CQRS

PROCESSING EVENTS

Martijn Blankestijn
@MartijnBlankest

CODE.STAR

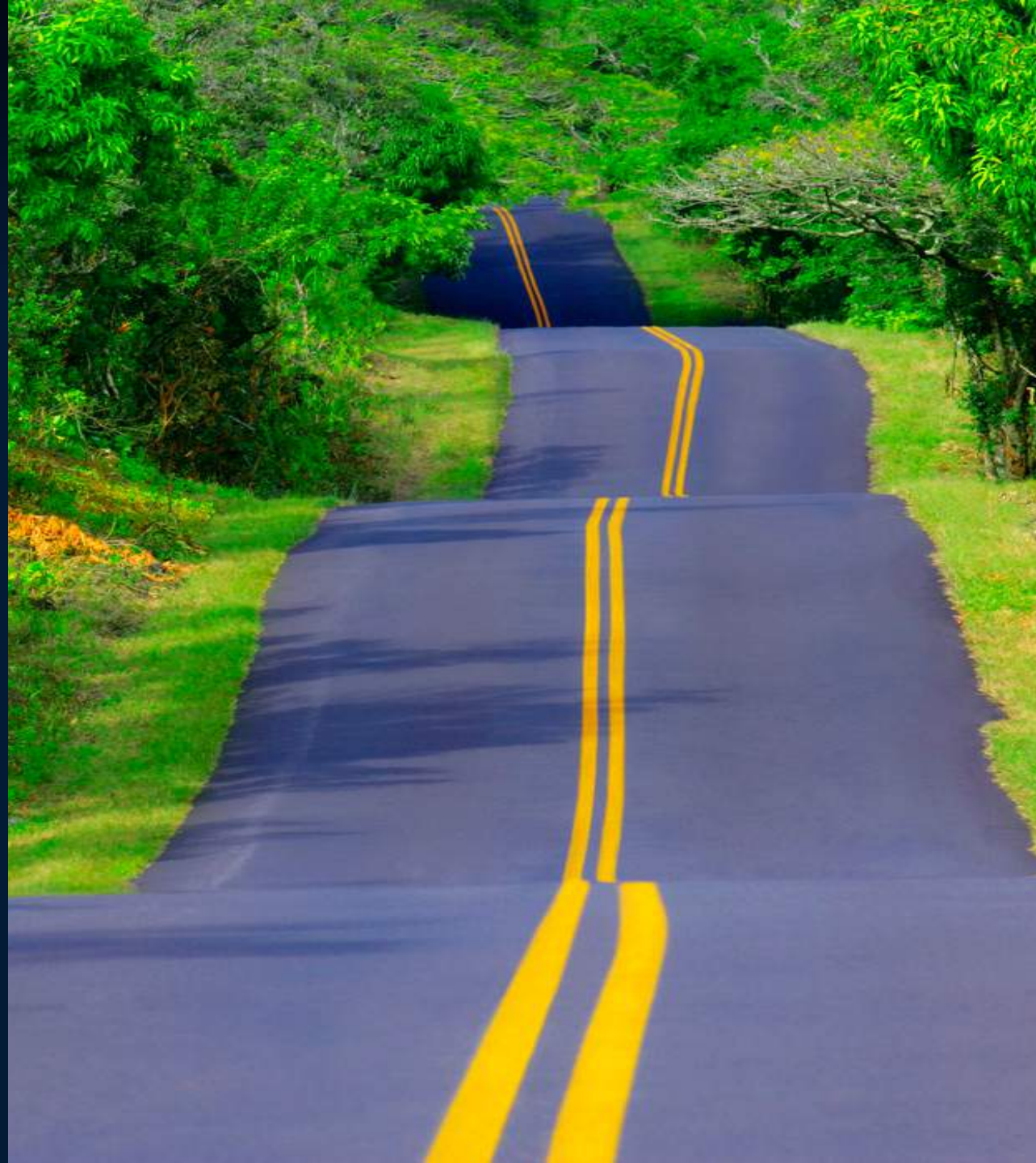
ING

Roadmap

Why this talk

Event Sourcing & CQRS

Query side processing



Why?

Appointments

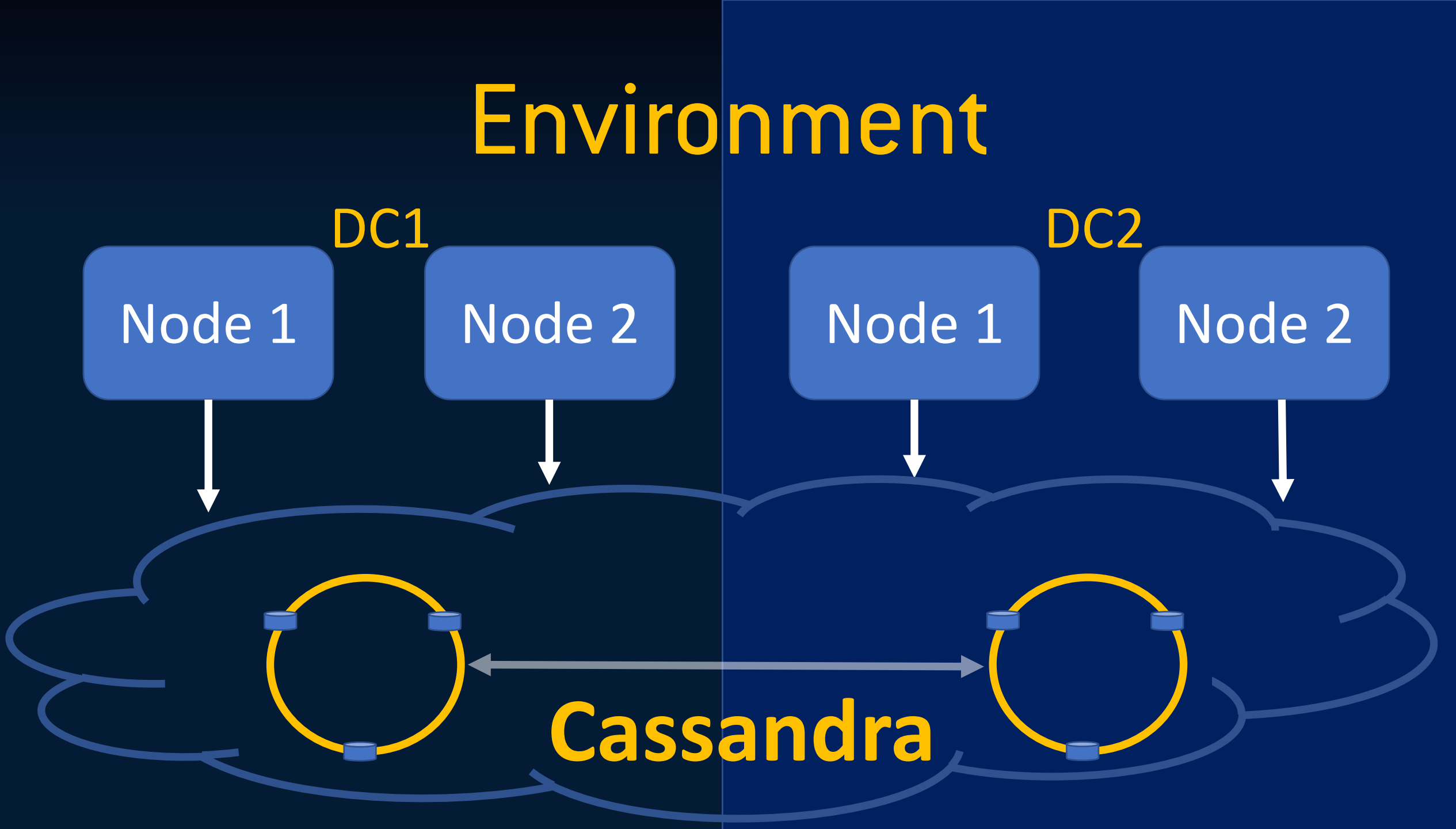
Make

Move

Reassign

Conclude

Cancel


Environment
DC1
DC2
Node 1
Node 2
Node 1
Node 2
Cassandra

Cassandra

Model around your queries

- determine what queries to support
- create a table for that query … 1 partition

Geo. W. Shuster

Dr.		Cr.	
1897 [illegible]	32 00	June 30 [illegible]	17 00
[illegible]	10 00	" [illegible]	32 10
" 28 [illegible]	23 10	" [illegible]	20 00
To Balance	69 10		69 10
		By Balance	23 10
		July 5 By Cash	65 00
	10 00	" [illegible] Comm	5 90
July 1 To Cash	25 00	" [illegible]	180 00
" 3 [illegible]	67 50	" 7 Cash	15 42
" 3 [illegible]	200 00	" [illegible]	2 00
" 7 [illegible]	175 00	" Downieville	160 00
" 7 [illegible] Bigelow	175 00	" 9 By Cash	80 00
" 12 [illegible] Bigelow	140 00	" 12 " Cash	45 15
17 [illegible] Spencer	85 00	[illegible] 4310	321 00
17 [illegible] Fairfield	135 00	" 19 [illegible]	63 00
17 [illegible]	20 00	" 23 Personal Exps	2 17
23 To Cash	2 50	" Other Exps	15 00
" [illegible]	1035 00	" 24 By Cash	4 90
	29 67	21 By [illegible]	37 43
23 To Balance		23 By Cash L. Shuster	44 00
		23 By [illegible] 2 00	1064 67
	1064 67		
		July 23 By Balance	29 67
		Aug 17 Amt. Due	31 80
		Oct 2 " "	48 20
			109 67
			66 00
Oct 2 To Note	60 00		43 67
" Int	6 00		3 50
	66 00	" 26 Amt Due	47 17
		" 27 By Cash	30 00
			17 17

Clinton P. Hillibes

Bieber Lumber Co.

Dr. 1897		Cr.	
May 31 To Miles #12 170	42 50	1897 Aug 10 By Cash	15 00
July 12 " " #27 170	37 50	" 23 " "	45 00
" " " #32 "	40 00	Oct " "	20 00
" " " #8 "	17 50	" 20 " "	54 00
Sept 22 " " 21 174	54 00	" 25 " "	40
	211 50	Balance	20 45
10% Disc.	21 15		
	190 35		
Extra freight to [illegible]	4 10		
	194 45		194 45
1898 July 1 To Bal	20 45	1898 Aug 25 By Cash	50 00
" 7 Scroll 32 181	40 00	" " " "	60 00
" " #7 Nicholson	14 50	" 26 " "	20 00
#27 Olinger (Bass)	51 50	Allowance for fr'ght on Dorris & Olinger	3 00
Aug #166 2nd Size	100 00		
	226 45	Sept 12 By Cash	30 00
10% Disc on 206.00	20 60	May 4 " "	15 00
	205 85	Sept 14 " "	30 00
#2 Baker (Special Price 181)		Oct 21 By Cash	110 00
#2 Mrs Mess	25 00		
#1416 1/2 Dorris	51 00	Oct 30 " G. Morse	90 00
	281 85		
May #1, C. 16.00 Dis 1.60	14 40	1899 Apl By Pasturage Mare	7 50
	296 25	1900 Feb 16 By Cash	10 00
		Feb 21 Bal	32 00
Sept 23 #1409 1/4 [illegible]	85 00		
" 30 #27 (J. R. Co.)	50 75		
" #4031 182	45 50		
	477 50		477 50
1900 Feb 21 To Bal.	32 00		

Event Sourcing

'ensures that all changes

to application state

are stored as a

sequence of events.'

https://martinfowler.com/eaaDev/EventSourcing.html

Time

Events

Created

10:00 – 12:00 9th Nov
with Smith
in Amsterdam

Current State

10:00 – 12:00 9th Nov
with Smith
in Amsterdam

Time

Events

Created
10:00 – 12:00 9th Nov
with Smith
in Amsterdam

Moved
14:00 – 16:00 2th Nov
with Jones
in Amsterdam
'Bid has been made'

Current State

14:00 – 16:00 2th Nov
with Jones
in Amsterdam
'Bid has been made'

Time

Events

Created
10:00 – 12:00 9th Nov
with Smith
in Amsterdam

Moved
14:00 – 16:00 2th Nov
with Jones
in Amsterdam
'Bid has been made'

Concluded
5-star
'product sold'

Current State

14:00 – 16:00 2th Nov
with Jones
in Amsterdam
Comment: Bid has been made
Concluded(5-star, product-sold)

Event Sourcing

Built-in audit log

Troubleshooting

Space requirements

Querying entities

Command-Query Responsibility Segregation

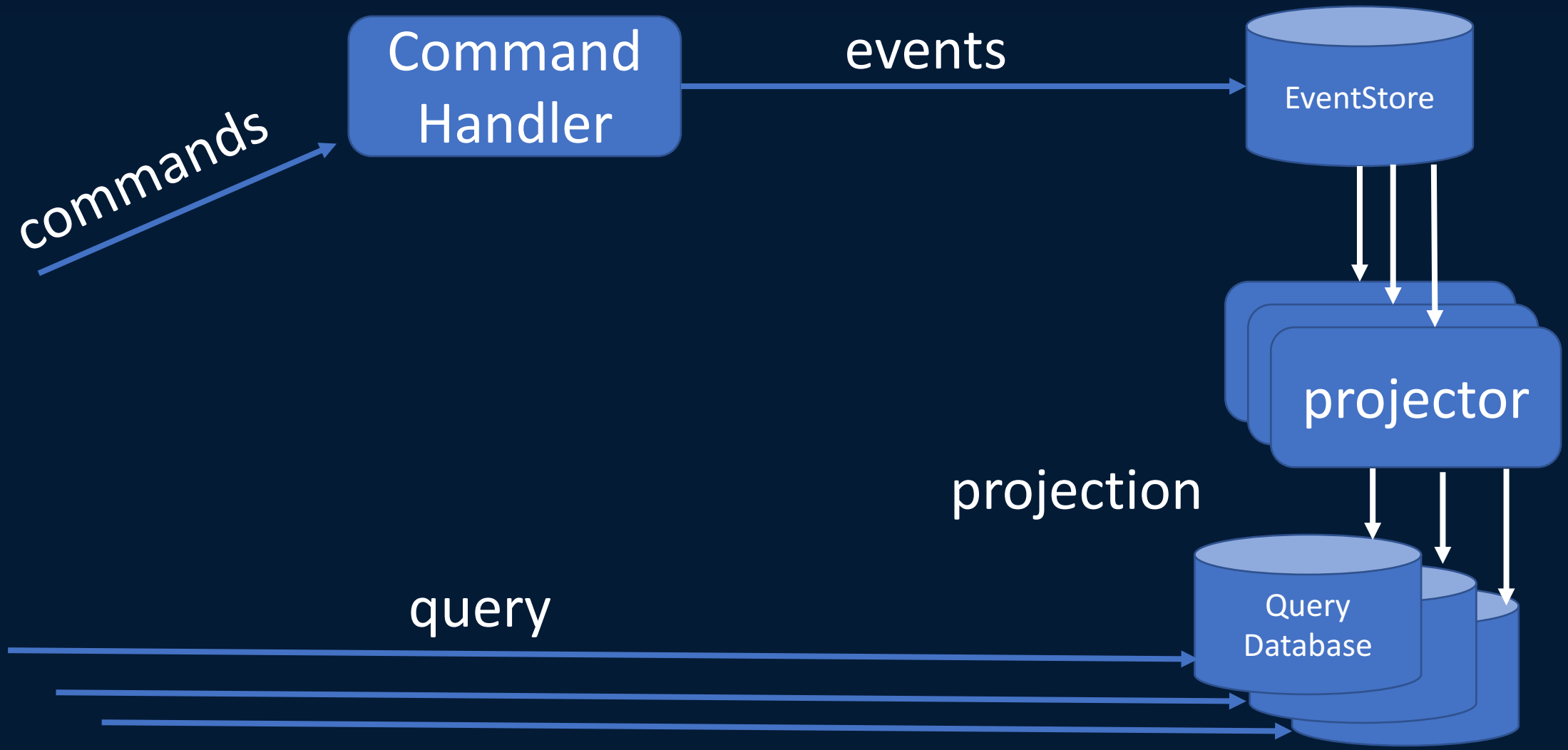
CQRS

Architectural pattern

with driving forces

- Collaboration
- Staleness

The CQRS universe



Advantages CQRS with ES

Scale read/write independently

Scale Query side per use-case


CHOOSE

‘CQRS with Event Sourcing’ Frameworks

Kafka as event store

Axon Framework

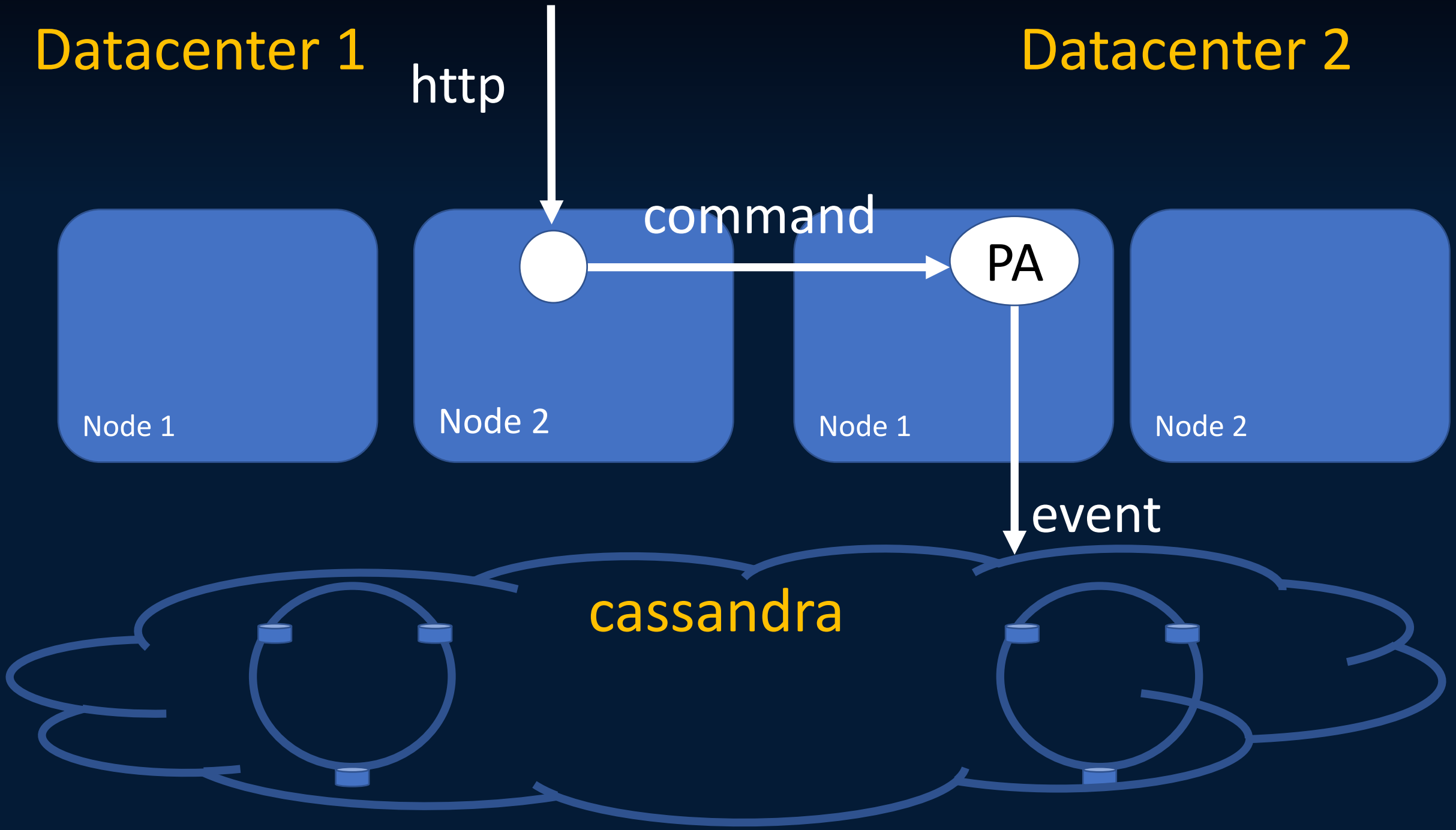
Eventuate

Akka Persistence

Persistence

Persistent Actor

Journal

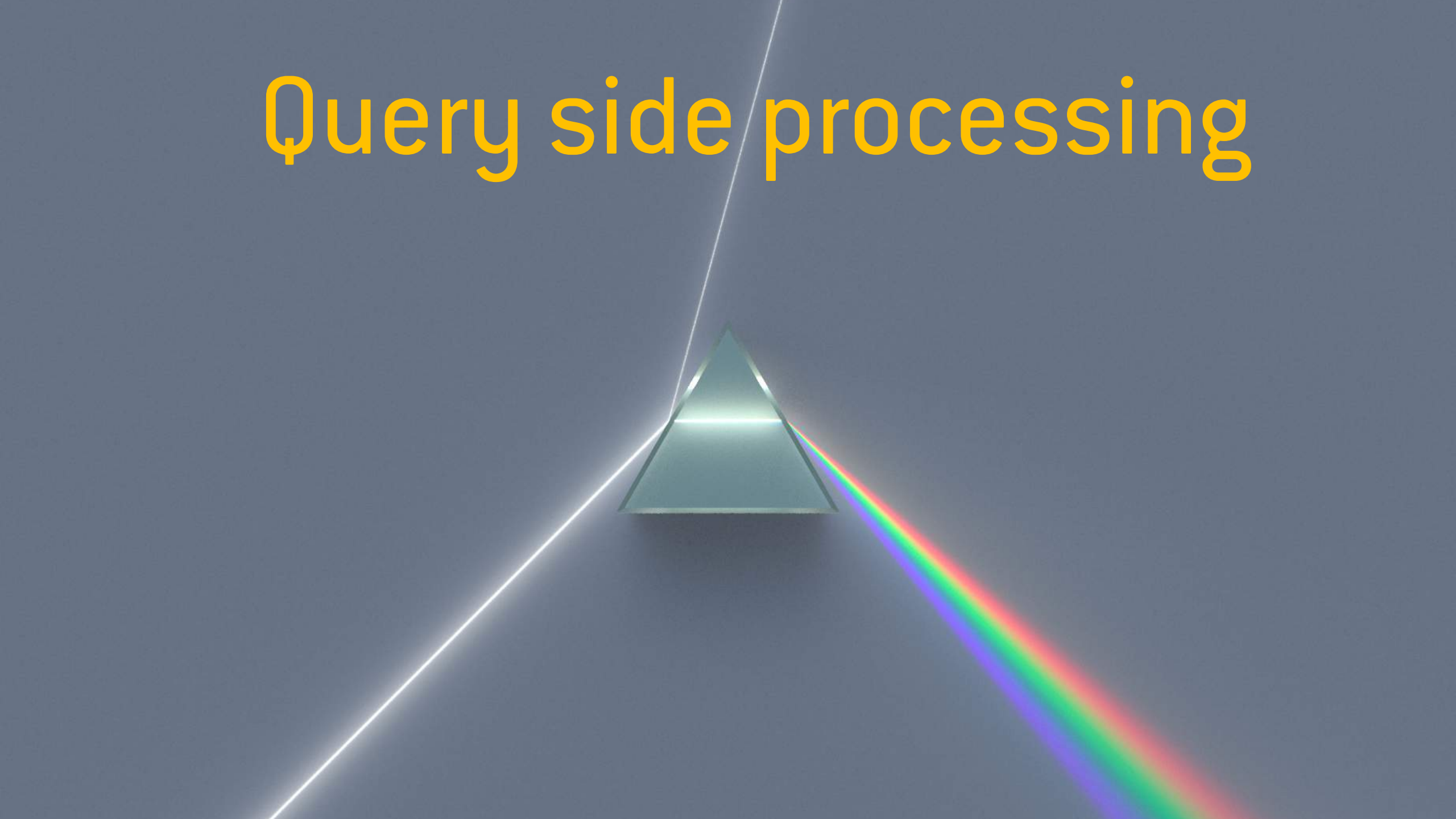

Datacenter 1
http
Datacenter 2
command
PA
Node 1
Node 2
Node 1
Node 2
event
cassandra

Cassandra Events table

persistence_id	partition_nr	sequence_nr	timestamp	timebucket
7c7ec816-efc6...	0	1	8ef7f9...	20171018
7c7ec816-efc6...	0	2	99e314...	20171018
7c7ec816-efc6...	0	3	a41f2b...	20171018

tag1	writer_uuid	ser_id	ser_manifest	event
appointment	f0088eec...	2	nl...Created	[payload]
appointment	f0088eec...	2	nl...Reassigned	[payload]
appointment	f0088eec...	2	nl...Moved	[payload]

Query side processing



Using Persistence Query

Using Persistence Query

```
PersistenceQuery(system)
  .readJournalFor[LeveldbReadJournal](
                                  Identifier)
  .eventsByTag("appointment")
  .map(println)
  .runWith(Sink.ignore)
```

Considerations read side

Resumability

Event order

Changing requirements

Scalability

Push or Pull (*)

Resumability

Resumability


```
val offset = readOffset().getOrElse(noOffset)

PersistenceQuery(system)
  .readJournalFor(Identifier)
  .eventsByTag(tag, offset)
  .map(processEvent)
  .map(saveOffset)
  ...
```

Eventual Consistency

The same stream elements

(in same order) are returned

for multiple executions of the query

on a best effort basis.

https://static.javadoc.io/com.typesafe.akka/akka-persistence-cassandra_2.12/0.58/akka/persistence/cassandra/query/scaladsl/CassandraReadJournal$.html

id	seq	timestamp
A	1	04.709
B	1	04.731
C	1	04.801

eventual-consistency-delay=100ms

id	seq	timestamp
A	*1*	*04.709*
B	*1*	*04.731*
C	1	04.801
A	3	04.824
C	2	04.957

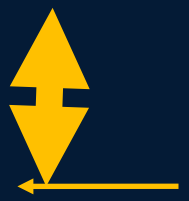
delayed-event-timeout = 1000ms

STOP

NOW (920)

eventual-consistency-delay=100ms

id	seq	timestamp
A	*1*	*04.709*
B	*1*	*04.731*
A	2	04.768
C	*1*	*04.801*
A	3	04.824
C	2	04.957
A	4	04.973



NOW (05.063)

NO ONE WANTS

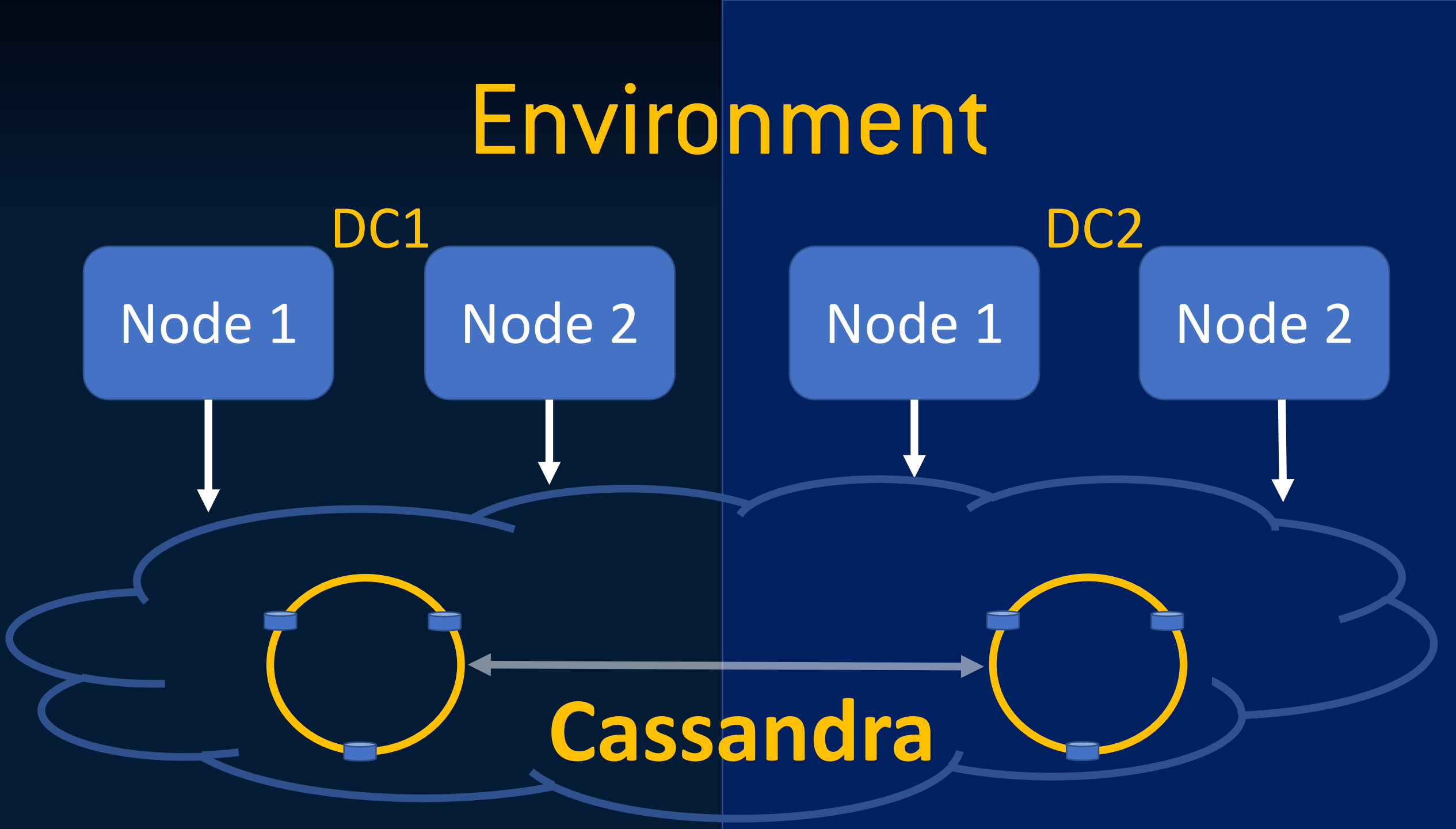
EVENTUAL CONSISTENCY.

IT'S A NECESSARY EVIL.

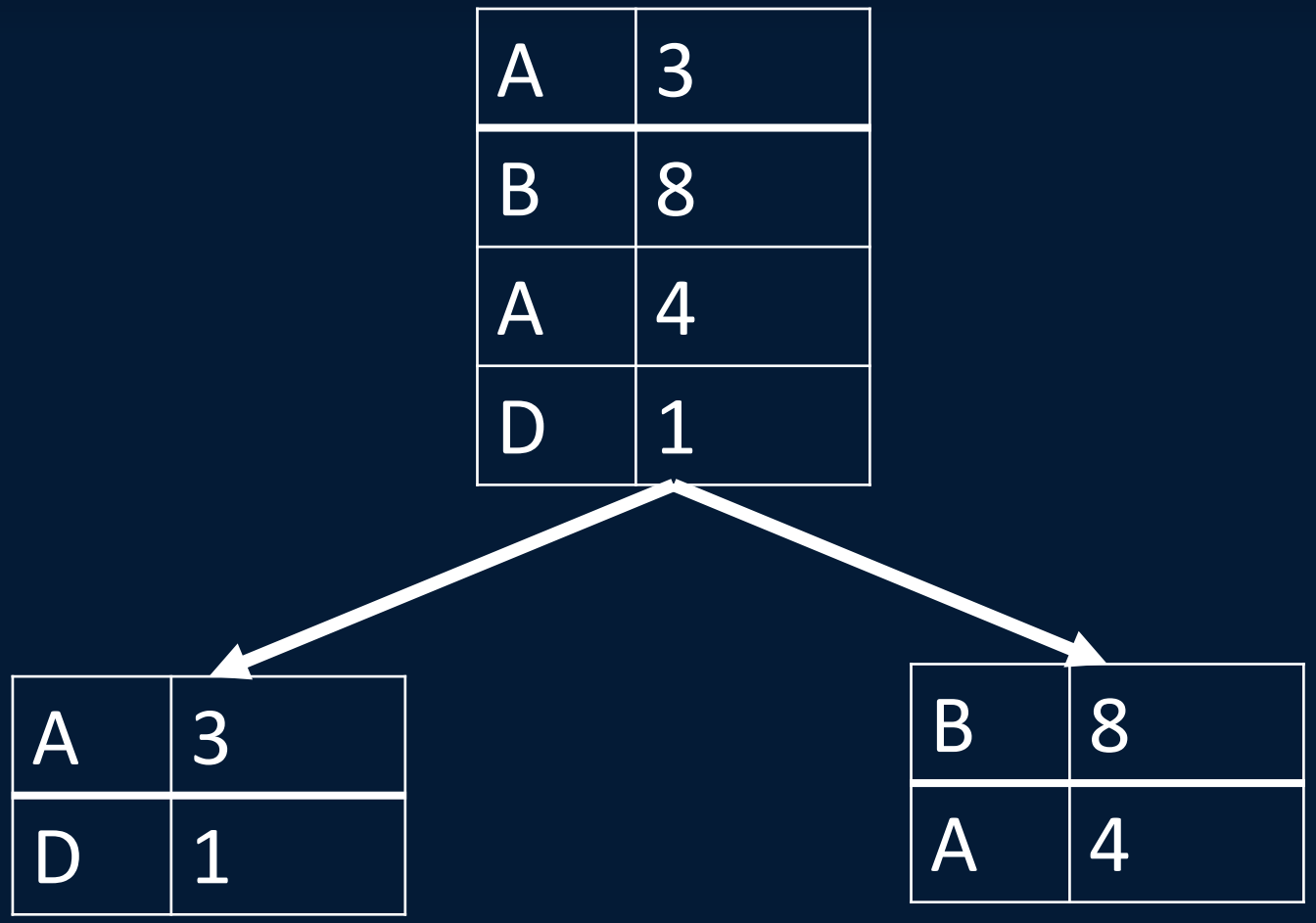
IT'S NOT COOL. IT'S USEFUL.

Jonas Bonér

How much latency can you accept ?


Environment
DC1
DC2
Node 1
Node 2
Node 1
Node 2
Cassandra

In parallel



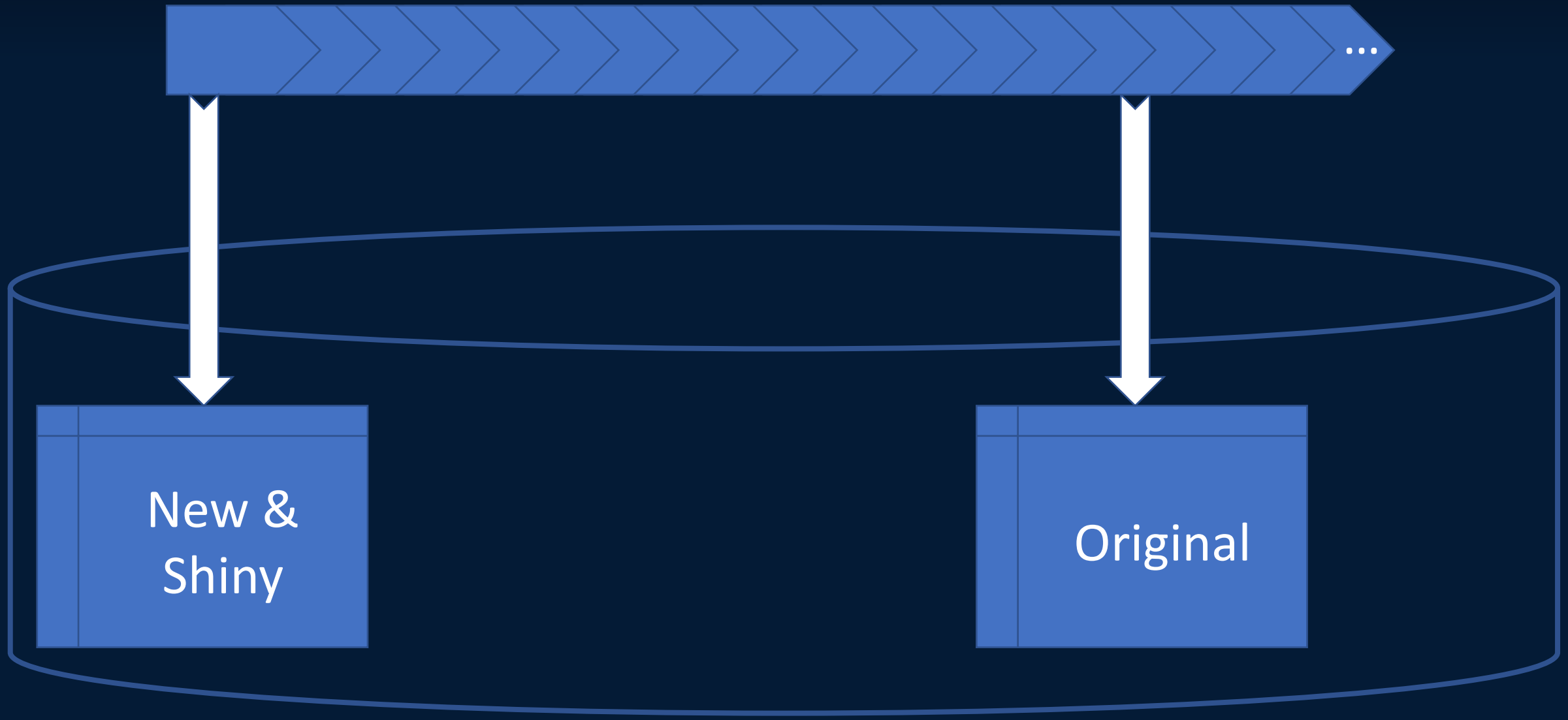

Read-side Sharding

Read side Sharding

- shard upon event data
- shard on the entity id

Changing requirements

Projection upgrade



The Dark Side of Event Sourcing:

Managing Data Conversion

http://files.movereem.nl/2017saner-eventsourcing.pdf

Event store upgrade techniques

Multiple versions

Upcasting

Lazy transformation

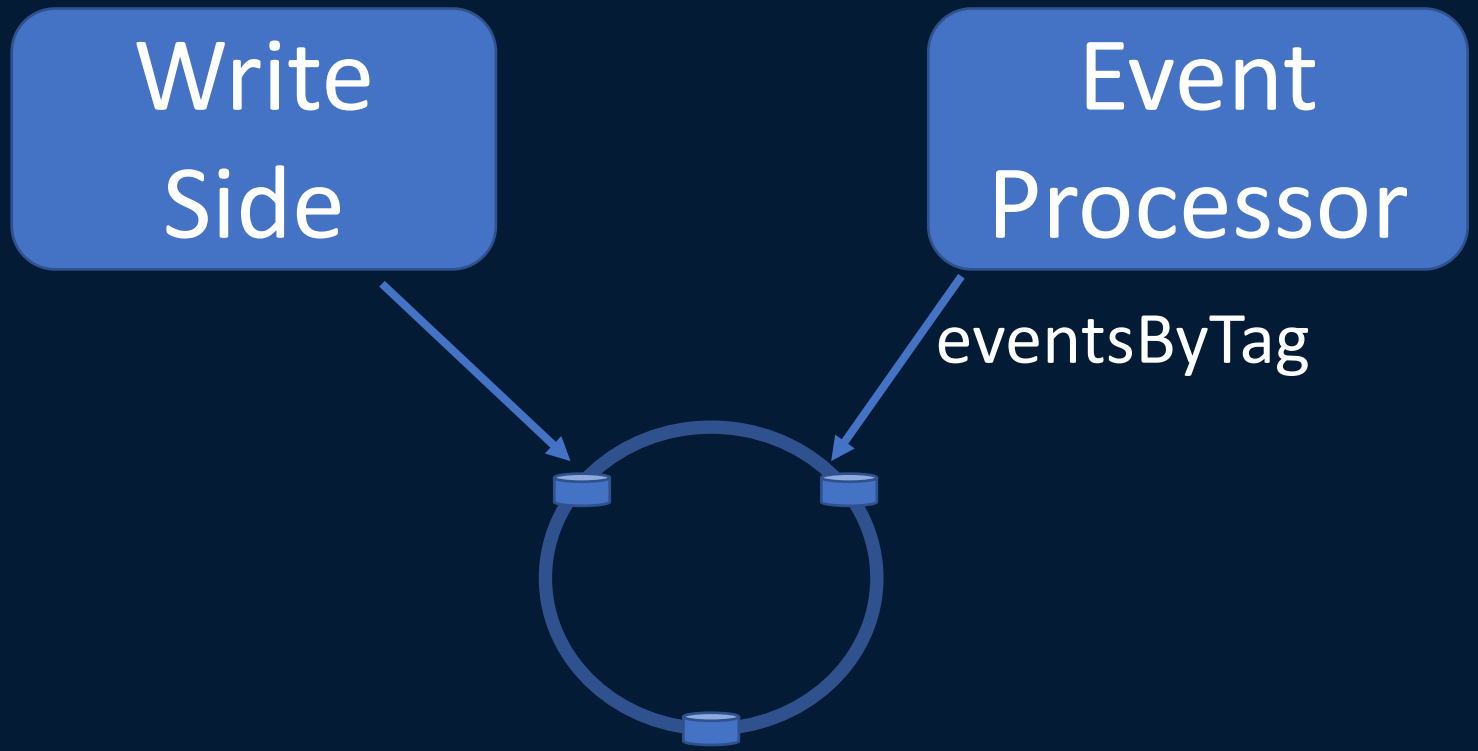
In place transformation

Copy and transformation

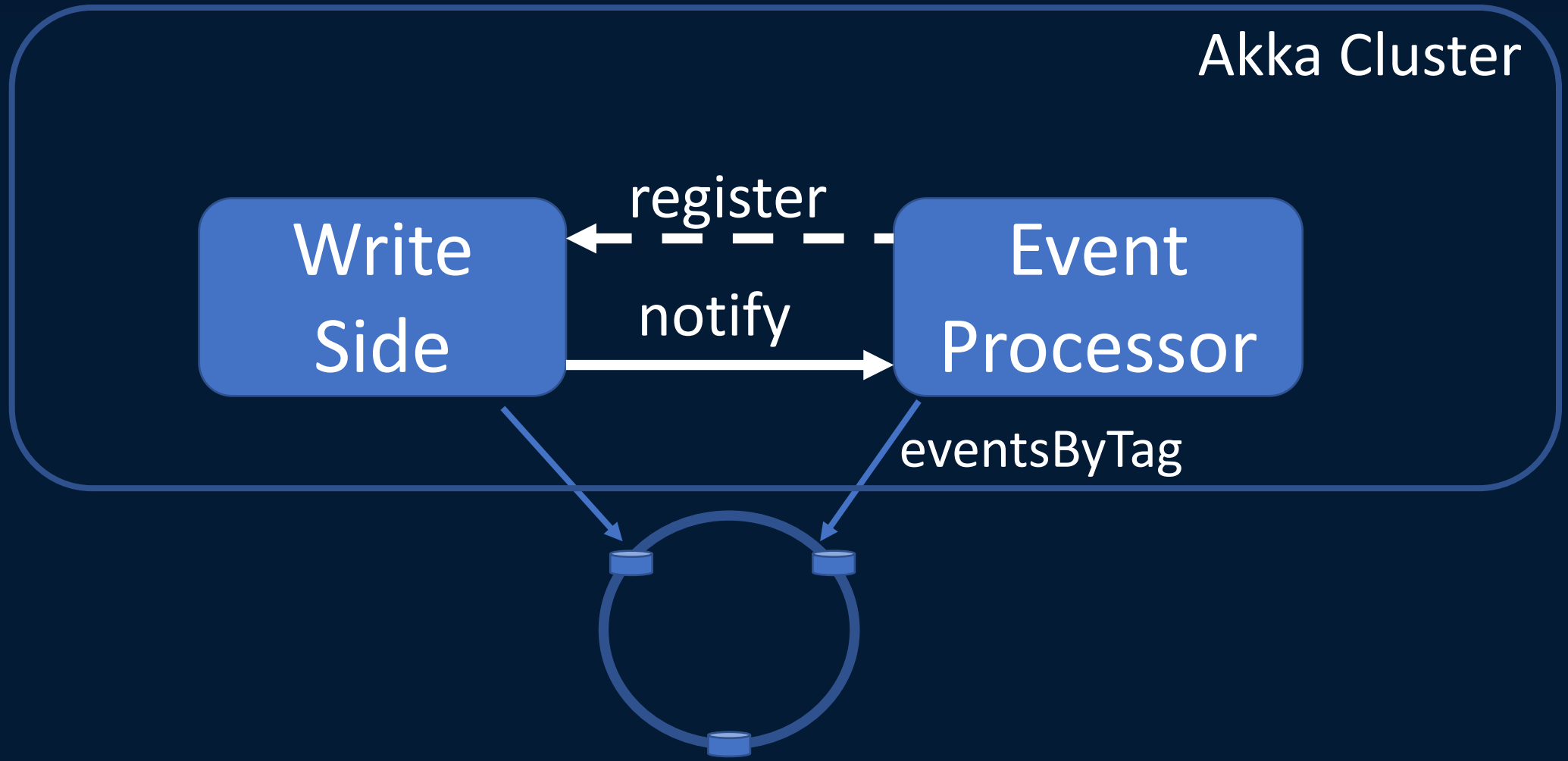
Event store upgrade techniques

	Read-side impact
Multiple versions	-
Upcasting	+
Lazy transformation	+
In place transformation	++
Copy and transformation	++

Pull



Push


Akka Cluster
register
Write Side
notify
Event Processor
eventsByTag

In conclusion

CQRS and Event Sourcing

Query side processing

- resumability
- event order
- requirement changes
- scale
- push vs pull

@MartijnBlankest