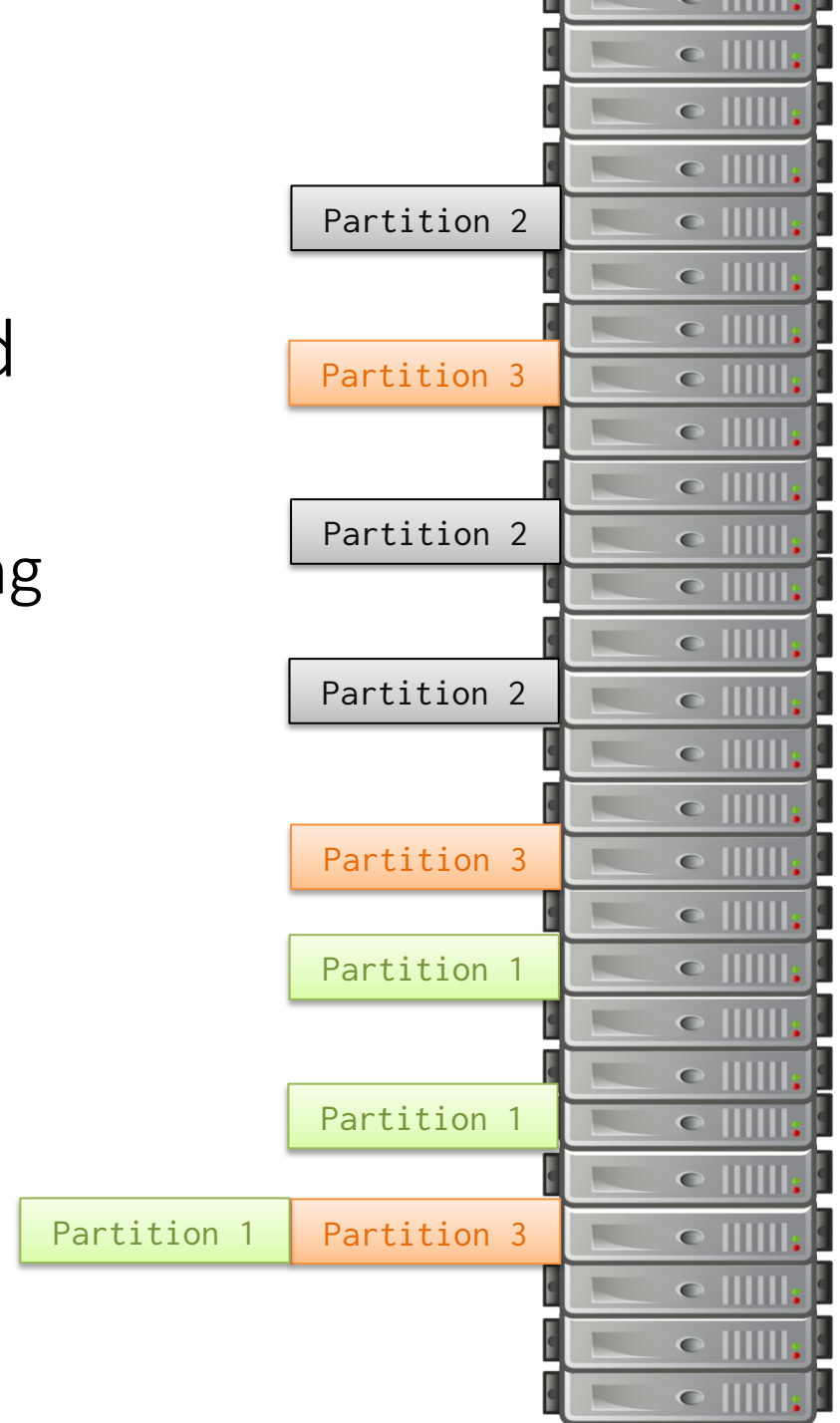
Replication

- Topics can be replicated
 - Choose factor per topic
 - Automatic load balancing

Problem?

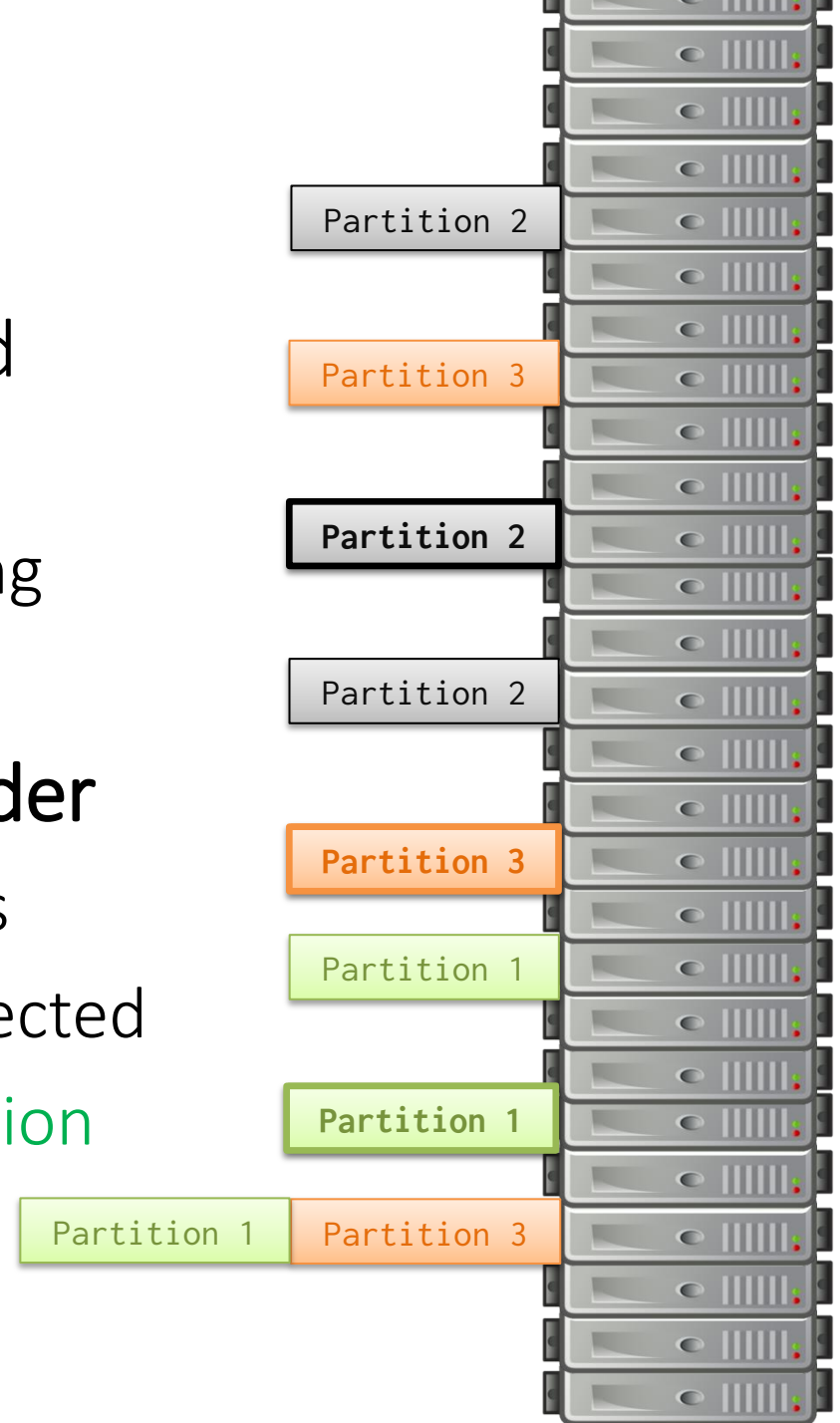


Order?



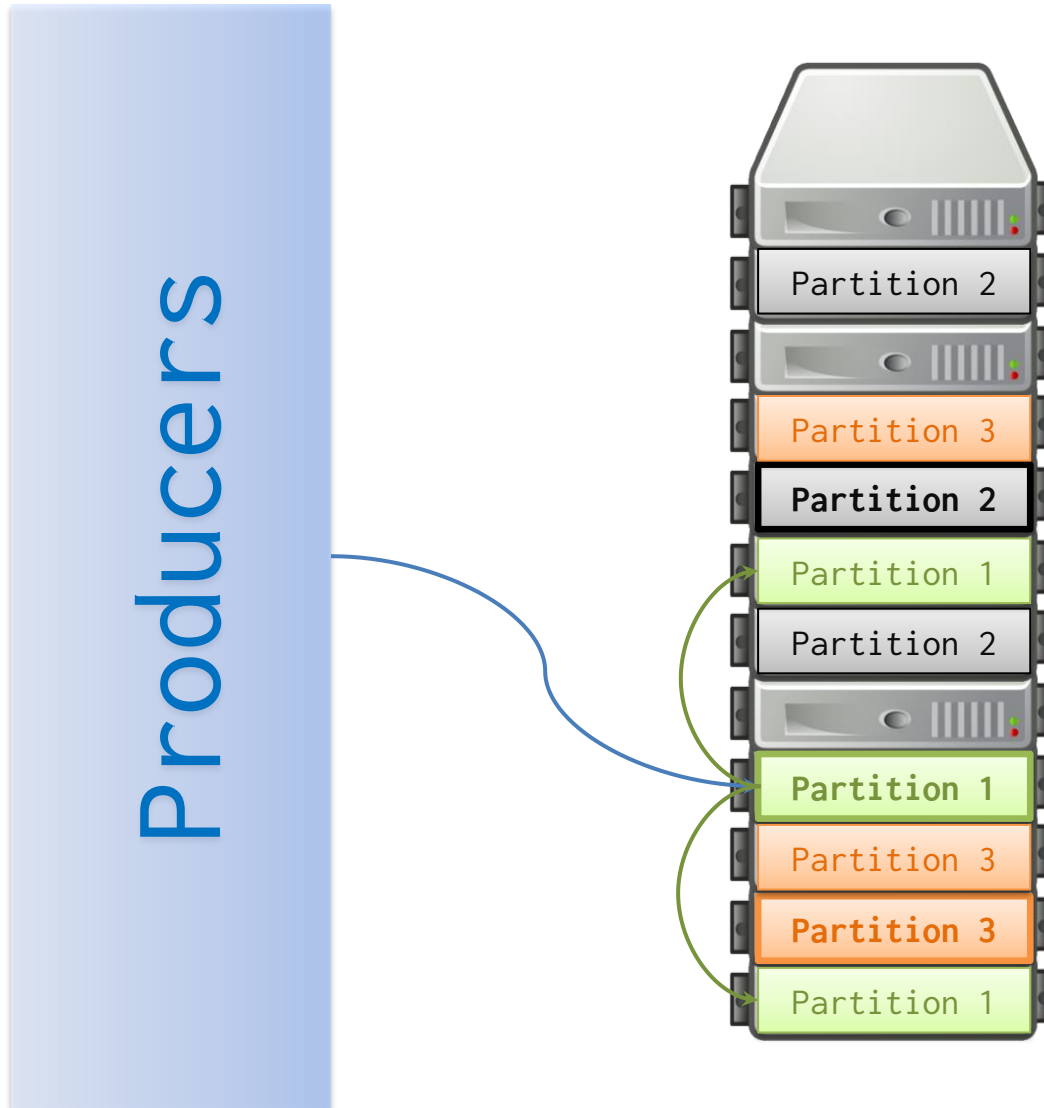
Leader

- Topics can be replicated
 - Choose factor per topic
 - Automatic load balancing
- One machine is the **leader**
 - The others are followers
 - Leader automatically elected
 - Ensures order per partition
 - Reads/writes to leader

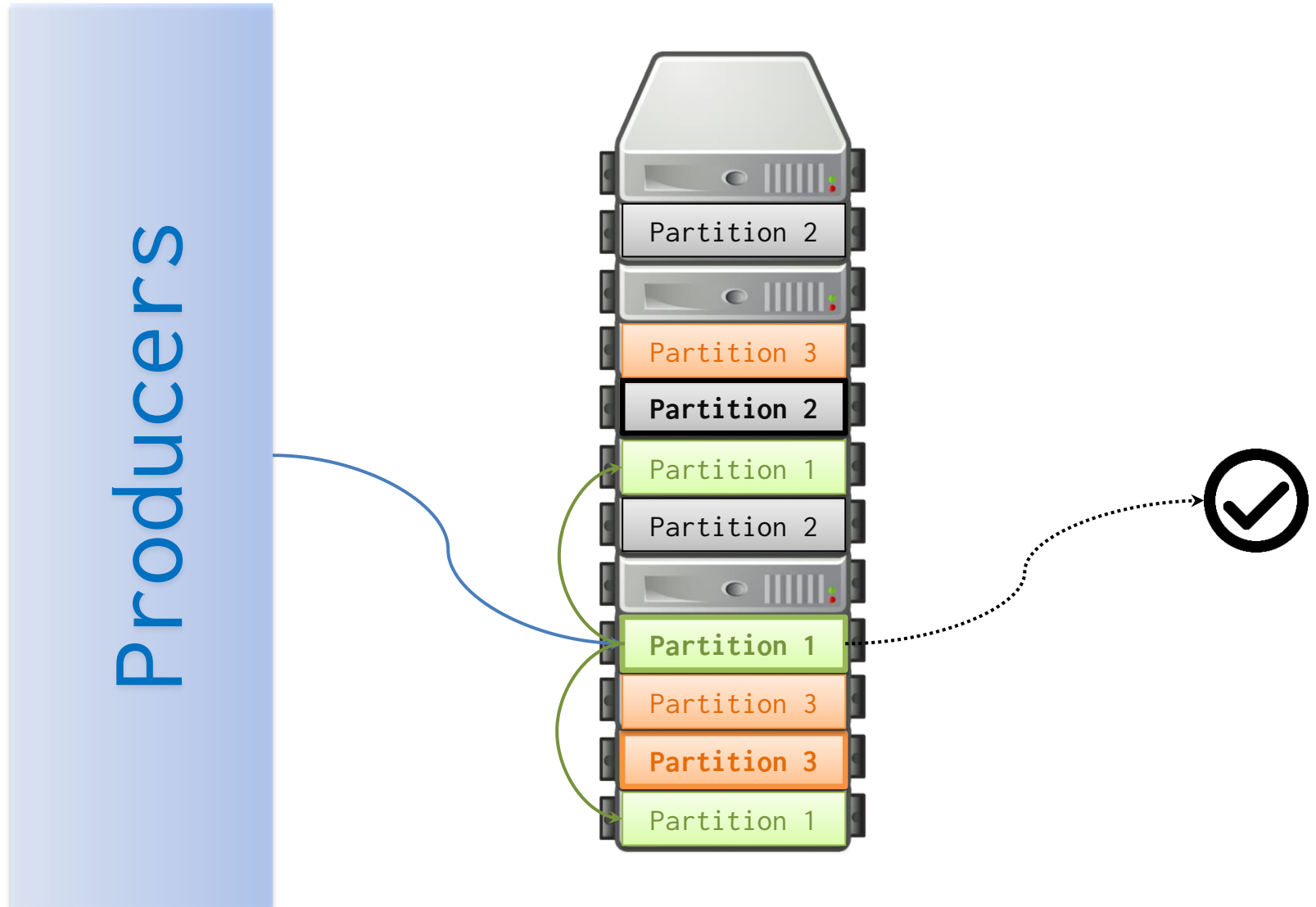


KAFKA: WRITE GUARANTEES

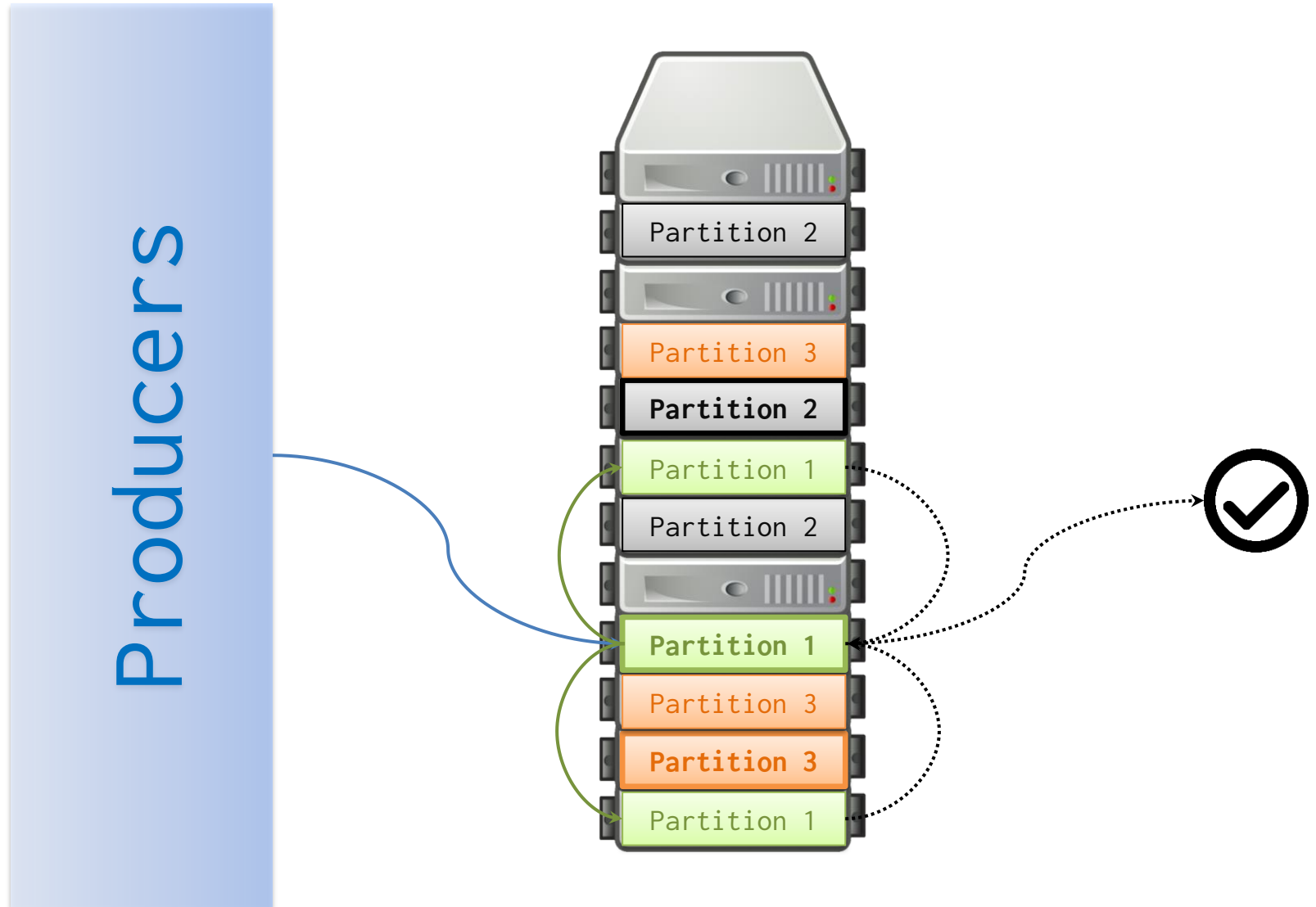
Writes: Asynchronous (No Guarantee)



Writes: Leader Commit



Writes: Leader Commit + Quorum (2)



Write Guarantees

- Asynchronous
 - No guarantee
 - Very low latency
- Leader Commit
 - Persistent on leader
 - Medium latency (disk write + network ack)
- Leader Commit + Quorum n
 - Persistent on leader + n machines
 - High latency (disk writes + network acks)

KAFKA: READS

Kafka tracks consumer offset

C1: 1-2

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



Kafka tracks consumer offset

C1: 3-4

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



1
2
3
4

Kafka tracks consumer offset

C1: 5-6

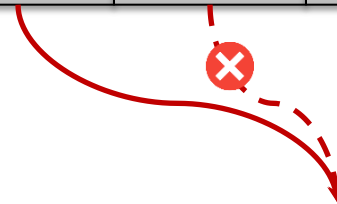
1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



1
2
3
4
5
6

Failures?

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



- 1
- 2
- 3
- 4
- 5
- 6

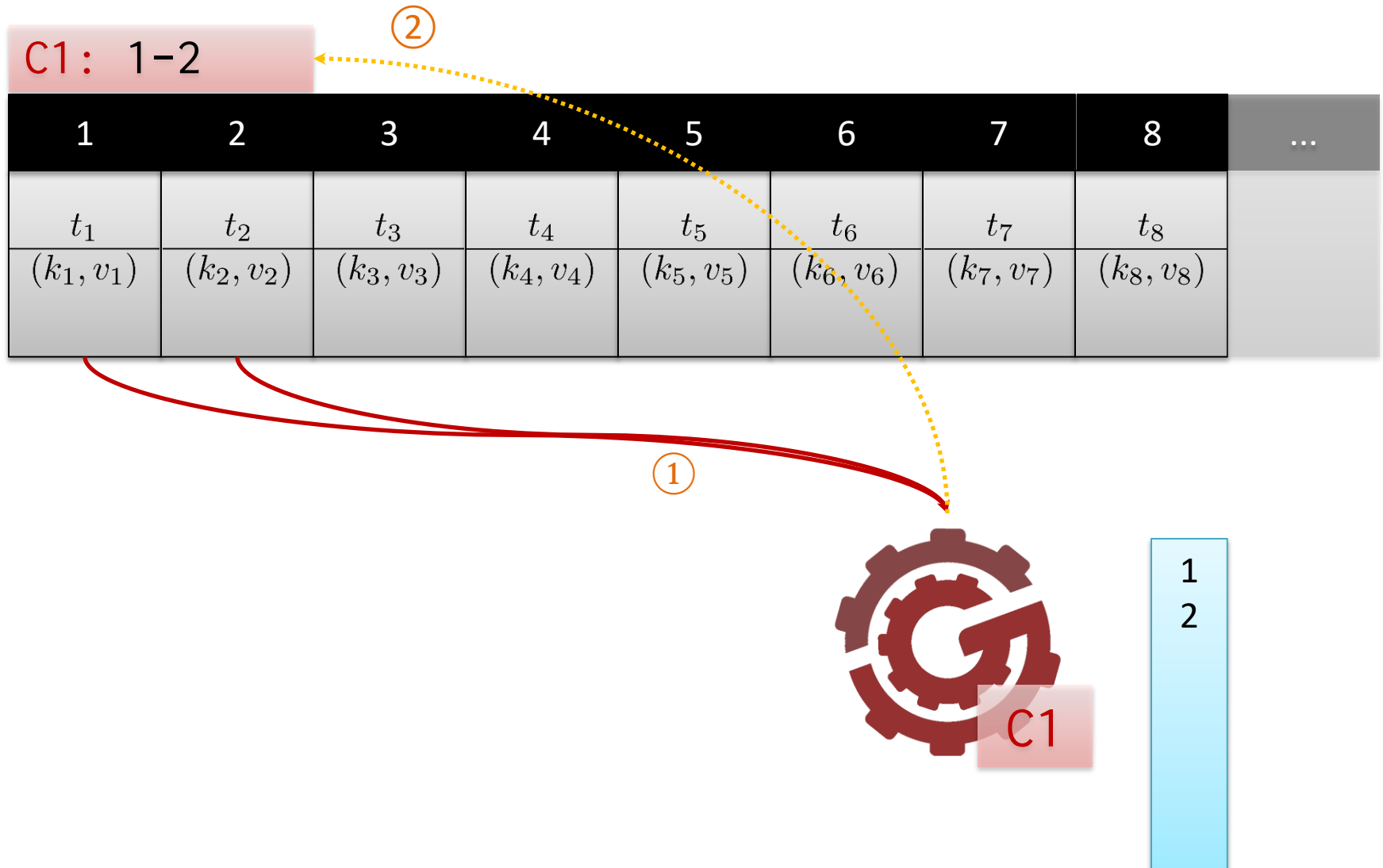
What should we do in the case of a read failure?

KAFKA: READ GUARANTEES

Read Guarantees

- At least once
 - Each value processed at least once
 - Consumer offset updated on consumer ACK
- At most once
- Effectively once
- Exactly once

Read: At Least Once (Default)



Read: At Least Once (Default)

C1: 1-2

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



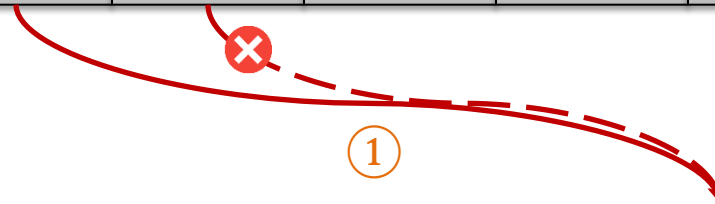
C1

1
2

Read: At Least Once (Default)

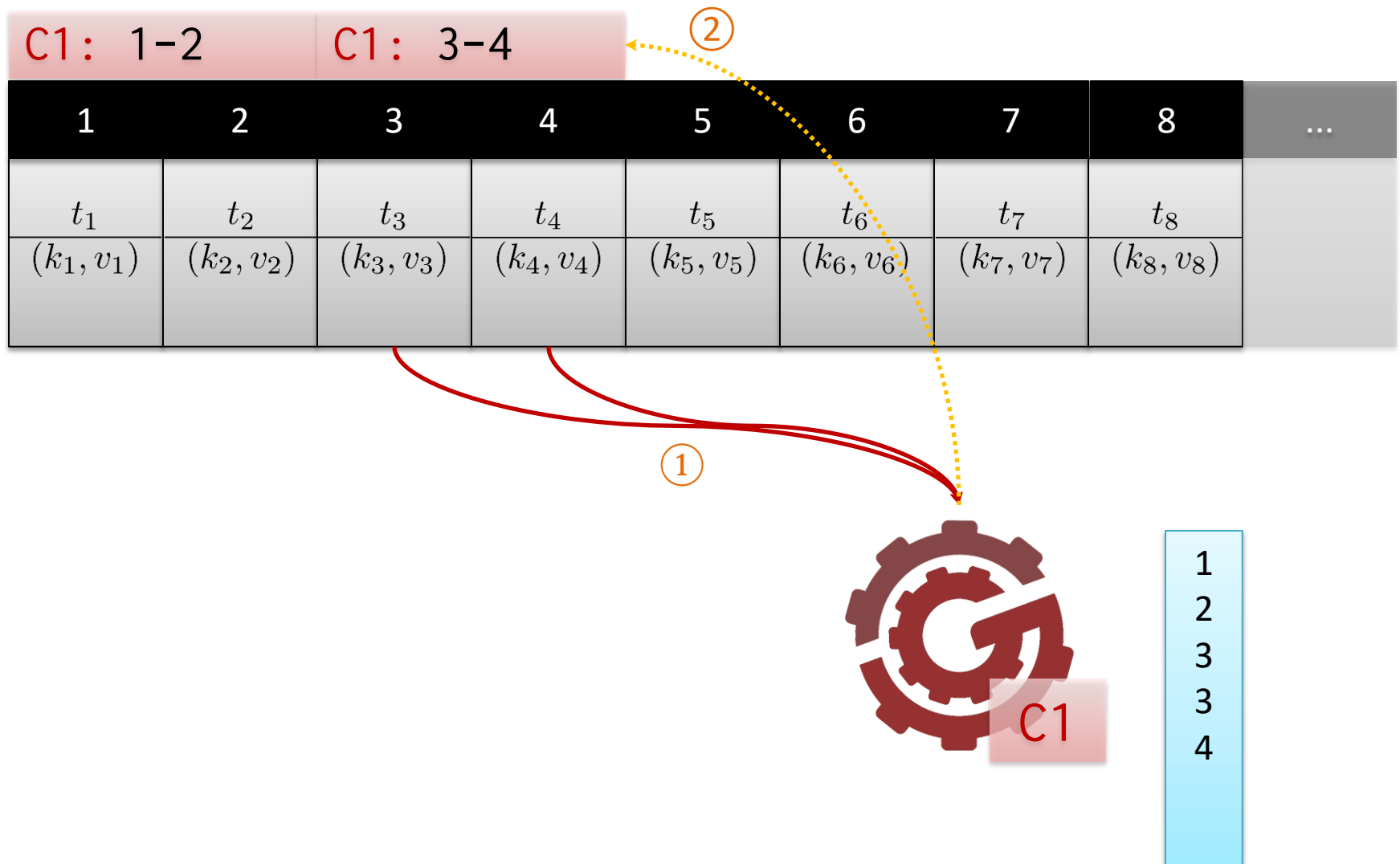
C1: 1-2

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

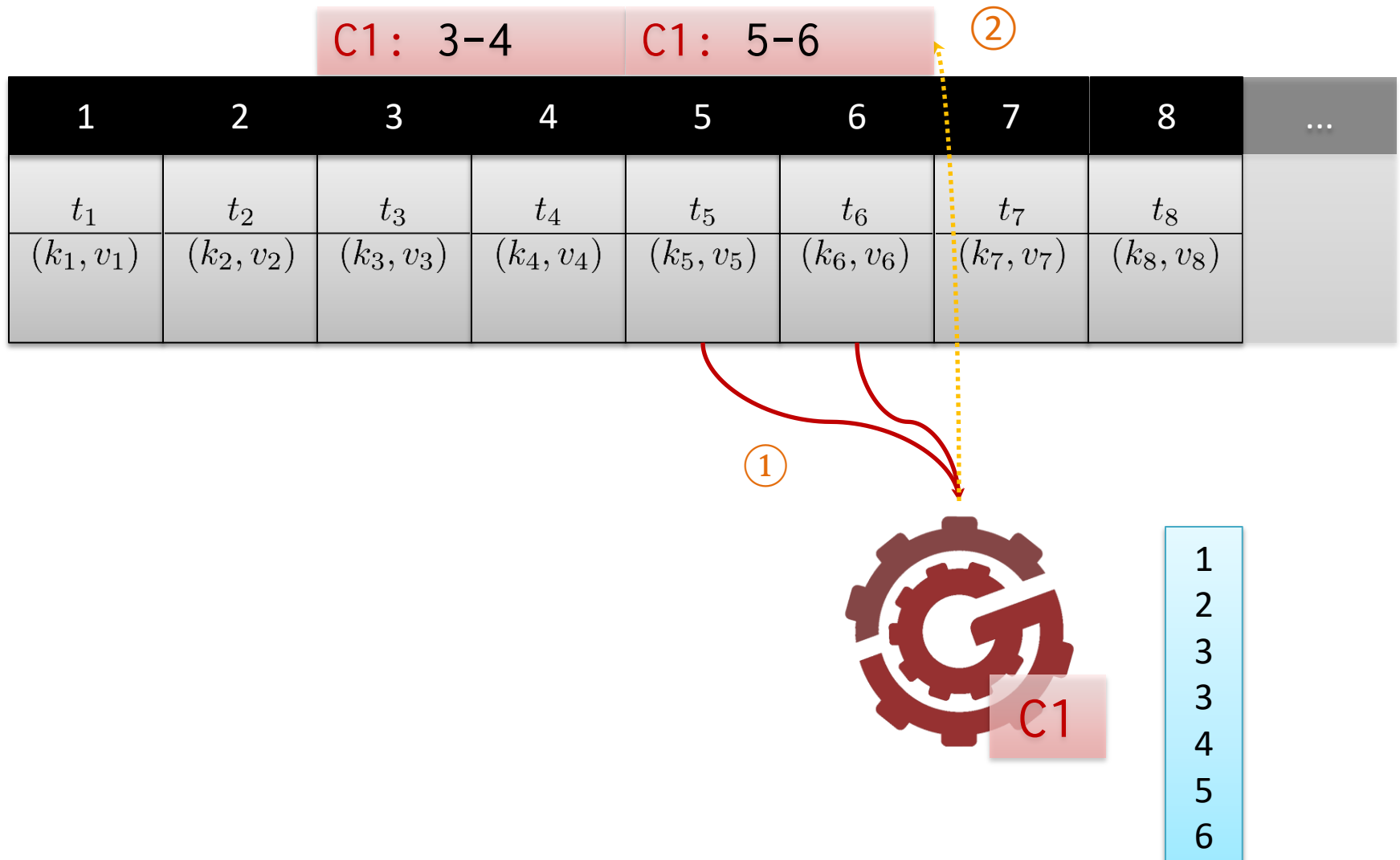


- 1
- 2
- 3

Read: At Least Once (Default)



Read: At Least Once (Default)



Read Guarantees

- At least once
- At most once
 - Each value processed at most once
 - Consumer offset updated immediately
- Effectively once
- Exactly once

Read: At Most Once

①

C1: 1-2

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

②



Read: At Most Once

①

C1: 3-4

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

②



- 1
- 2
- 3

Read: At Most Once

①

C1: 5-6

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

②

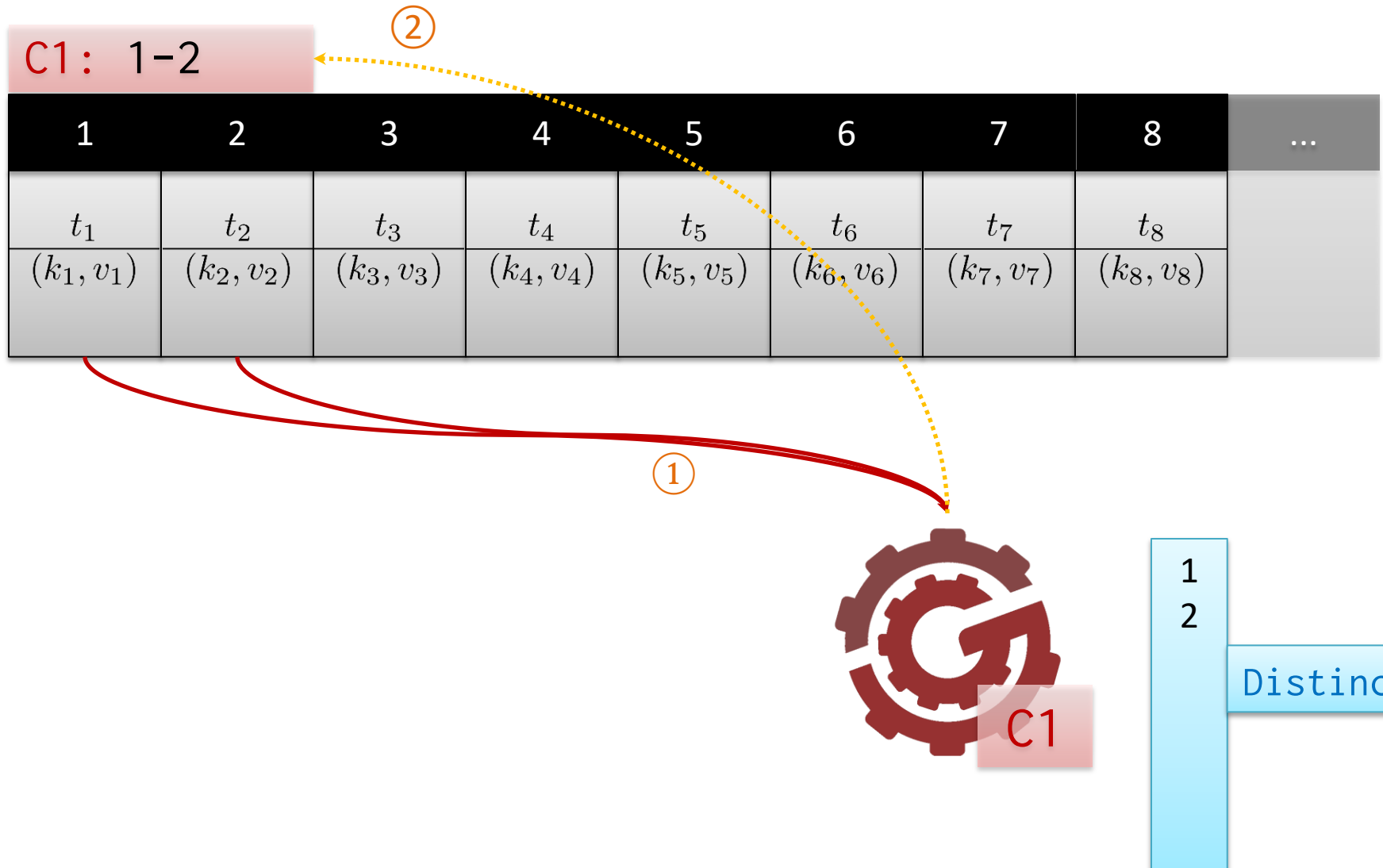


1
2
3
5
6

Read Guarantees

- At least once
- At most once
- Effectively once
 - At least once but ...
 - Consumer takes care of duplicates
- Exactly once

Read: Effectively Once



Read: Effectively Once

C1: 1-2

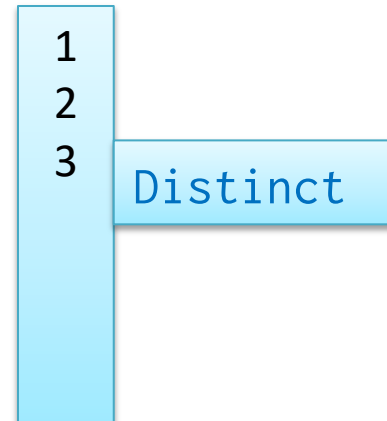
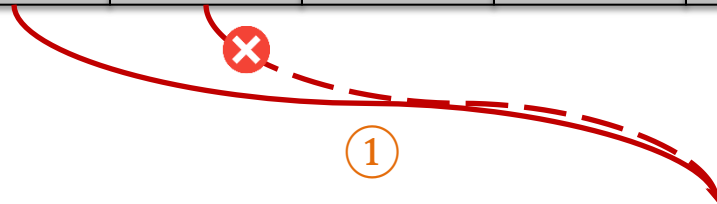
1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



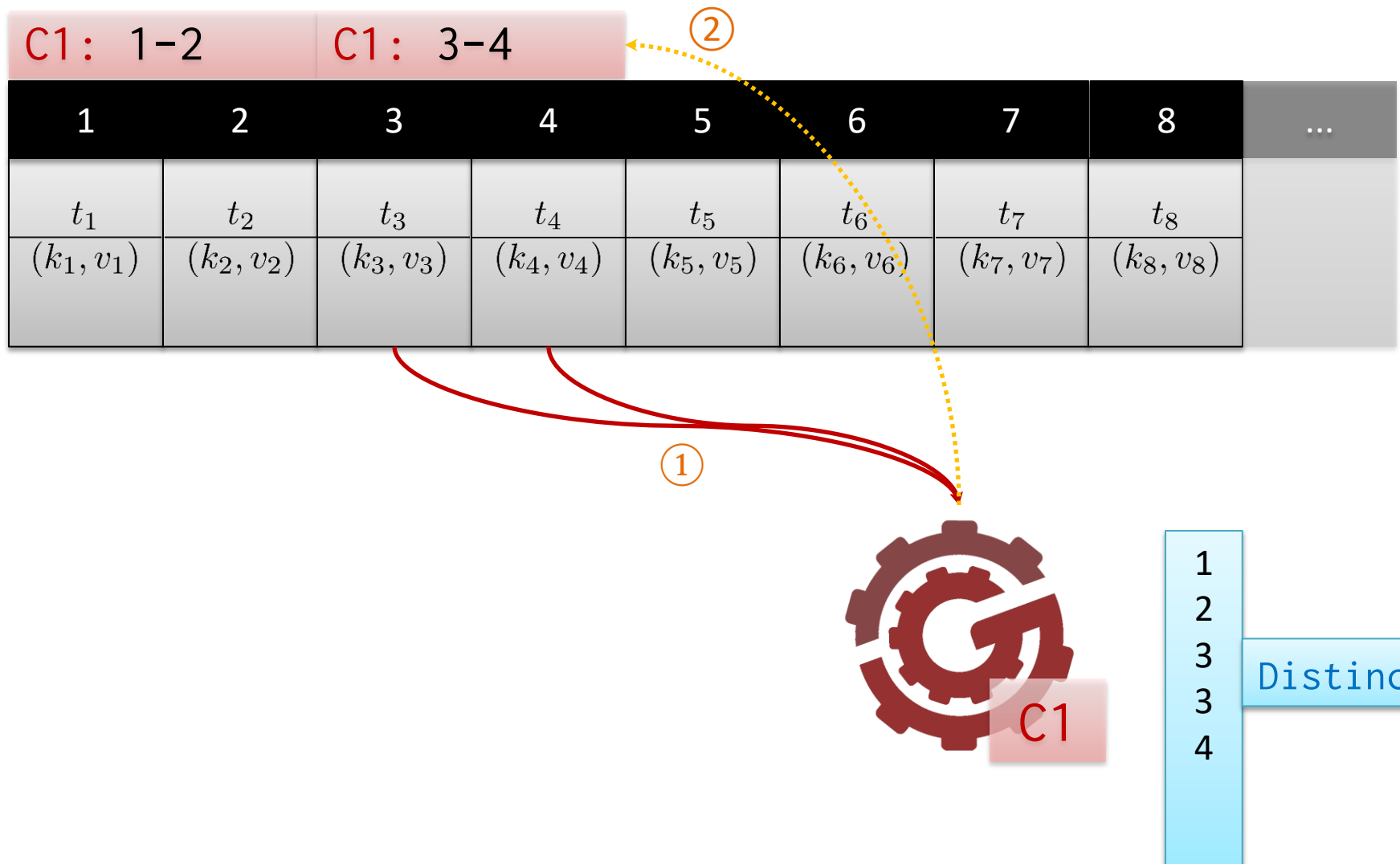
Read: Effectively Once

C1: 1-2

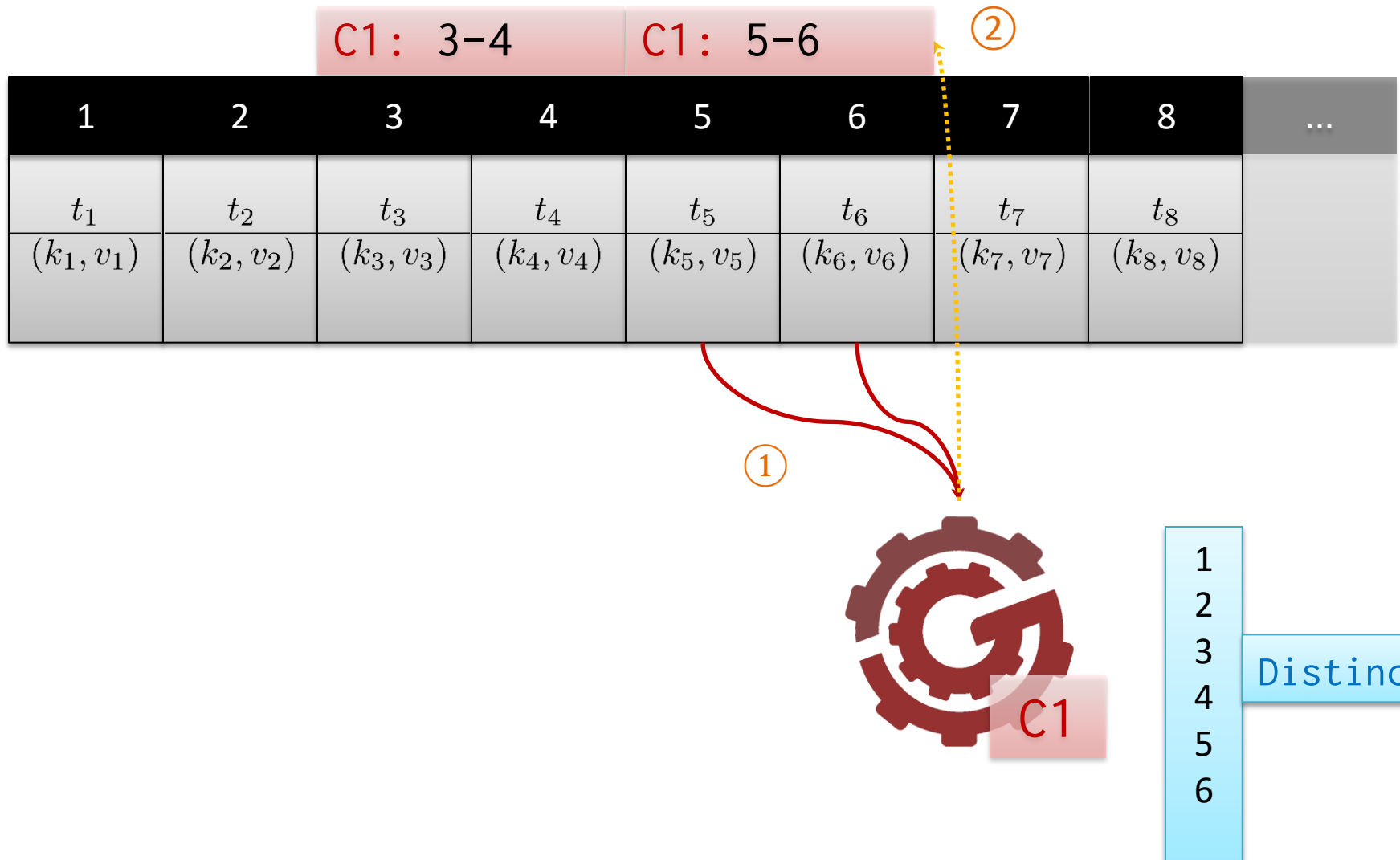
1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



Read: Effectively Once



Read: Effectively Once

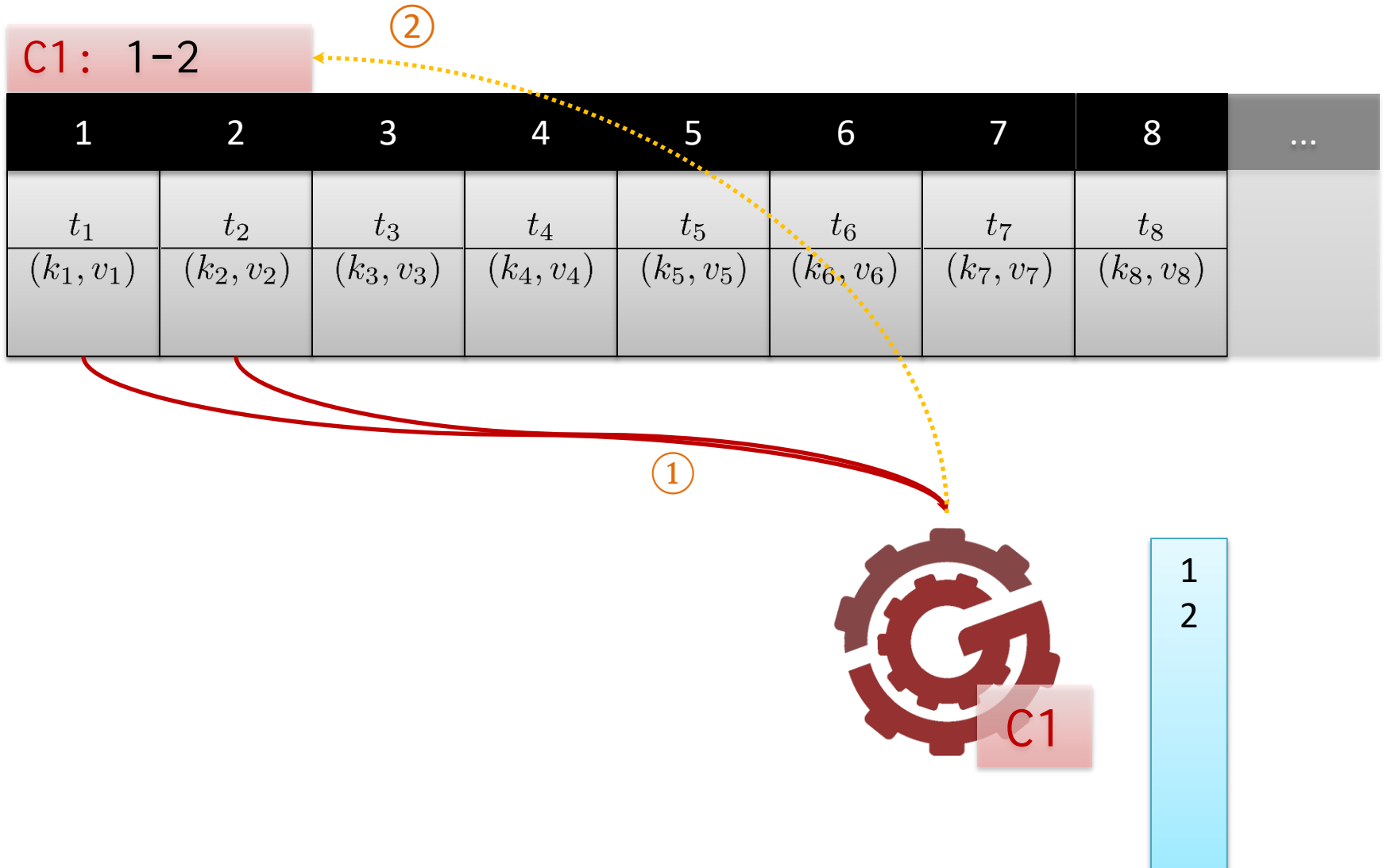


Read Guarantees

- At least once
- At most once
- Effectively once
- Exactly once
 - Data and offset updated as a single transaction

Read: Exactly Once

Transaction 1: ①, ②



Read: Exactly Once

C1: 1-2

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



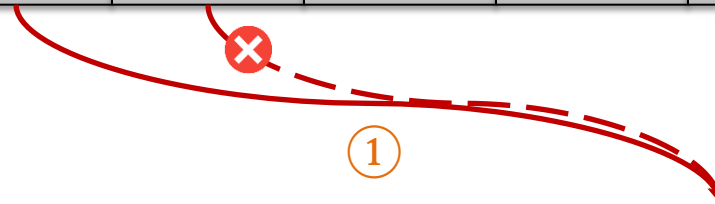
1
2

Read: Exactly Once

Transaction 2: ①, ②

C1: 1-2

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	



Read: Exactly Once

Transaction 3: ①, ②

C1: 1-2		C1: 3-4						
1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

①

②



1
2
3
4

Read: Exactly Once

Transaction 4: ①, ②

1	2	3	4	5	6	7	8	...
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	
(k_1, v_1)	(k_2, v_2)	(k_3, v_3)	(k_4, v_4)	(k_5, v_5)	(k_6, v_6)	(k_7, v_7)	(k_8, v_8)	

C1: 3-4

C1: 5-6

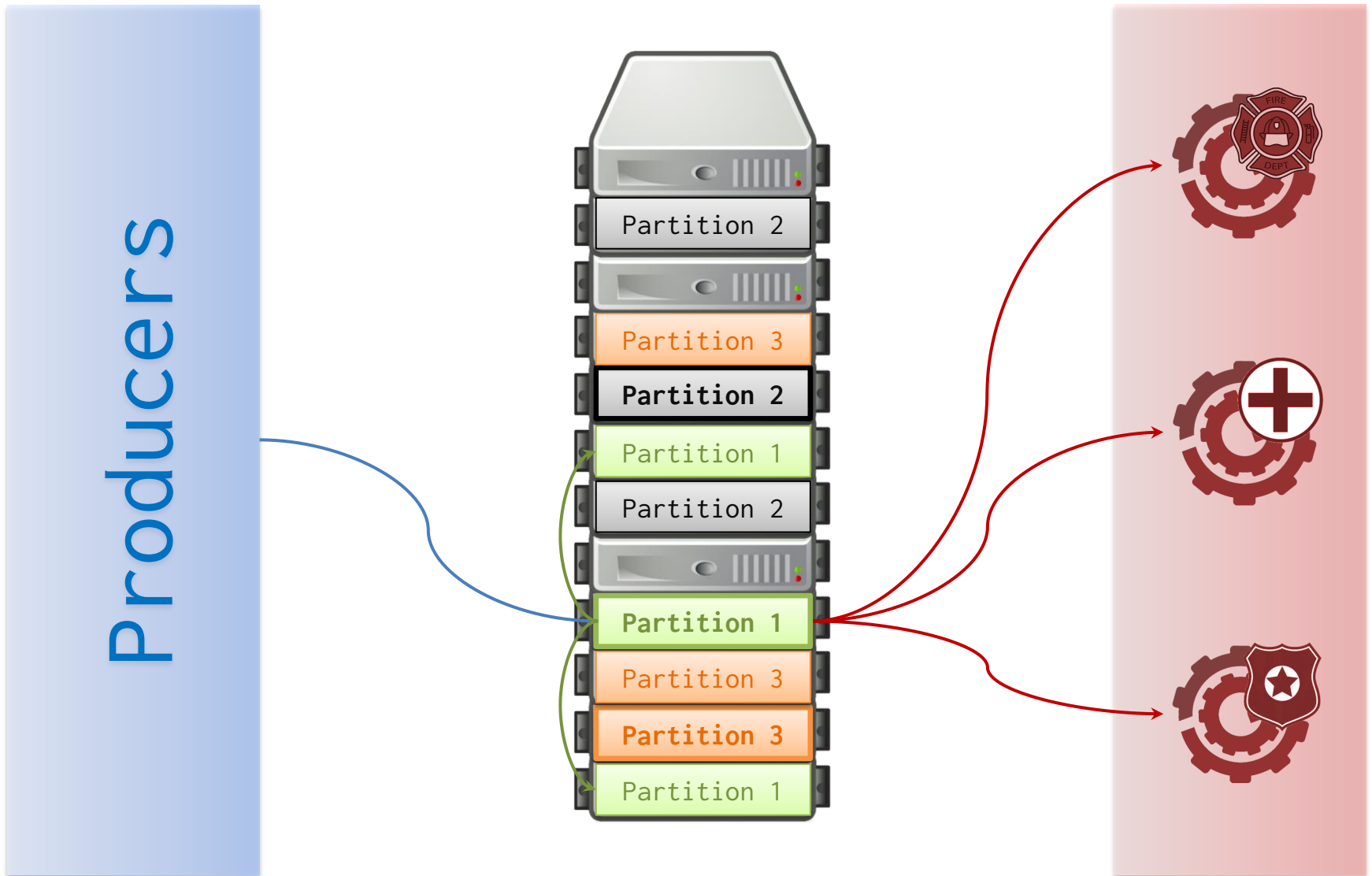
②

①



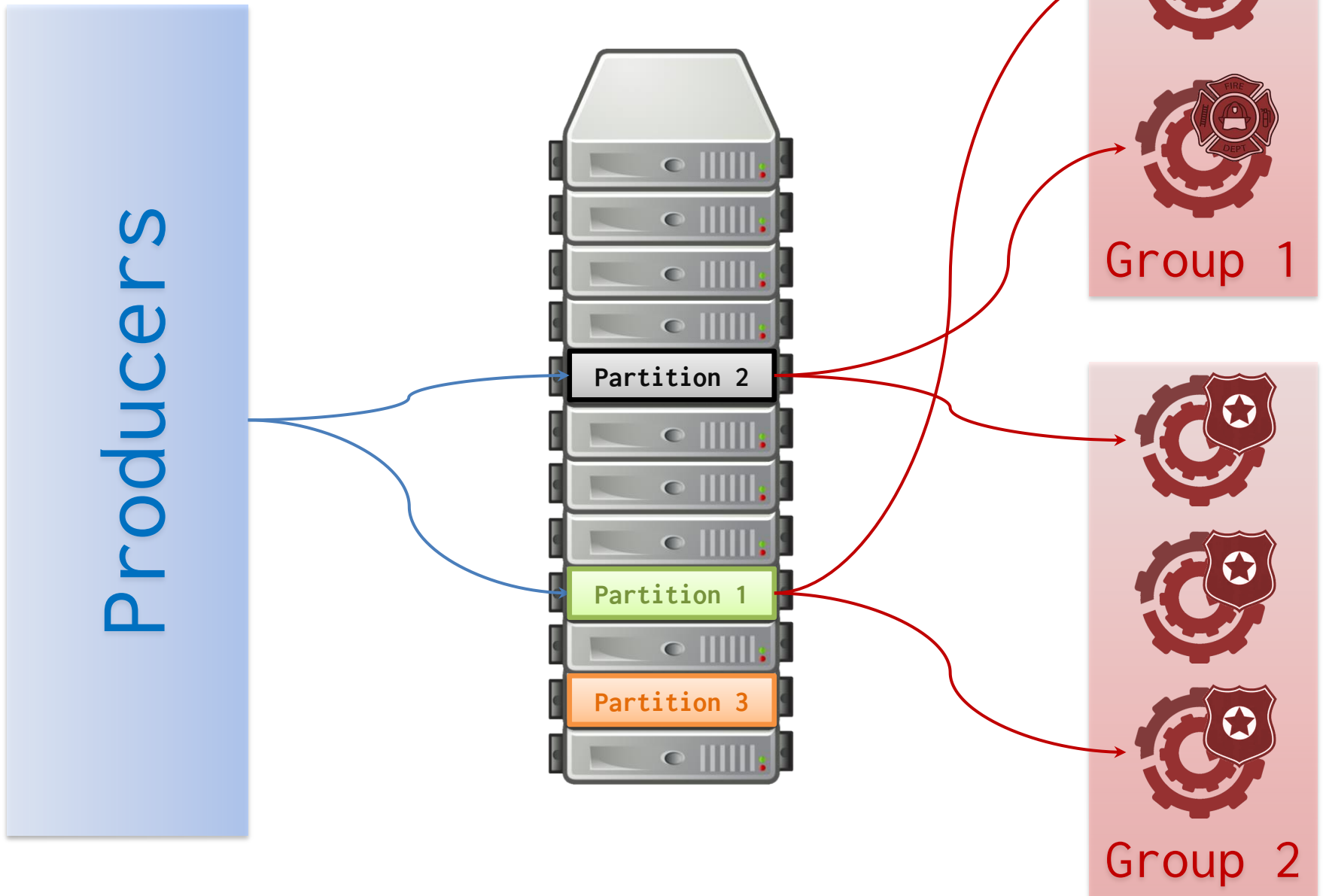
- 1
- 2
- 3
- 4
- 5
- 6

Leader Replication and Reads



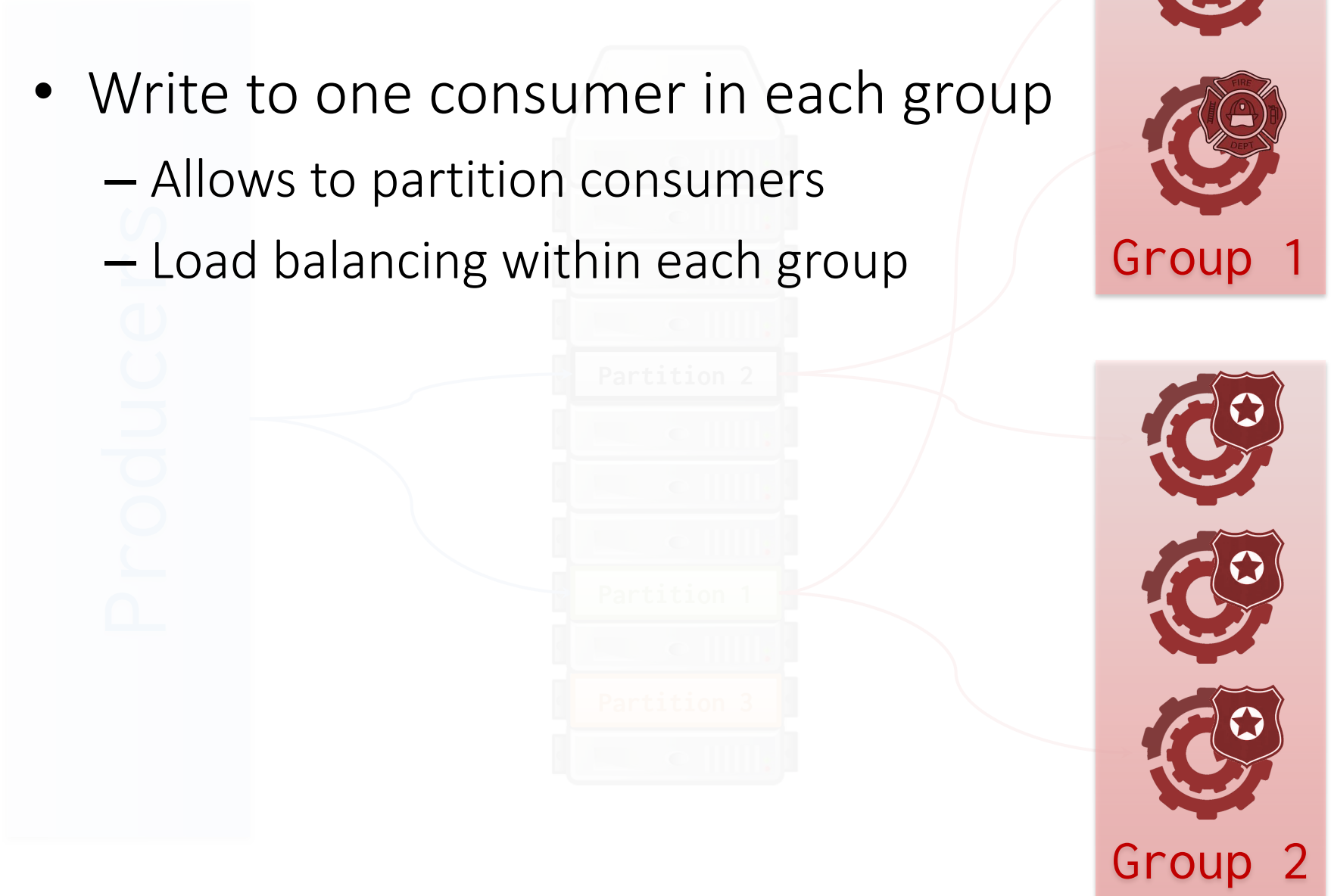
KAFKA: CONSUMER GROUPS

Consumer Groups



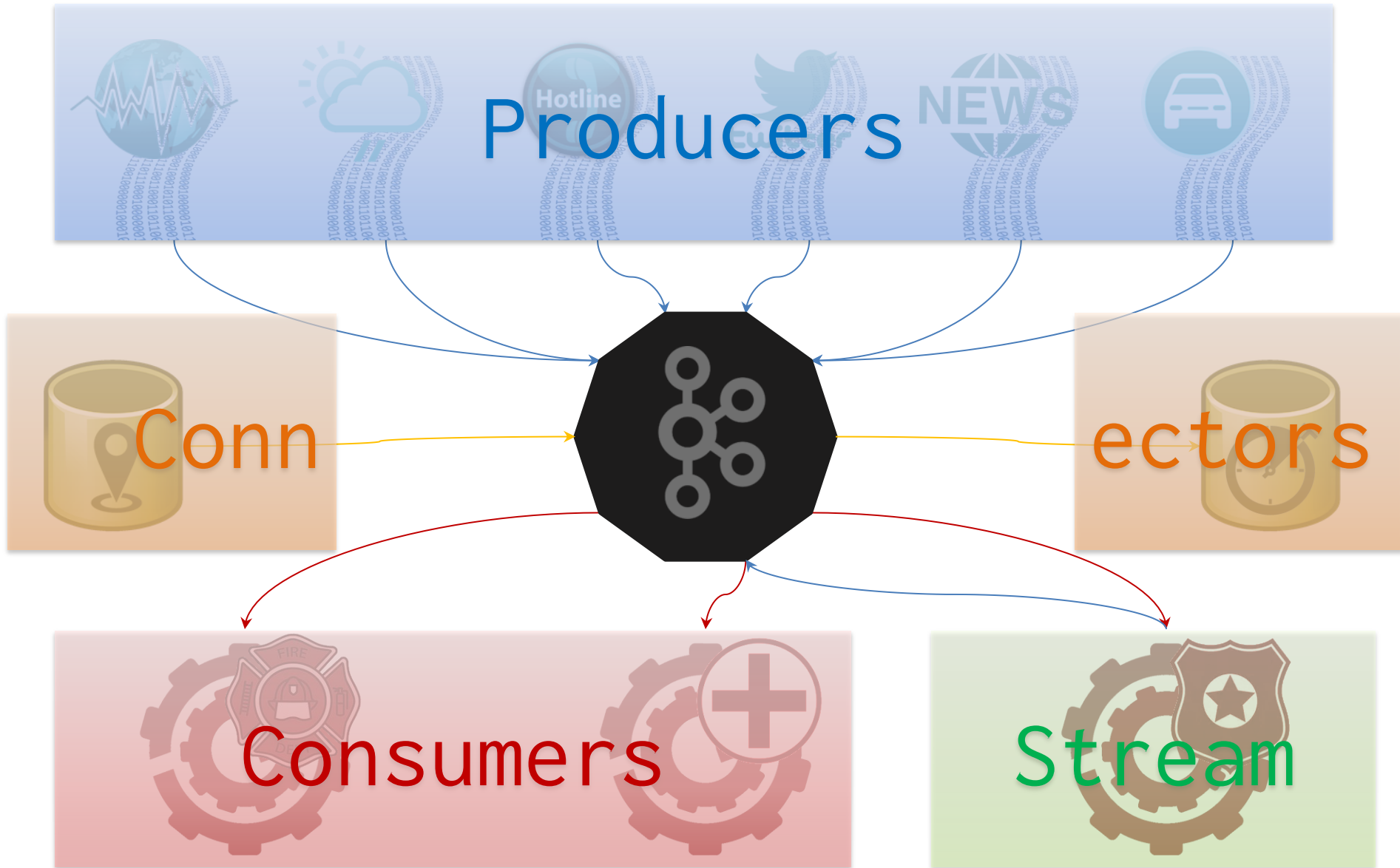
Consumer Groups

- Write to one consumer in each group
 - Allows to partition consumers
 - Load balancing within each group



KAFKA: STREAMS AND CONNECTORS

Kafka Overview



Kafka Overview

- **Producer API:**
 - Append records to topics (push)
- **Consumer API:**
 - Read records from topics (pull)
- **Connector API:**
 - Read/write to external components
 - For example, a database or other streaming platforms
- **Stream API (Producer + Consumer):**
 - Read records from input topics
 - Append records to output topics

OPTIMISATIONS AND OTHER FEATURES













Kafka Optimisations

- **Log Compaction**
 - Repeated sequential values are suppressed
- **Direct Disk-to-Network**
 - When data don't need to be loaded into JVM
- **Consumer / Producer Quotas**
 - Set limits to avoid saturating the system
- ...

Kafka Streams API

- **Aggregation** (e.g., count messages)
- **Joins** (e.g., "unify" two streams)
- **Windowing** (define retention period)
- **Continuous Querying** (KSQL)

Available Frameworks

												
	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 2016)	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	838	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	IgniteDataStream	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 Scala	Java	Scala	Java	Java	Java
API languages	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	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 Localitys AsialInfo Opentable Fairdata 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.bitbucket.io/an-overview-of-apache-streaming-technologies/>



Questions?