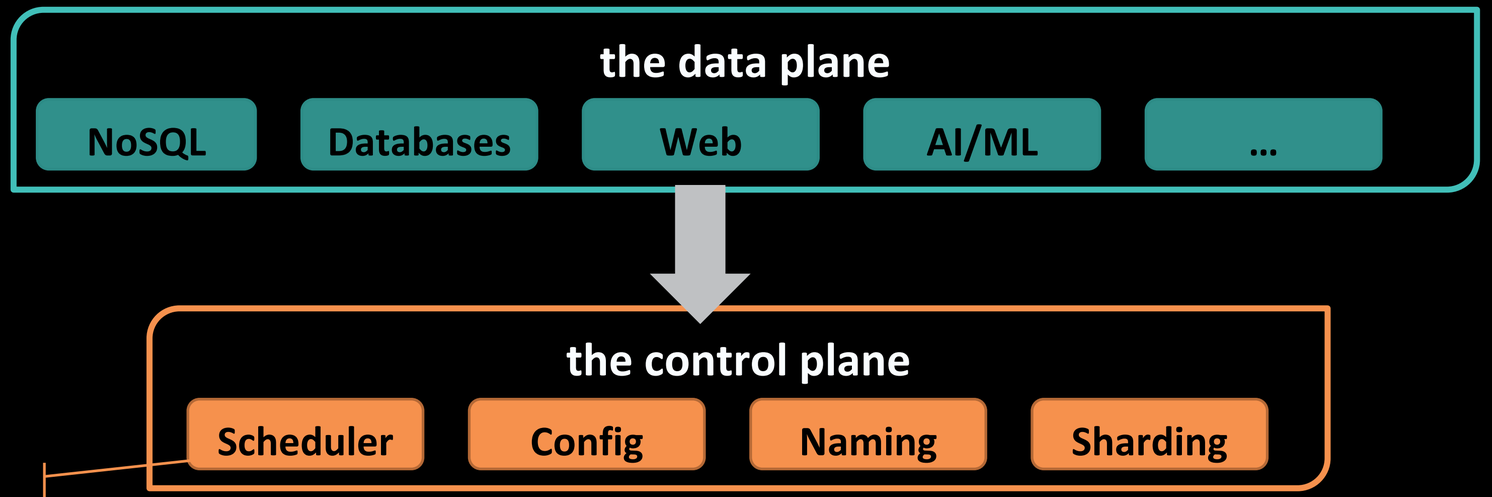
Virtual Consensus in Delos

Mahesh Balakrishnan, Jason Flinn, Chen Shen, Mihir Dharamshi, Ahmed Jafri, Xiao Shi
Santosh Ghosh, Hazem Hassan, Aaryaman Sagar, Rhed Shi, Jingming Liu, Filip Gruszczynski
Xianan Zhang, Huy Hoang, Ahmed Yossef, Francois Richard, Yee Jiun Song

Facebook, Inc.

the Facebook stack

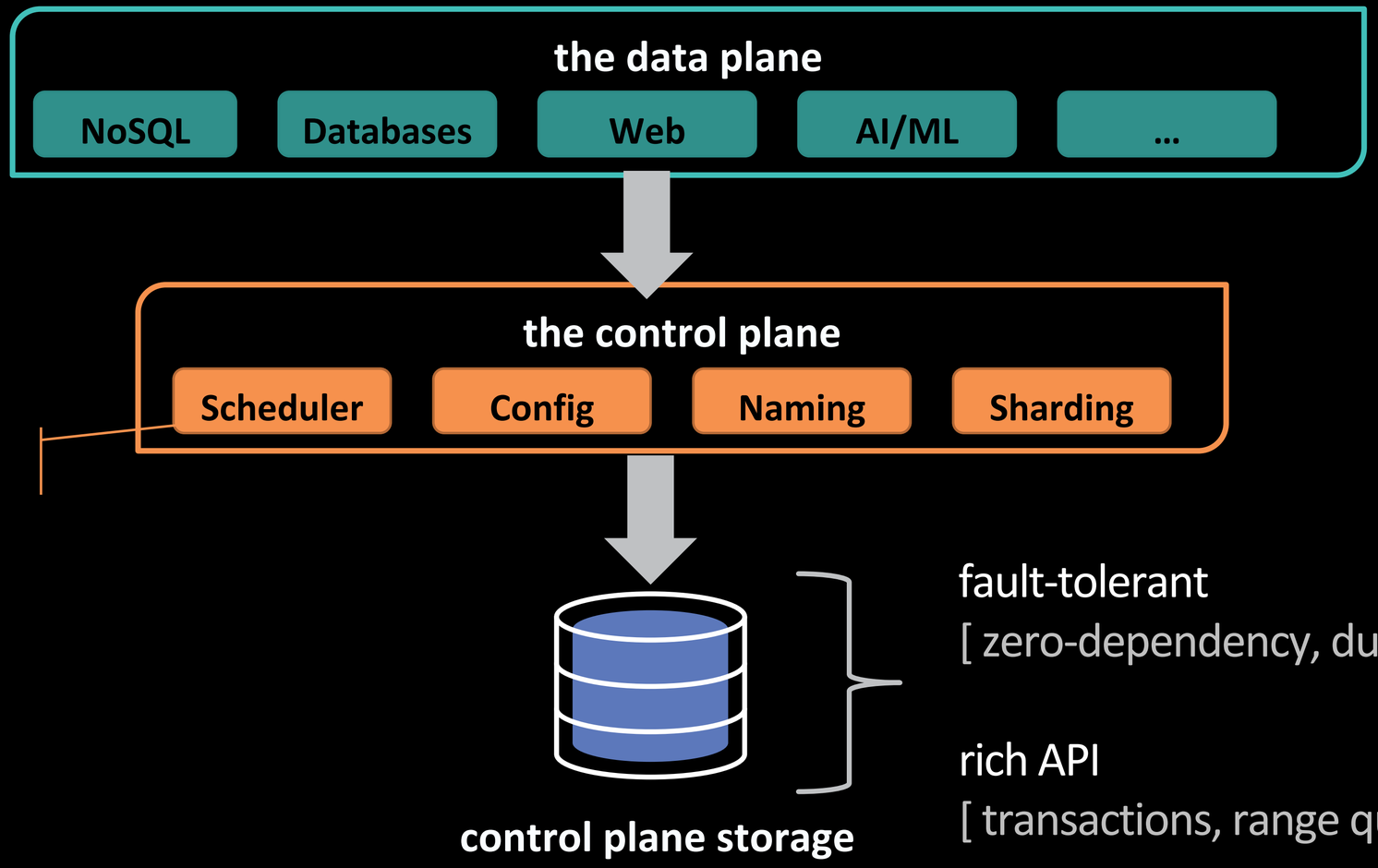
turtles all the way down…



[Twine, OSDI 2020]

the Facebook stack

turtles all the way down…



fault-tolerant

[zero-dependency, durable, highly available]

rich API

[transactions, range queries, secondary indices]

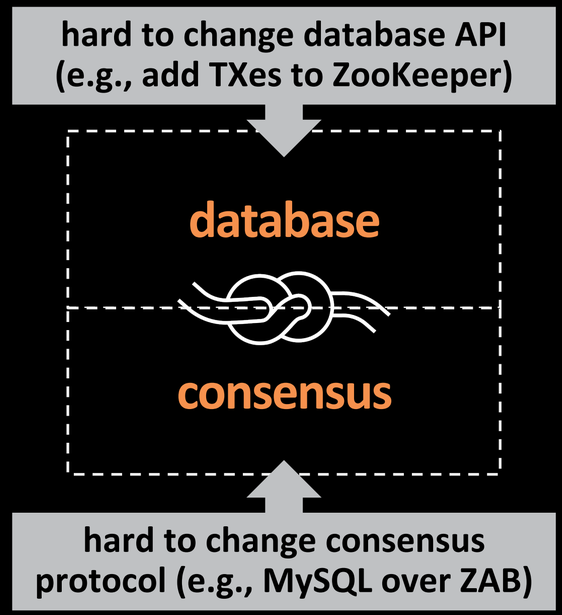
the need for a new storage system

why not use an existing system?

the need for a new storage system

why not use an existing system?

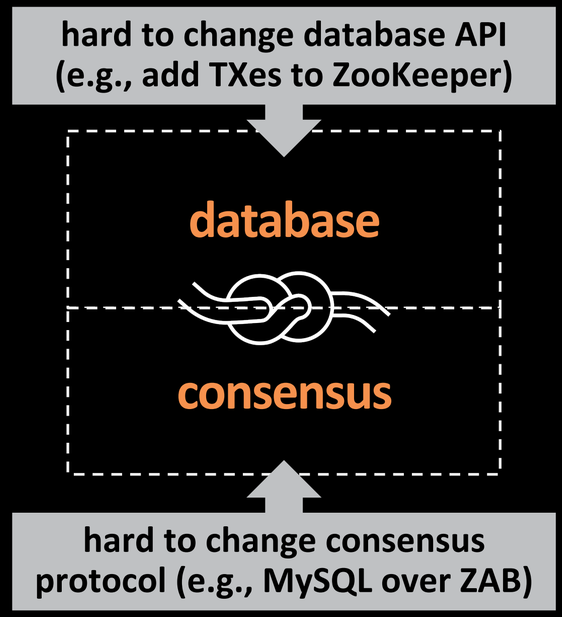
why not modify an existing system?



the need for a new storage system

why not use an existing system?

why not modify an existing system?



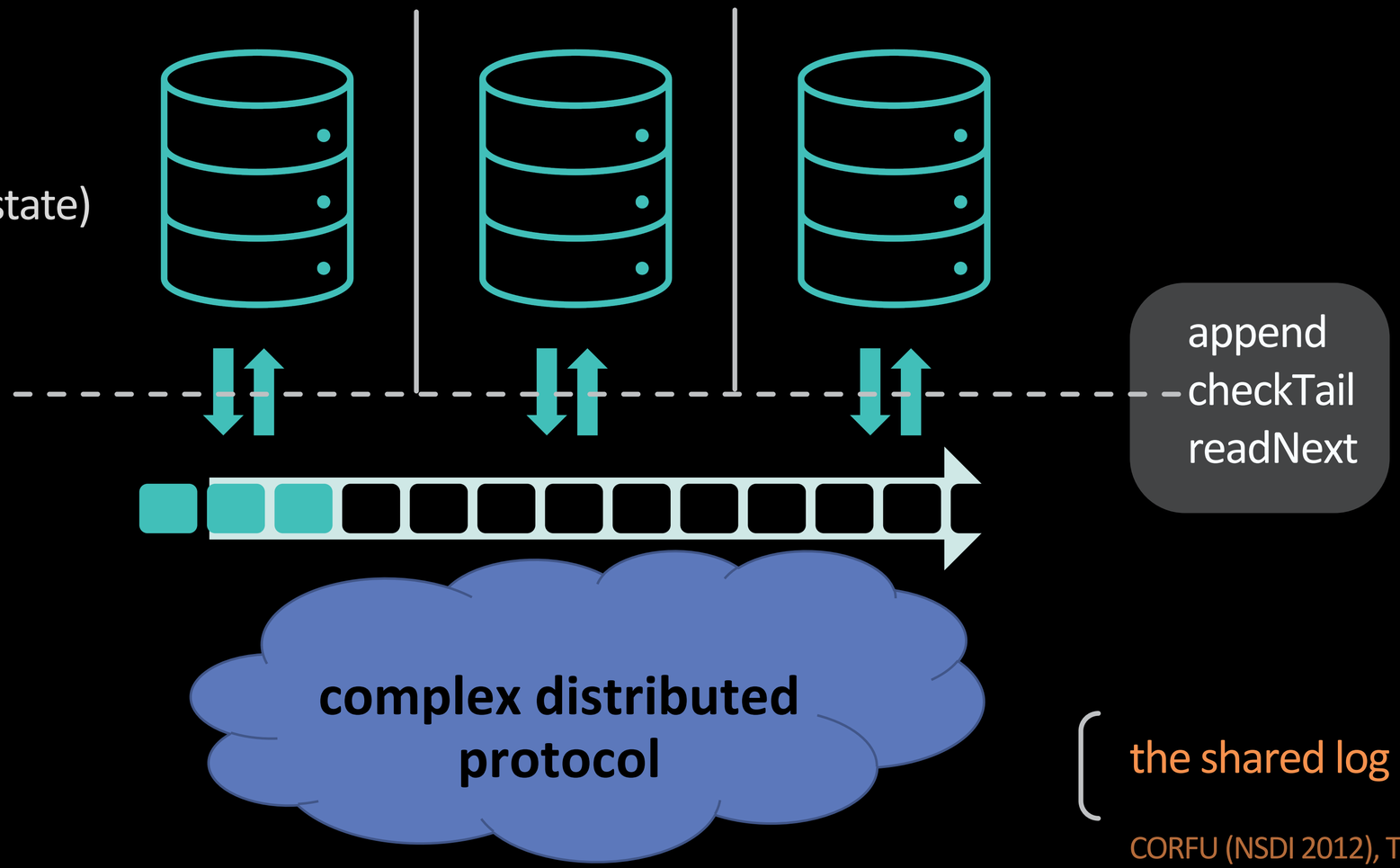
problem statement circa 2017:
can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*

the Delos storage system

or: “how to build a production-ready storage system in eight months”

the Delos storage system

or: “how to build a production-ready storage system in eight months”



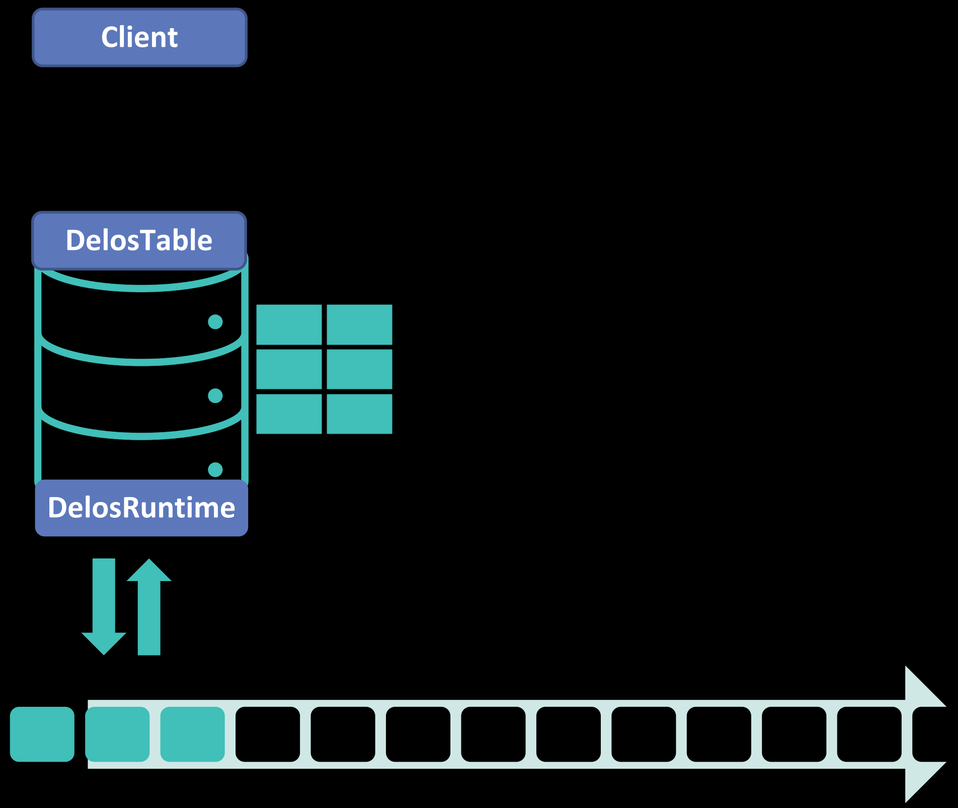
the shared log is an API for consensus

CORFU (NSDI 2012), Tango (SOSP 2013), Hyder (SIGMOD 2015), CorfuDB, LogDevice, Scalog (NSDI 2020)...

the Delos storage system: above the log

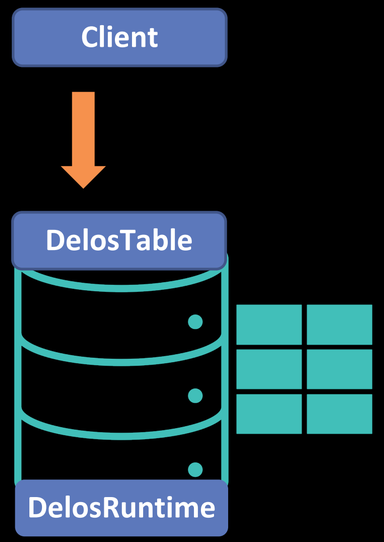
database
(materialized state)

shared log
(consensus)

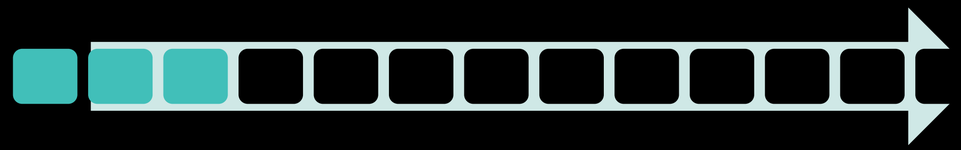


the Delos storage system: above the log

database
(materialized state)

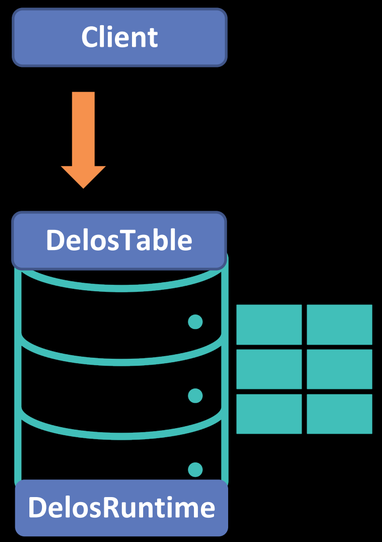


shared log
(consensus)

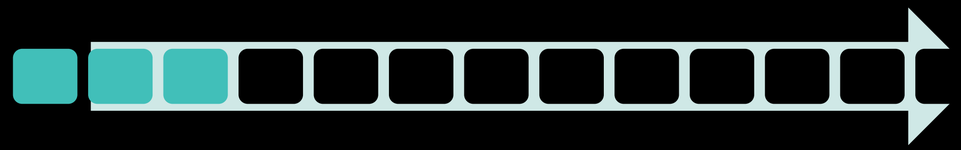


the Delos storage system: above the log

database
(materialized state)

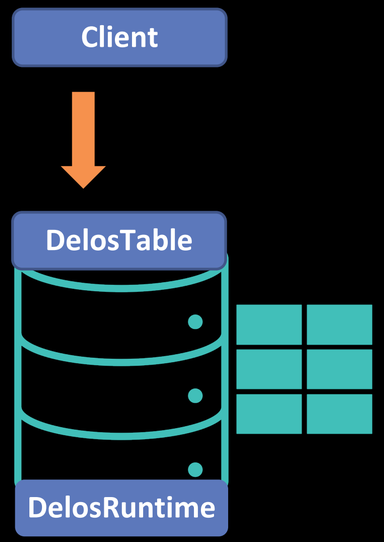


shared log
(consensus)

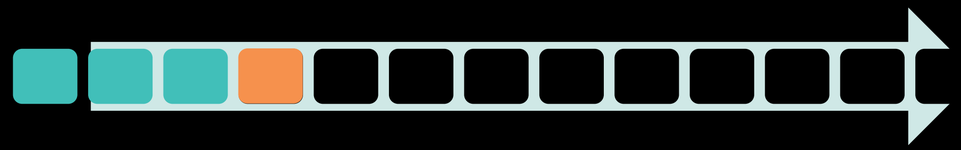


the Delos storage system: above the log

database
(materialized state)

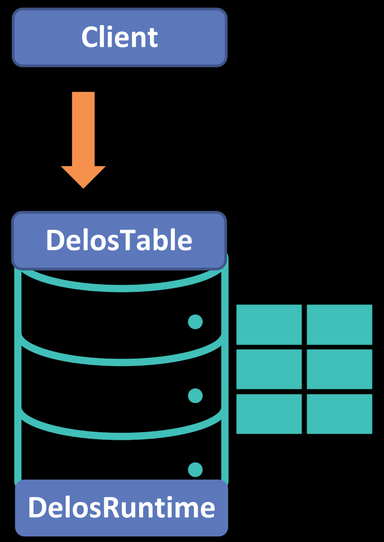


shared log
(consensus)

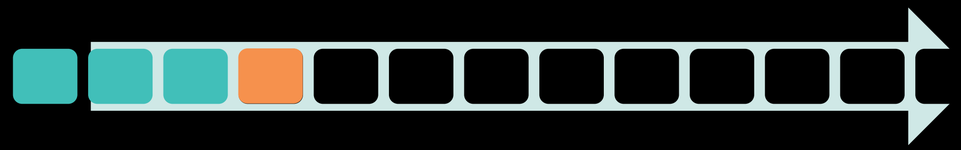


the Delos storage system: above the log

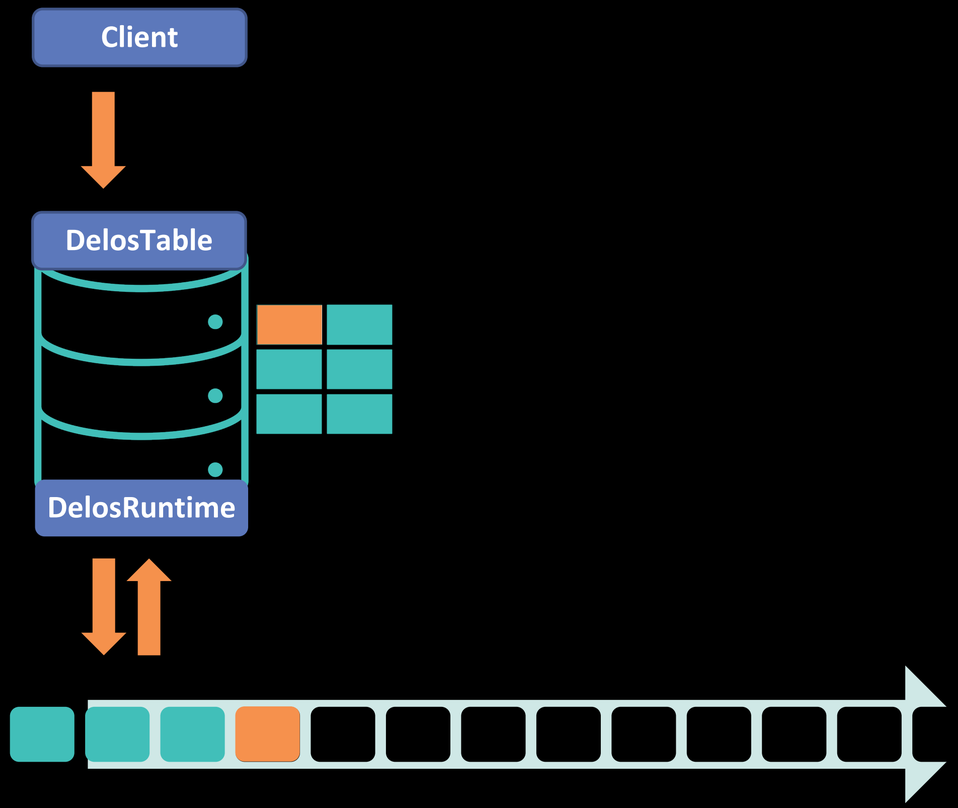
database
(materialized state)



shared log
(consensus)



the Delos storage system: above the log



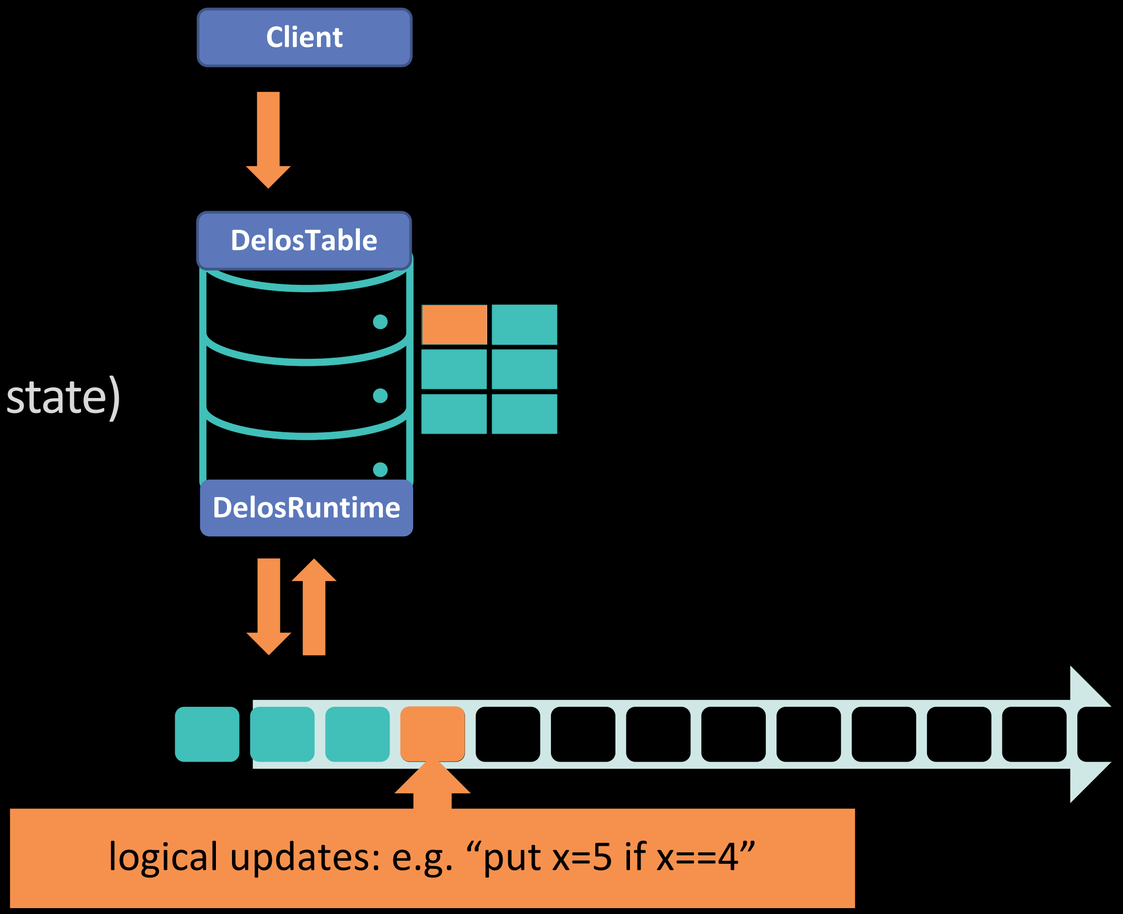
database
(materialized state)

shared log
(consensus)

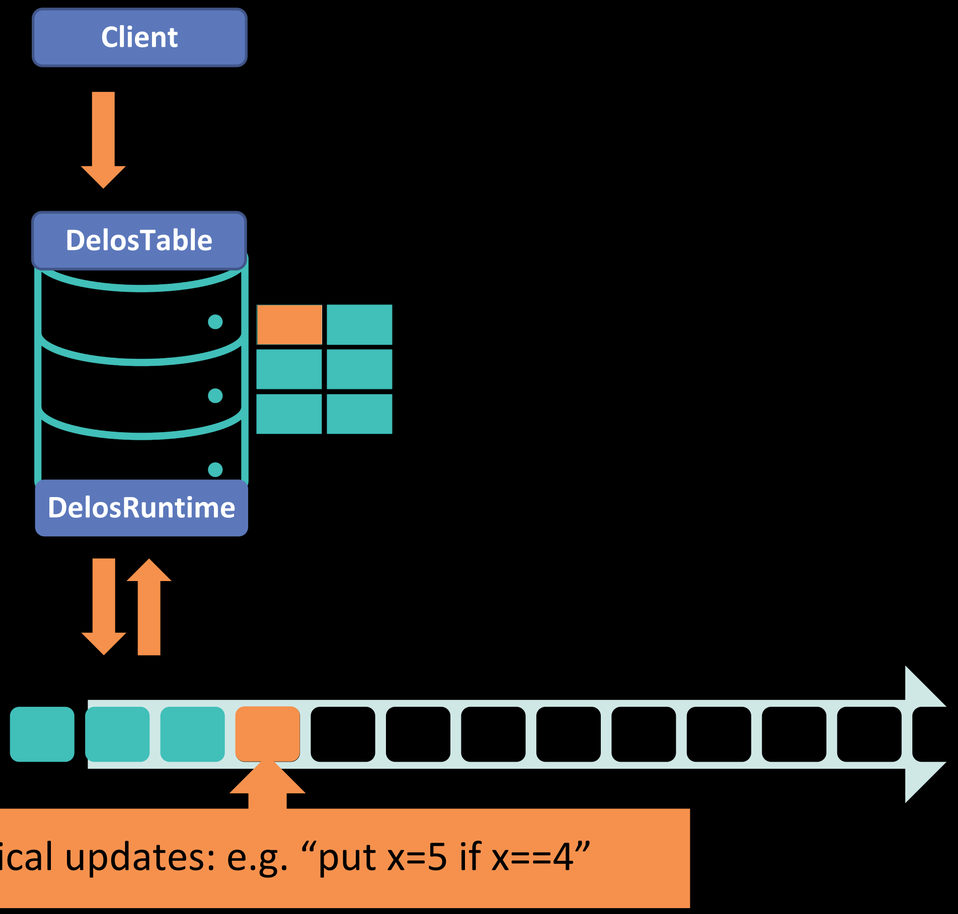
the Delos storage system: above the log

database
(materialized state)

shared log
(consensus)

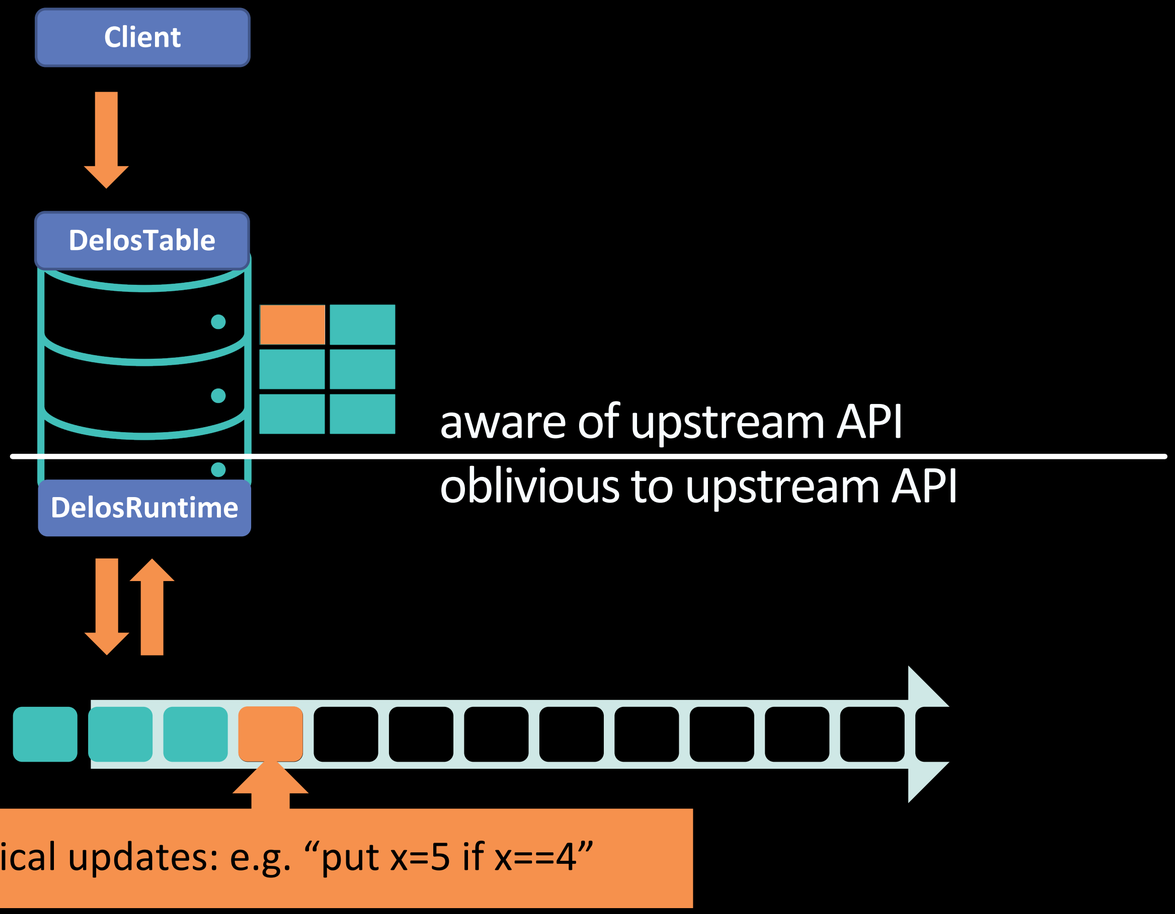


the Delos storage system: above the log



simple protocols above the log

the Delos storage system: above the log

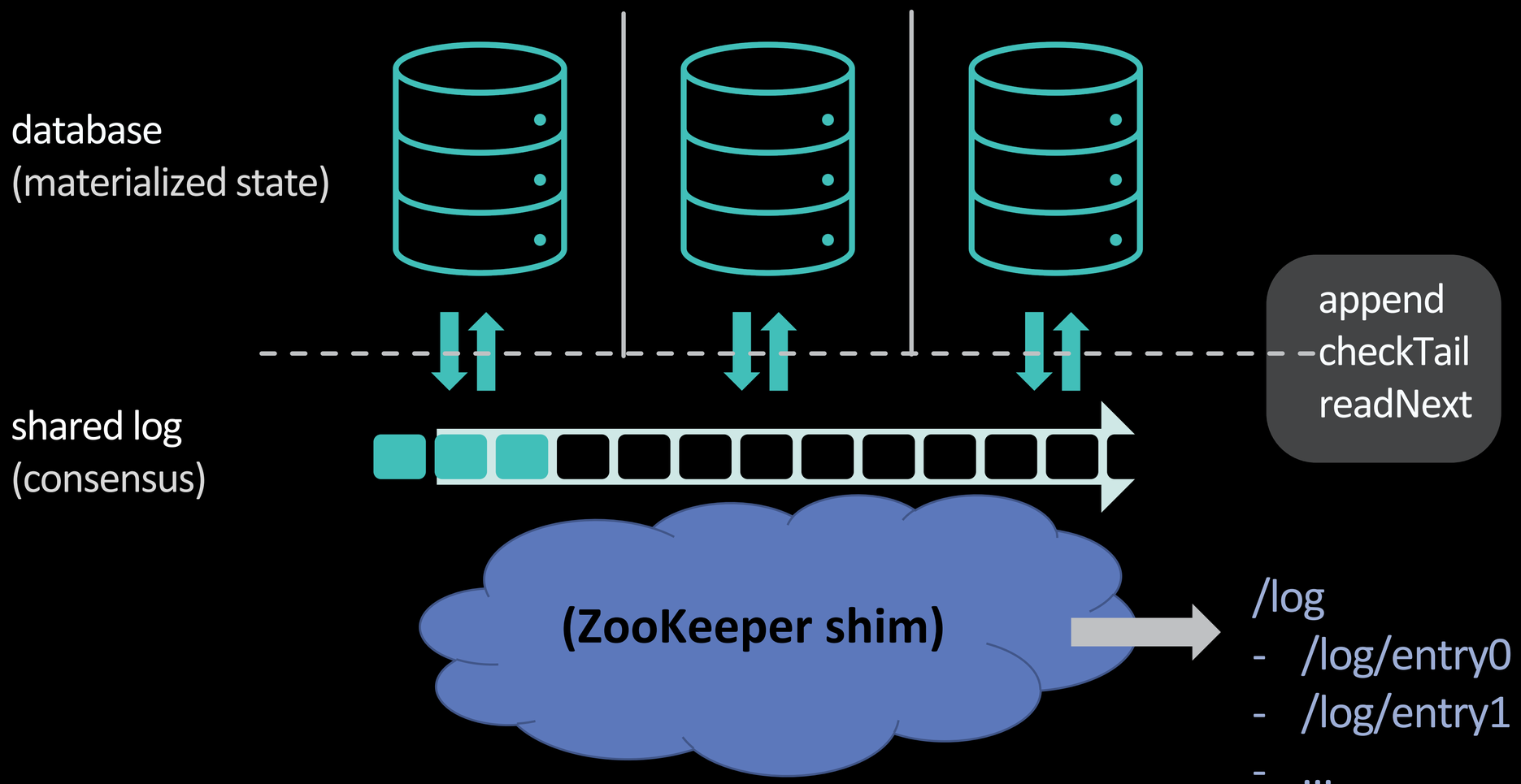


simple protocols above the log

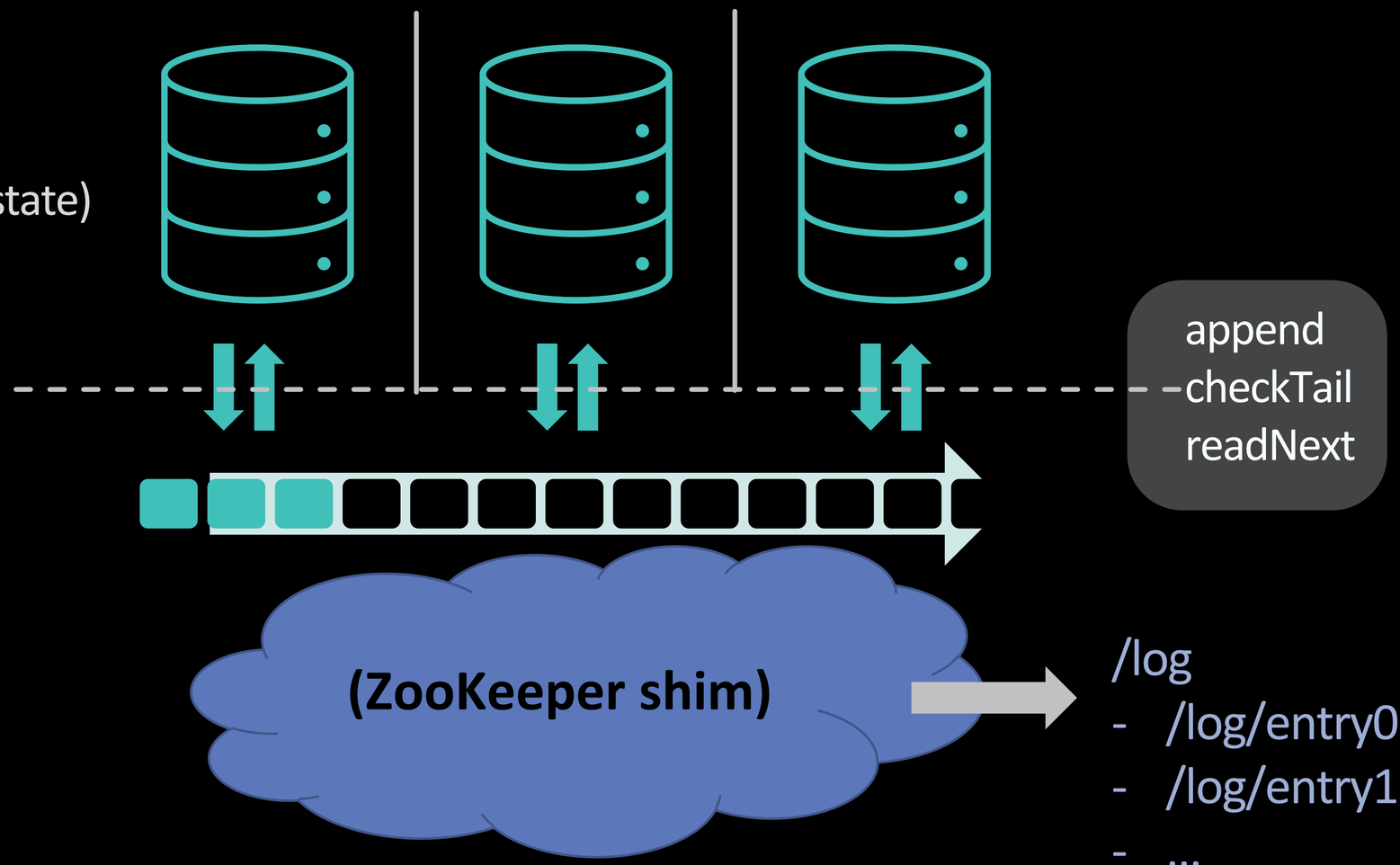
easy to support new APIs

the Delos storage system: below the log

the Delos storage system: below the log



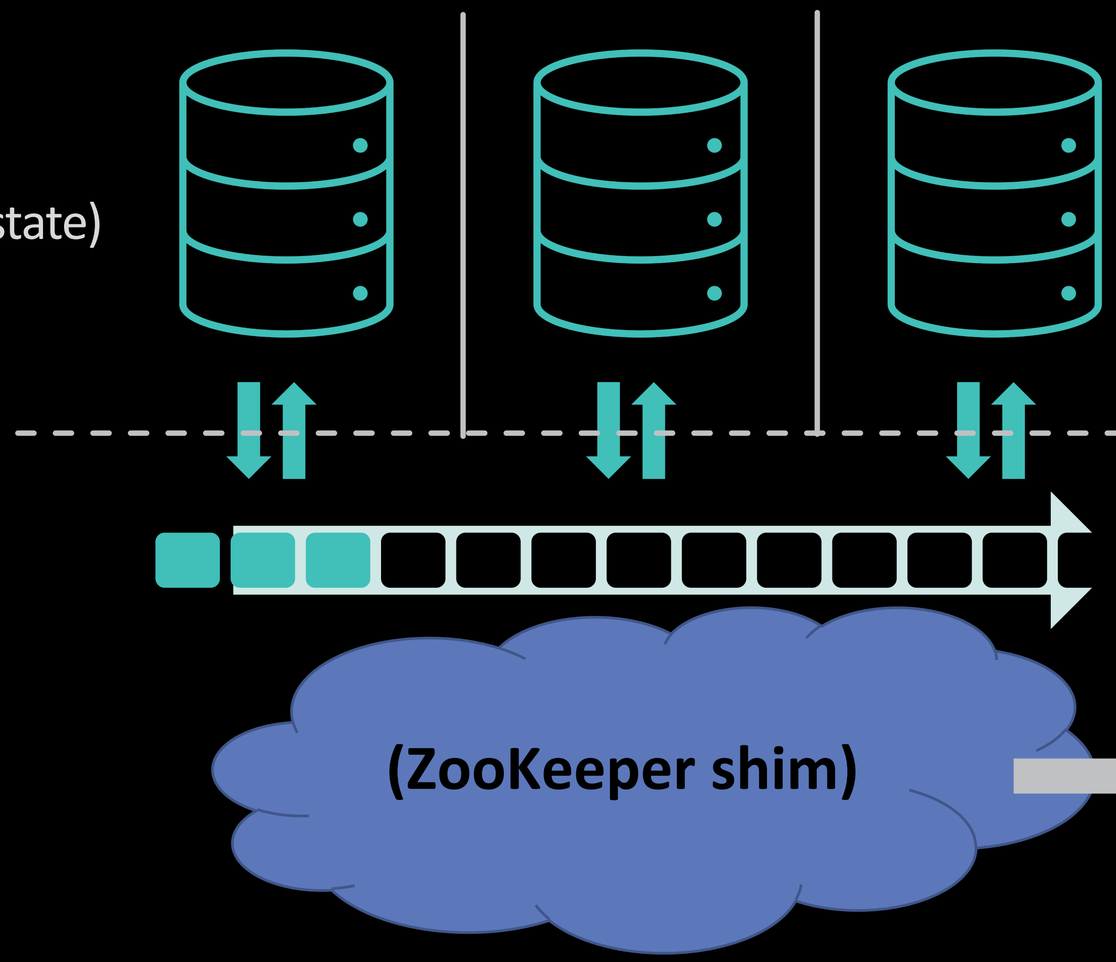
the Delos storage system: below the log



pros:

- fast to build/deploy
- highly reliable

the Delos storage system: below the log



pros:
- fast to build/deploy
- highly reliable

cons:
- very inefficient and slow
- service dependency

how do we develop a new shared log?
(without re-implementing MultiPaxos…)

how do we develop a new shared log?
(without re-implementing MultiPaxos…)

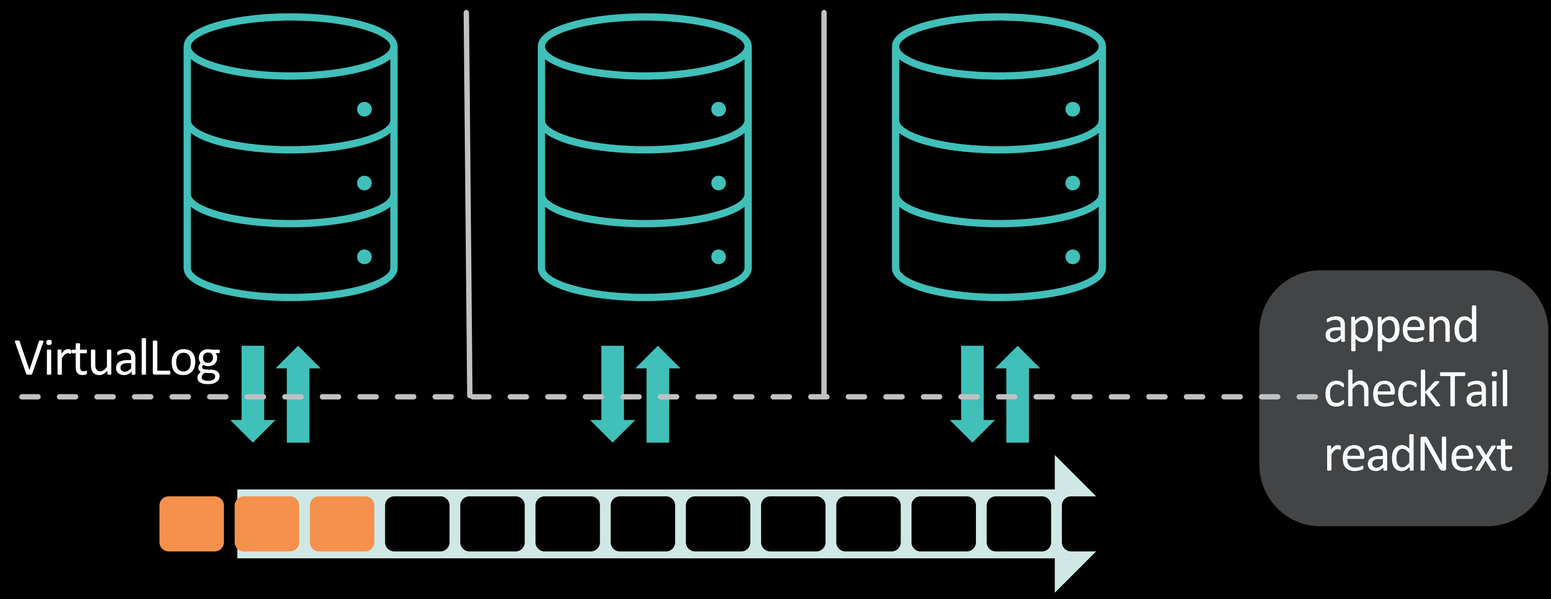
how do we deploy a new shared log?
(without service downtime…)

how do we develop a new shared log?
(without re-implementing MultiPaxos…)

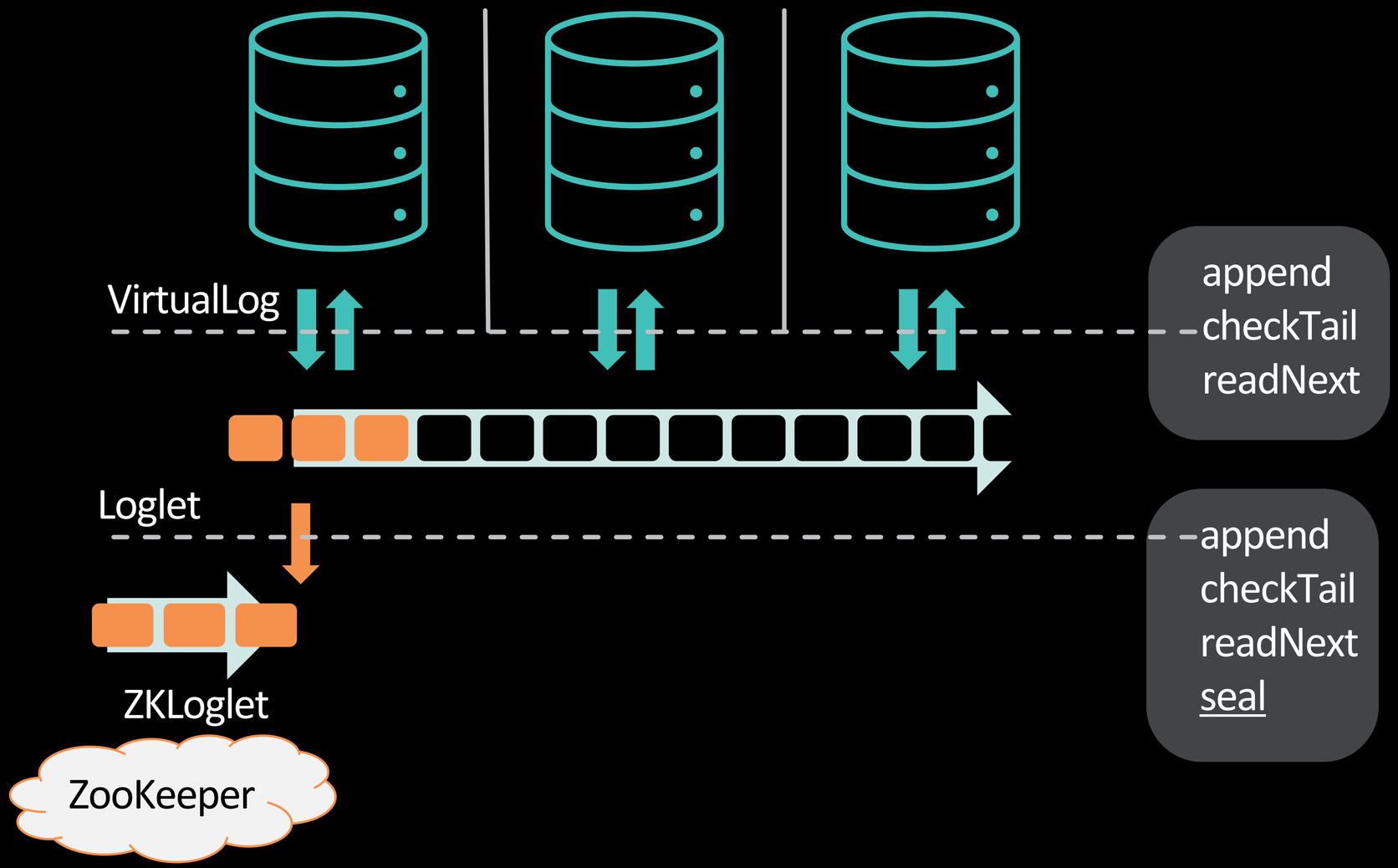
how do we deploy a new shared log?
(without service downtime…)

Virtual Consensus!

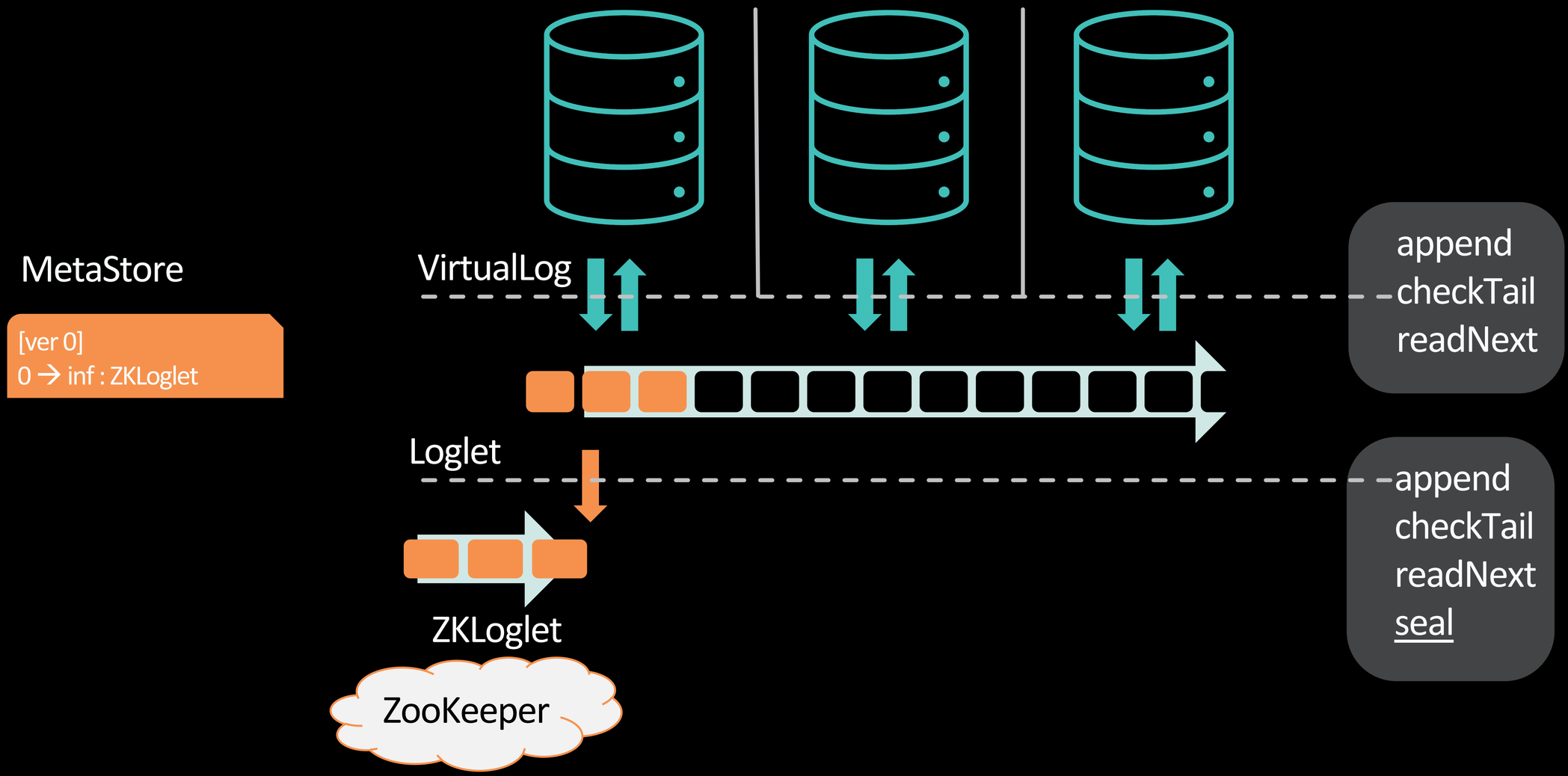
virtualizing consensus via the VirtualLog



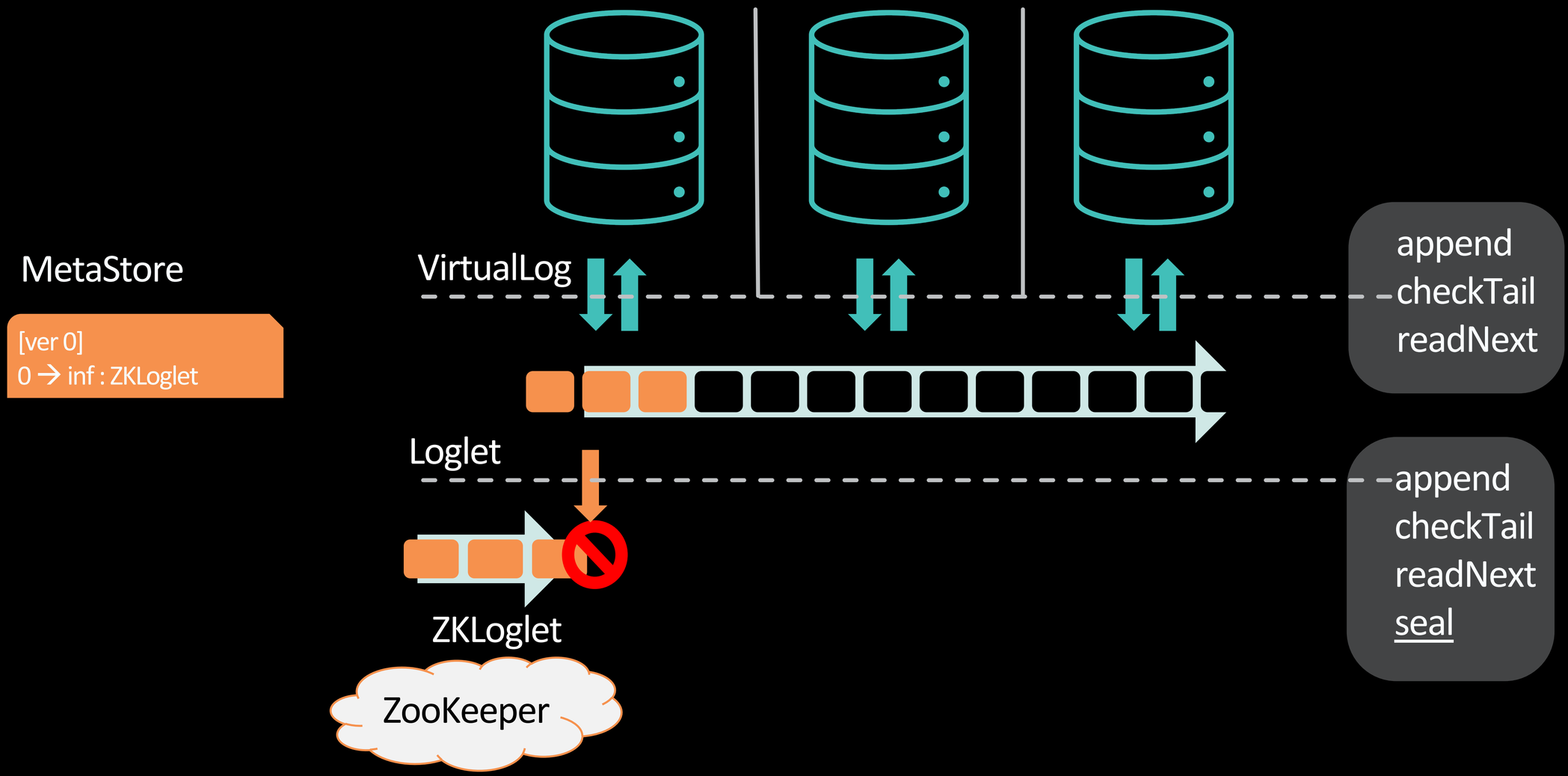
virtualizing consensus via the VirtualLog



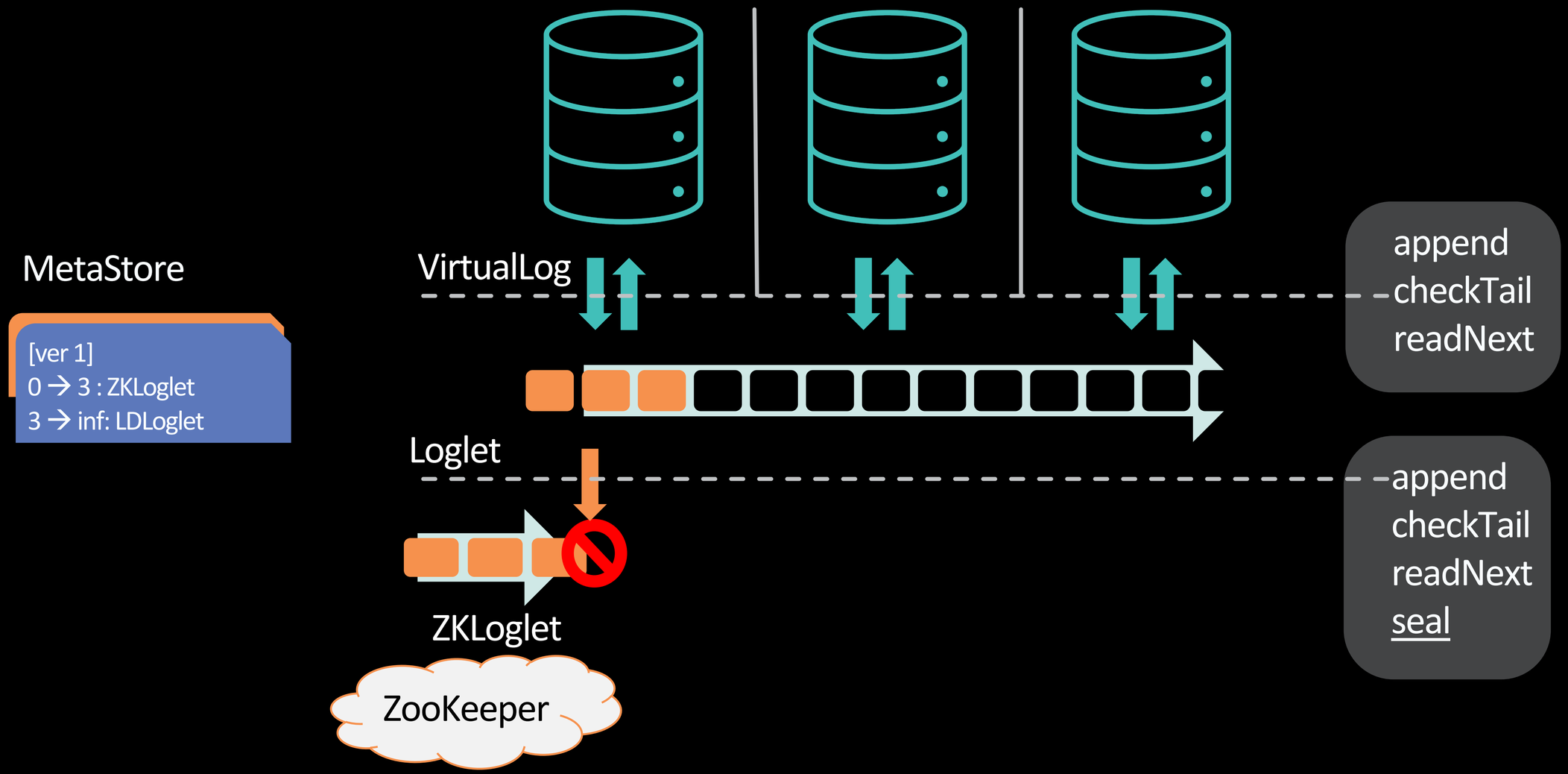
virtualizing consensus via the VirtualLog



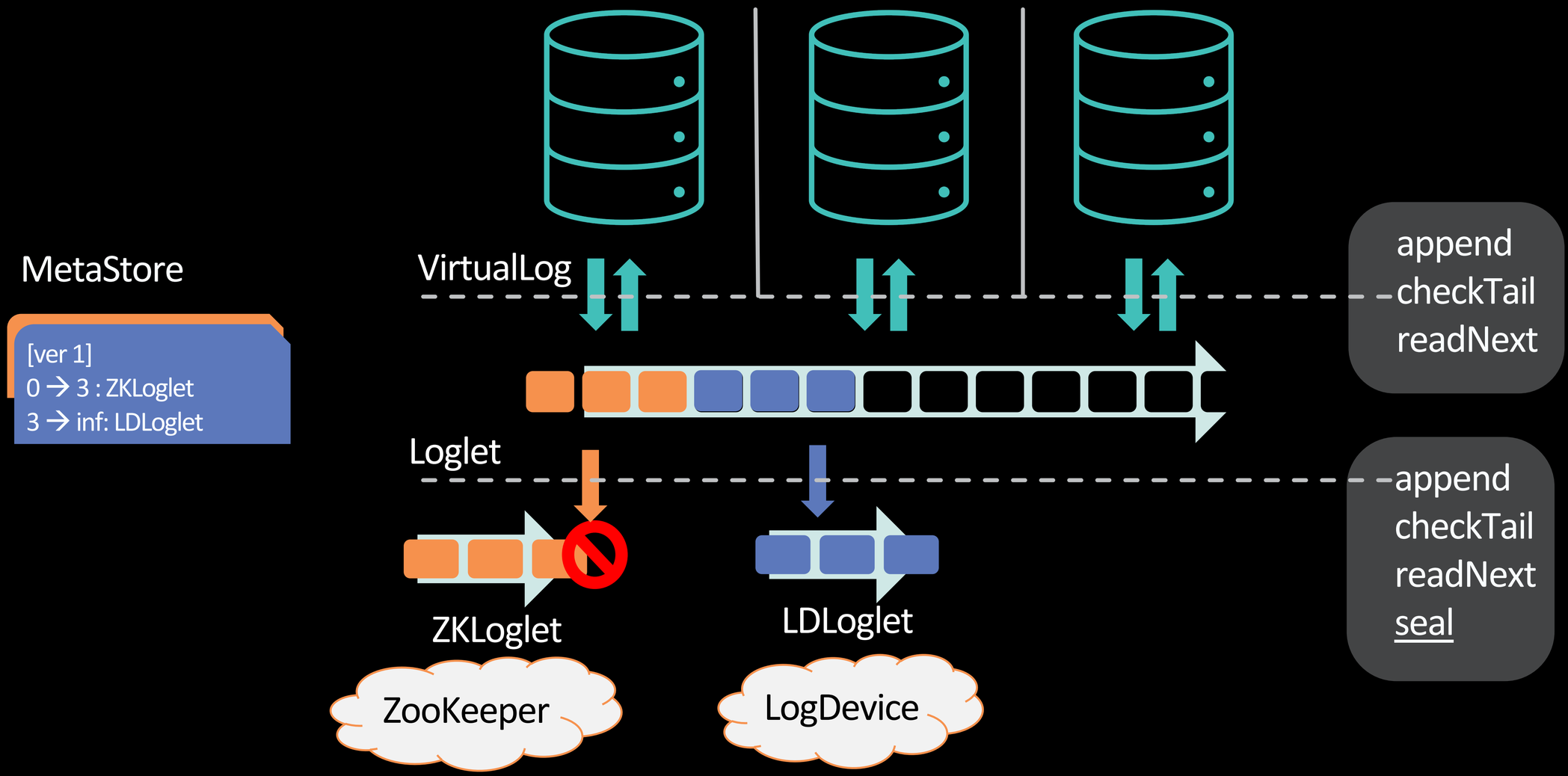
virtualizing consensus via the VirtualLog



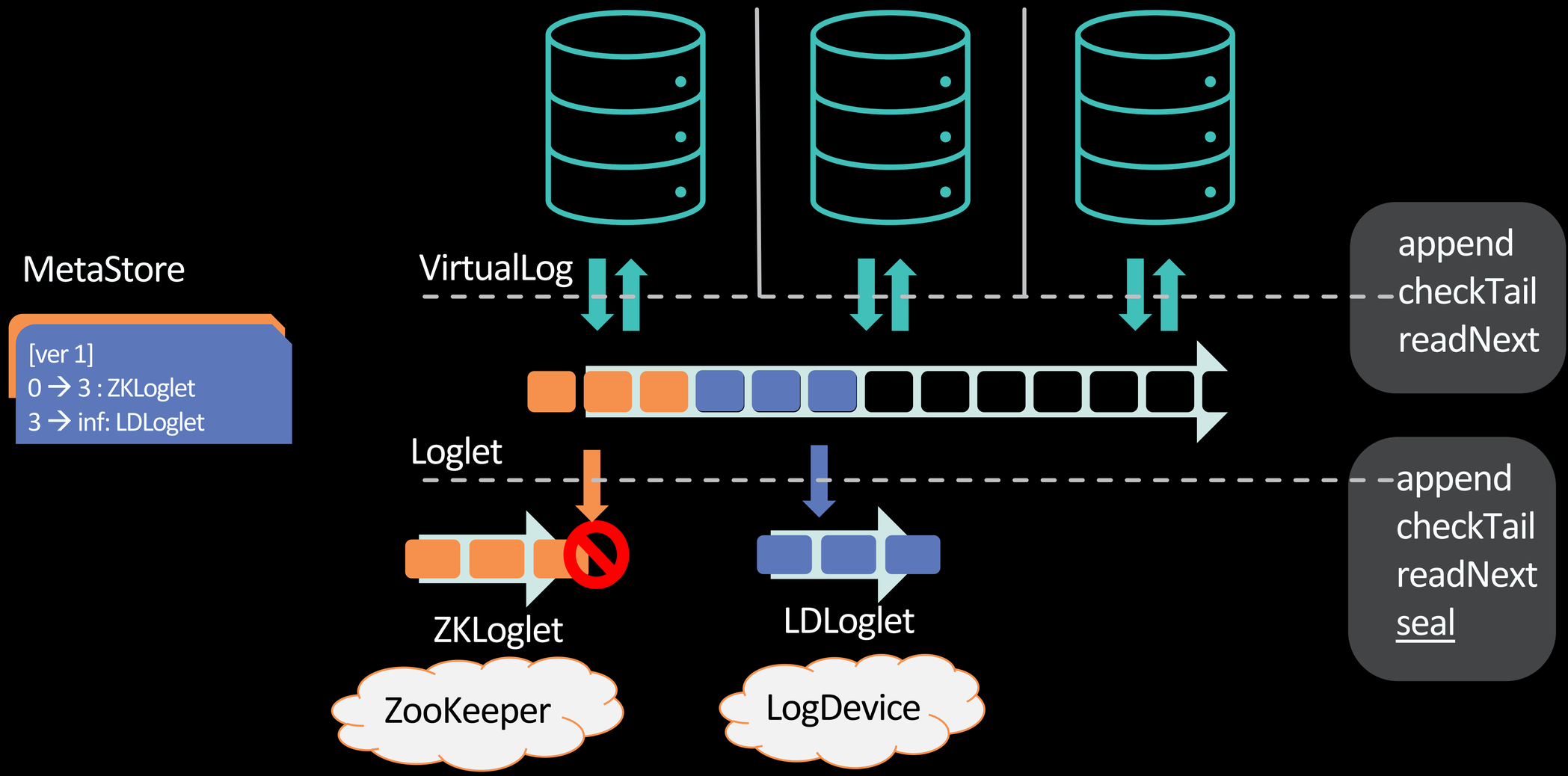
virtualizing consensus via the VirtualLog



virtualizing consensus via the VirtualLog



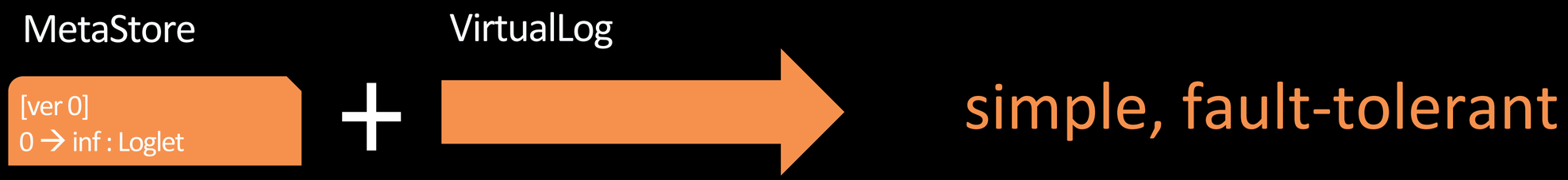
virtualizing consensus via the VirtualLog



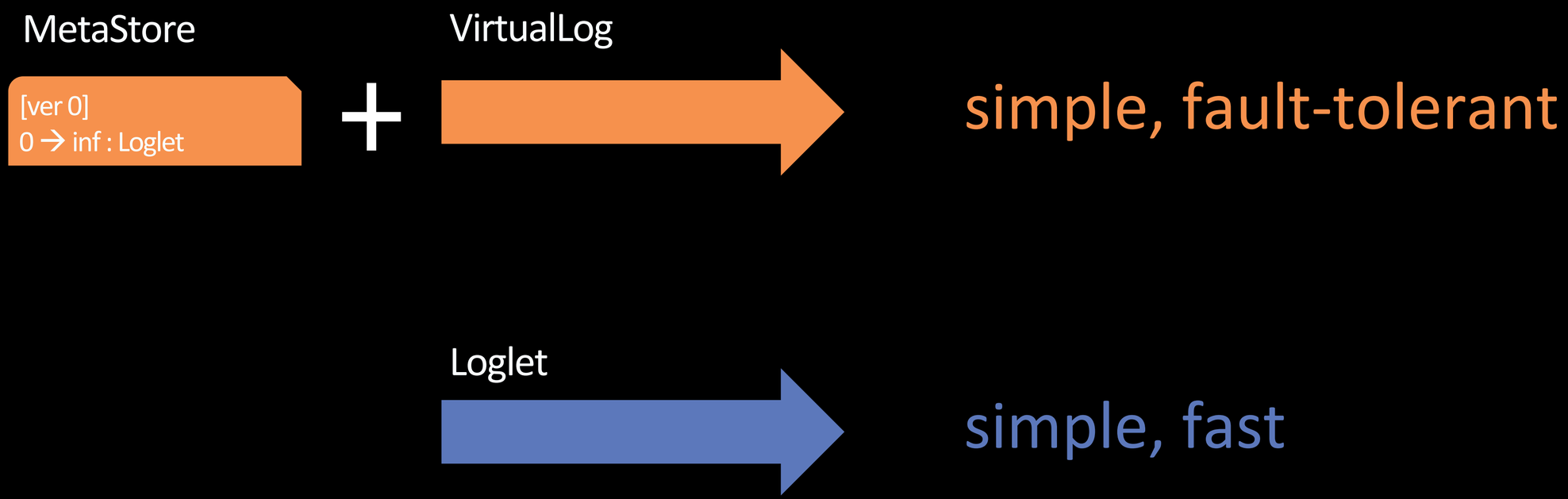
we can deploy a new Loglet without downtime!

difficult to build a log that is simple, fast, fault-tolerant

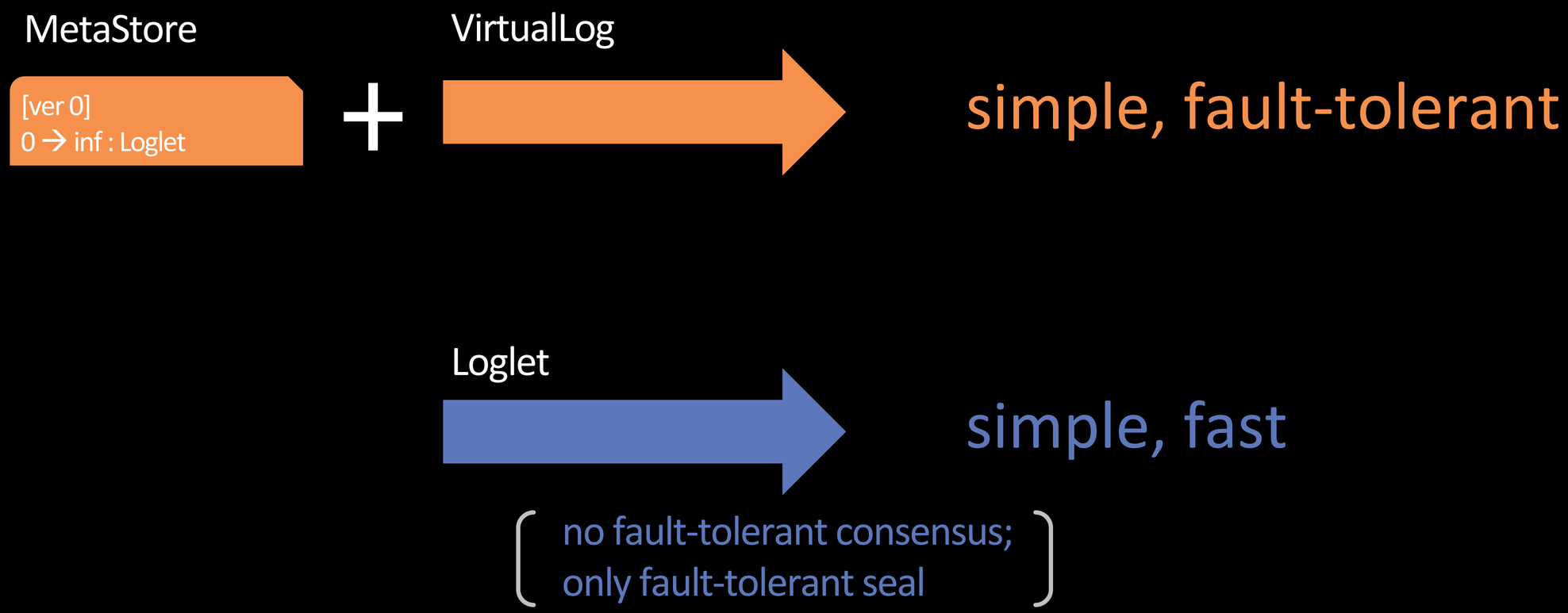
difficult to build a log that is simple, fast, fault-tolerant



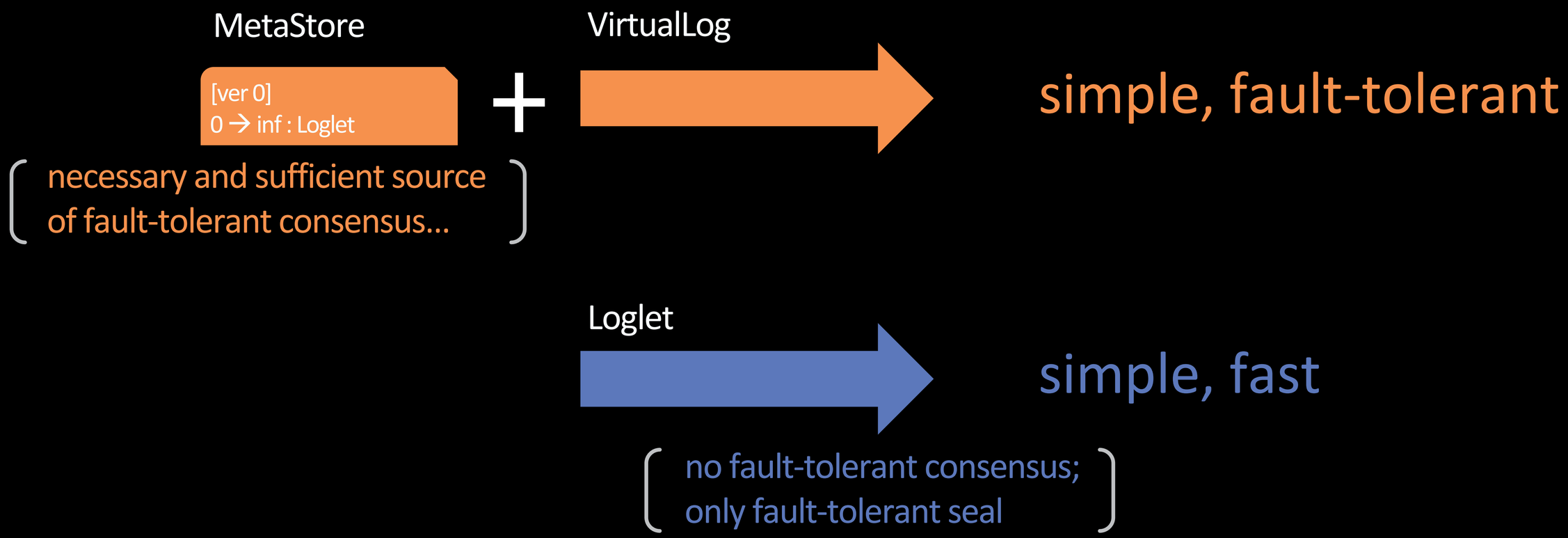
difficult to build a log that is simple, fast, fault-tolerant



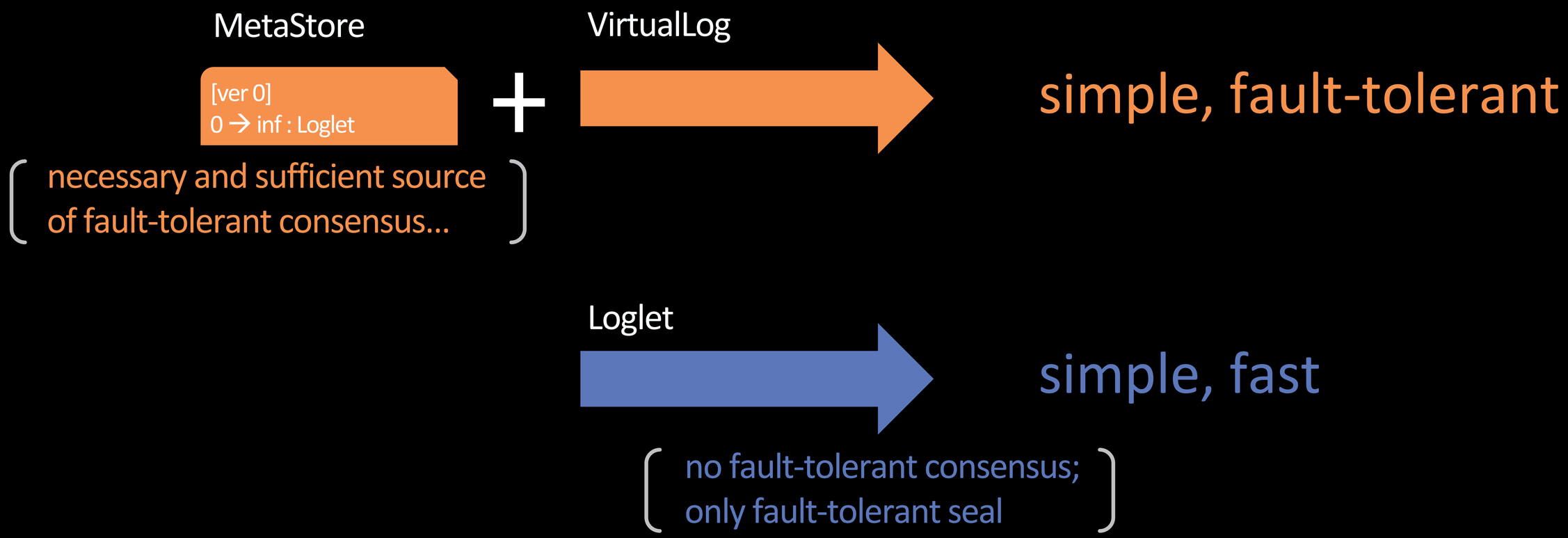
difficult to build a log that is simple, fast, fault-tolerant



difficult to build a log that is simple, fast, fault-tolerant

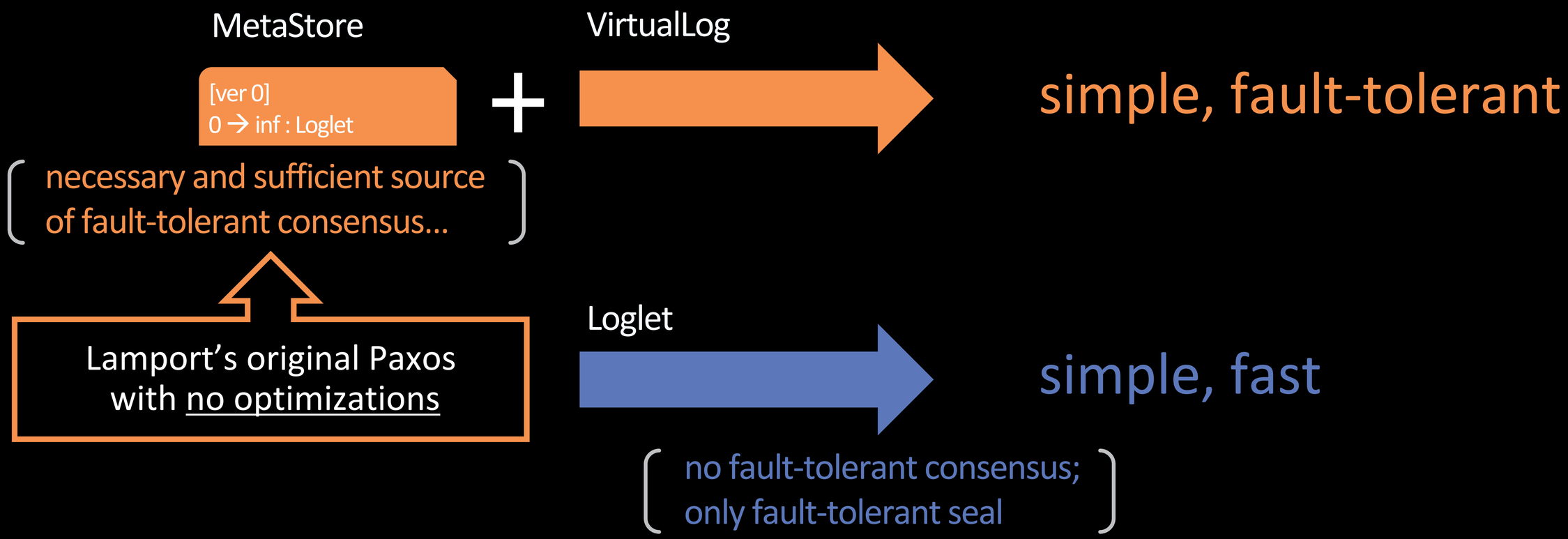


difficult to build a log that is simple, fast, fault-tolerant



the VirtualLog handles all reconfiguration (including leader election);
the Loglet provides failure-free ordering

difficult to build a log that is simple, fast, fault-tolerant



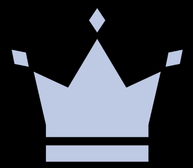
the VirtualLog handles all reconfiguration (including leader election);
the Loglet provides failure-free ordering

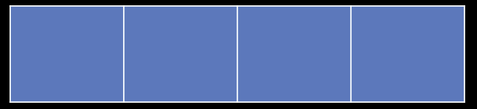
the NativeLoglet

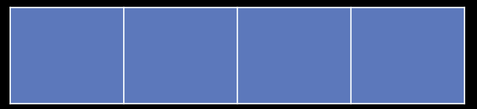
Delos Runtime (client)

sequencer

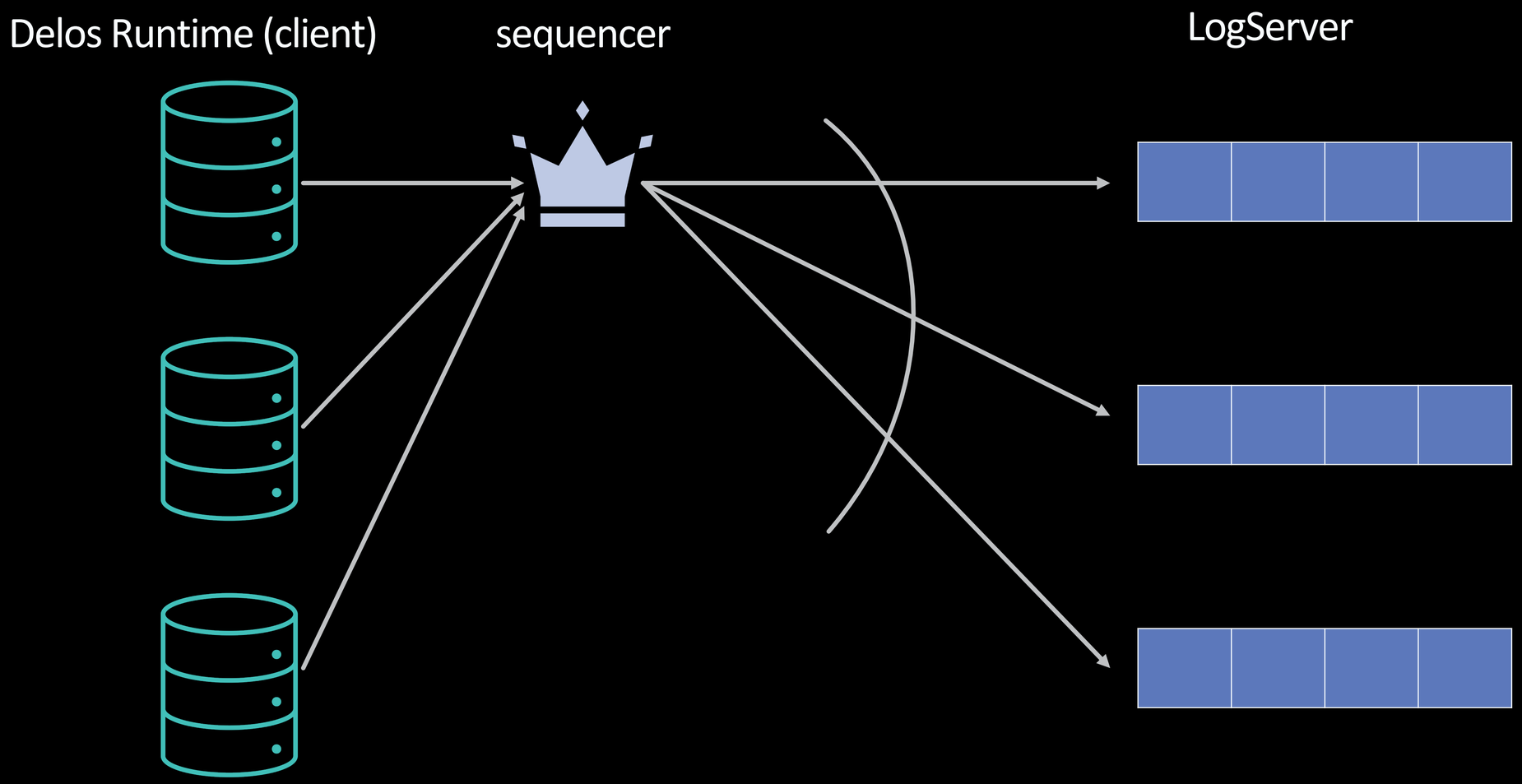
LogServer





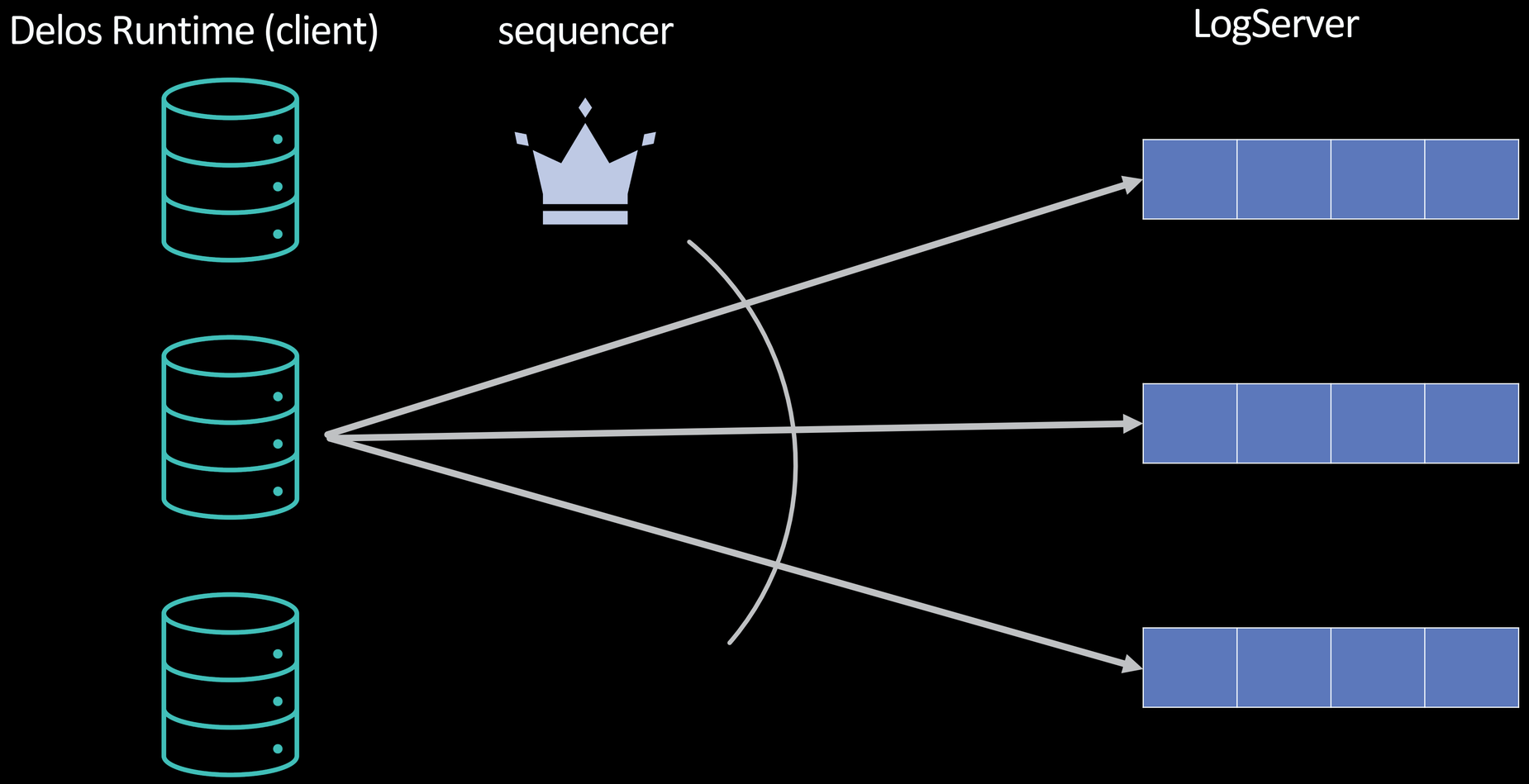


the NativeLoglet



1 quorum appends

the NativeLoglet



1 quorum appends

2 quorum checkTail

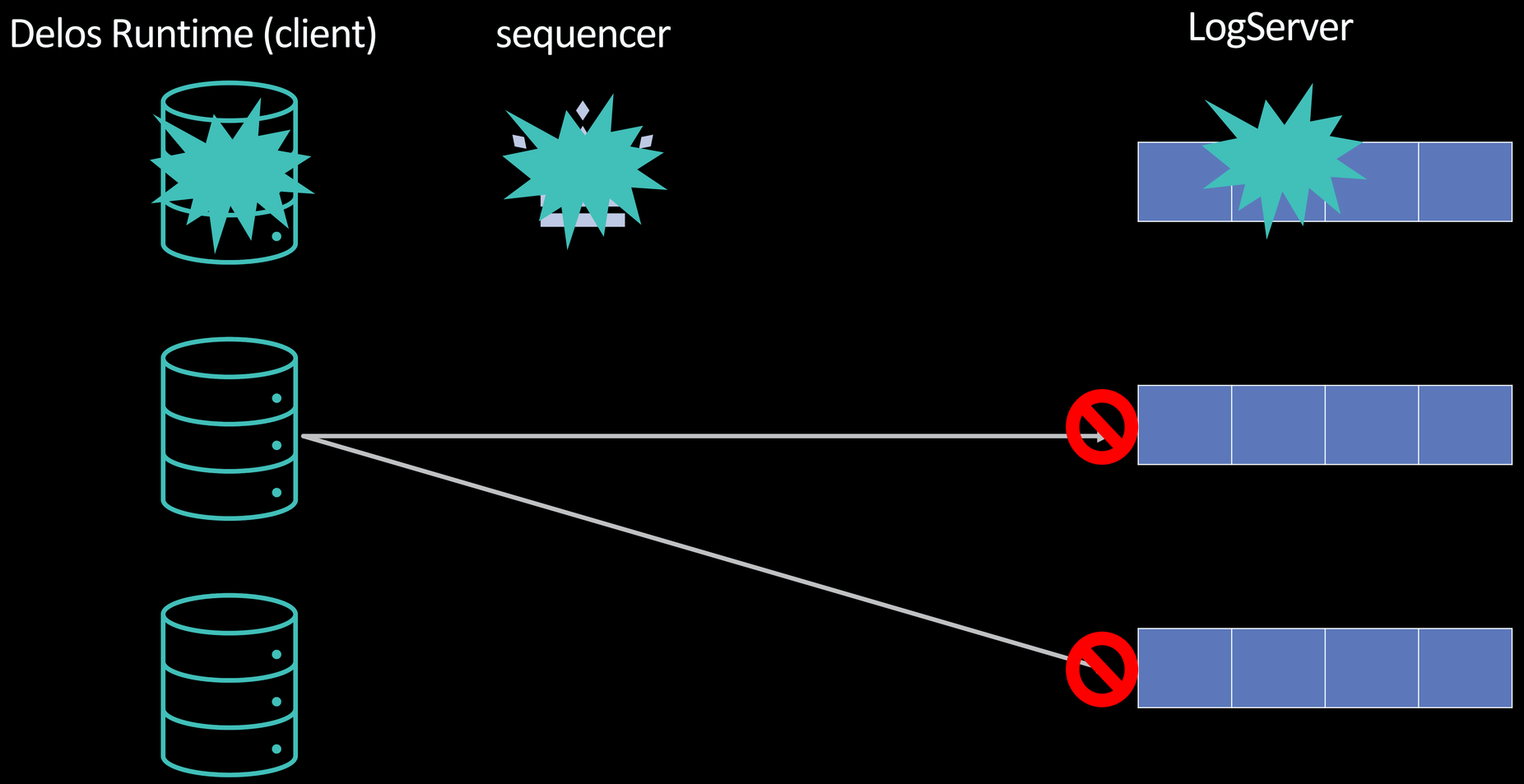
the NativeLoglet

1 quorum appends

2 quorum checkTail

3 fast local reads

the NativeLoglet



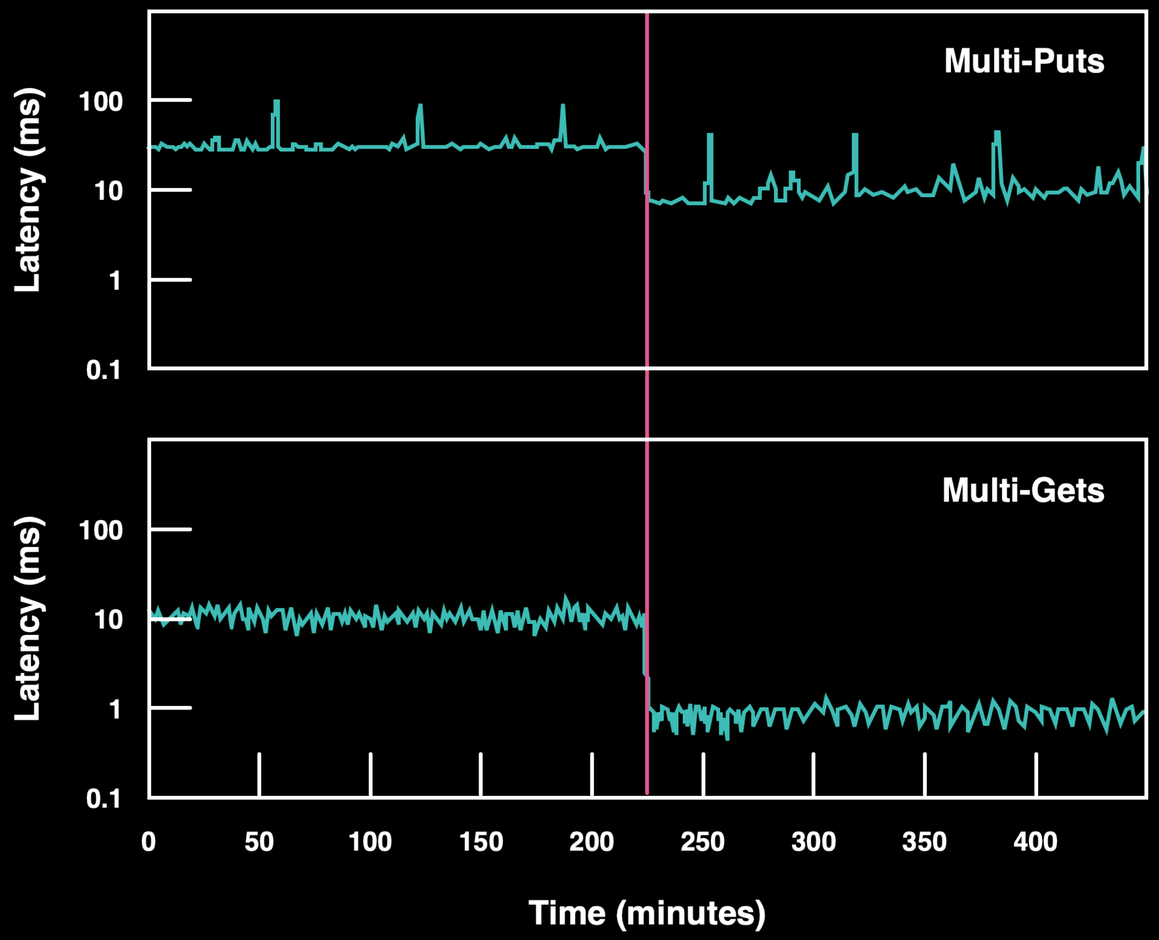
1 quorum appends

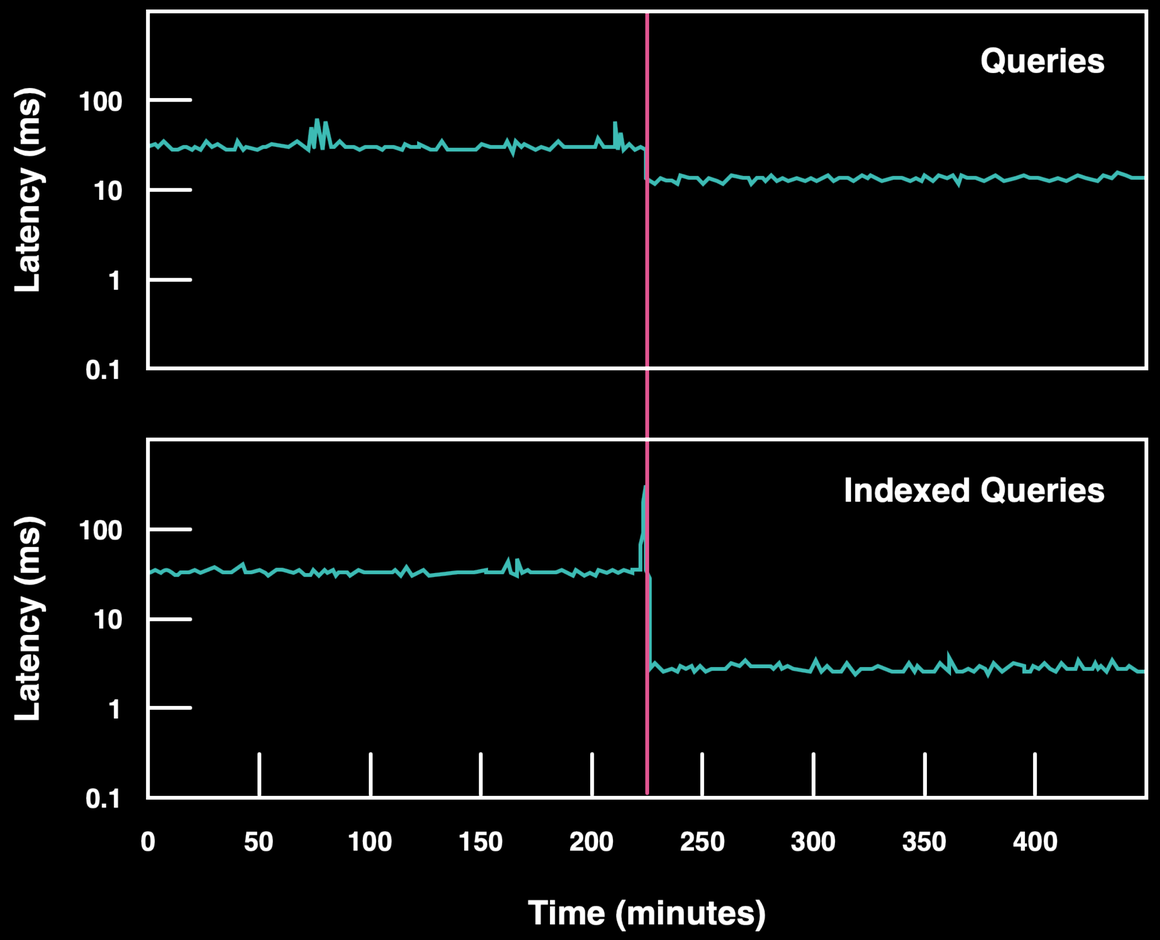
2 quorum checkTail

3 fast local reads

4 fault-tolerant seal

switching logs mid-flight





deploying Loglets: converged vs. disaggregated

deploying Loglets: converged vs. disaggregated

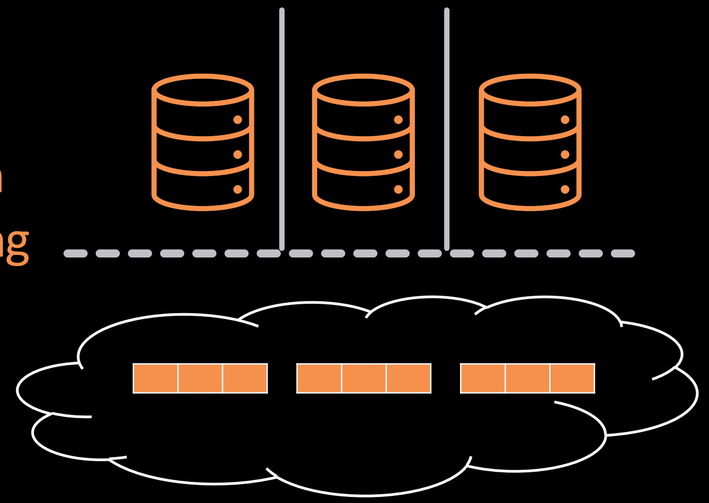
log+DB on each server:

- fast local log reads
- fate-sharing

deploying Loglets: converged vs. disaggregated

log+DB on each server:
- fast local log reads
- fate-sharing

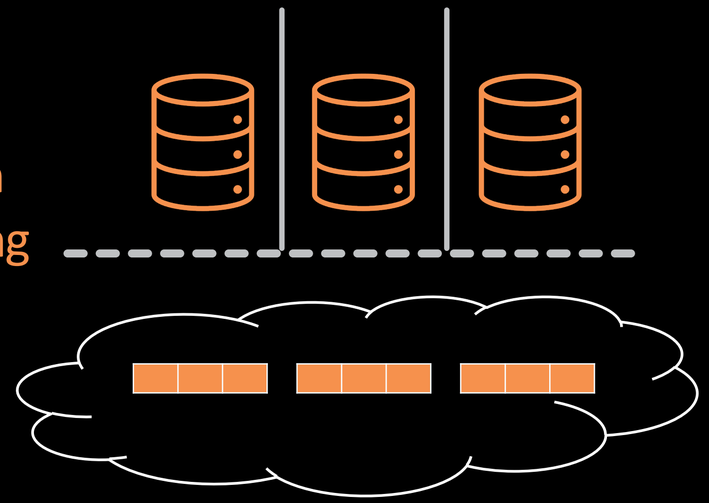
separate log and DB:
- less I/O contention
- independent scaling



deploying Loglets: converged vs. disaggregated

log+DB on each server:
- fast local log reads
- fate-sharing

separate log and DB:
- less I/O contention
- independent scaling

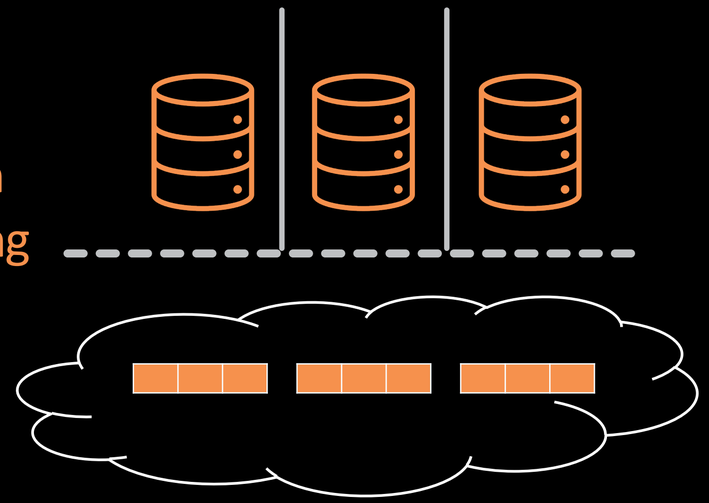


converged is preferred in production:
the DB wants fate-sharing with the log…

deploying Loglets: converged vs. disaggregated

log+DB on each server:
- fast local log reads
- fate-sharing

separate log and DB:
- less I/O contention
- independent scaling

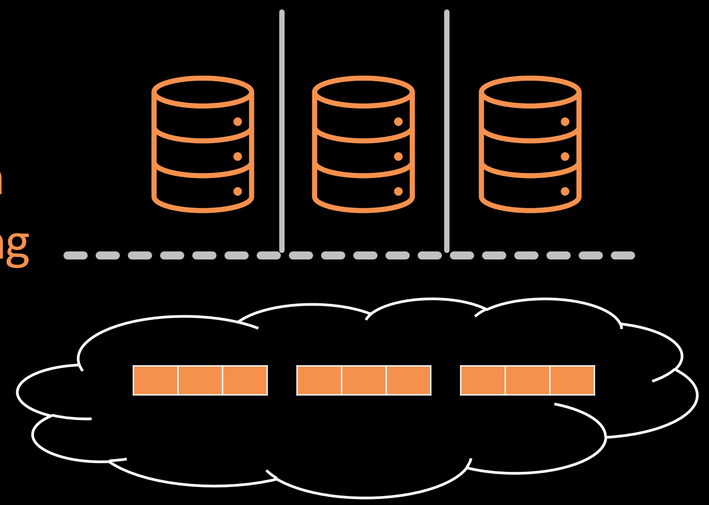


converged is preferred in production:
the DB wants fate-sharing with the log…
(unless its own fate is bad…)

deploying Loglets: converged vs. disaggregated

log+DB on each server:
- fast local log reads
- fate-sharing

separate log and DB:
- less I/O contention
- independent scaling



converged is preferred in production:
the DB wants fate-sharing with the log…

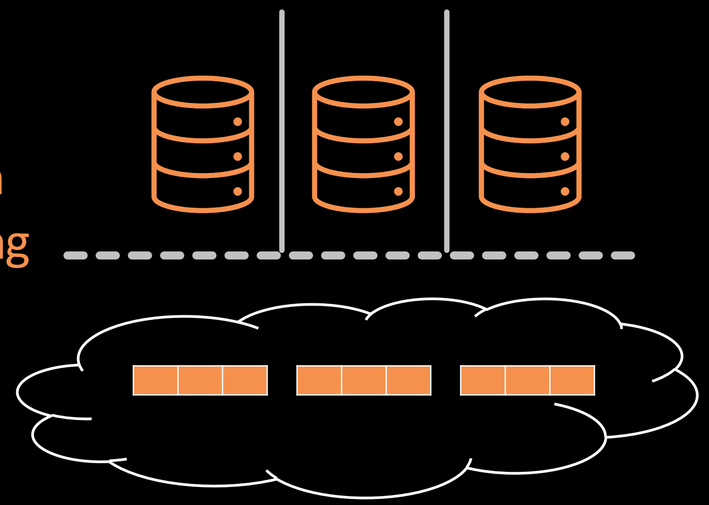
(unless its own fate is bad…)

… we can decouple fate on demand by reconfiguring to a disaggregated log

deploying Loglets: converged vs. disaggregated

log+DB on each server:
- fast local log reads
- fate-sharing

separate log and DB:
- less I/O contention
- independent scaling



converged is preferred in production:
the DB wants fate-sharing with the log…

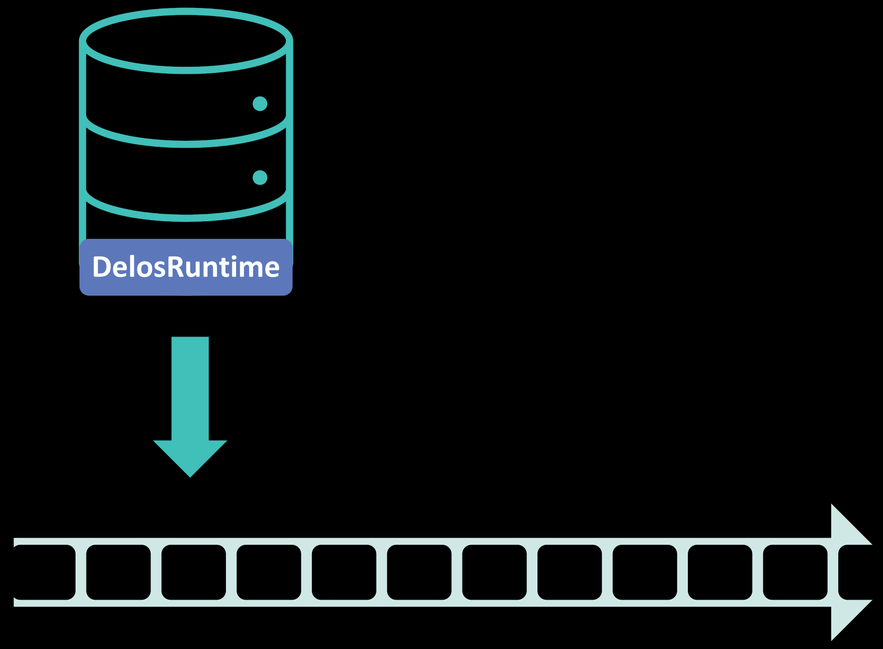
(unless its own fate is bad…)

… we can decouple fate on demand by reconfiguring to a disaggregated log

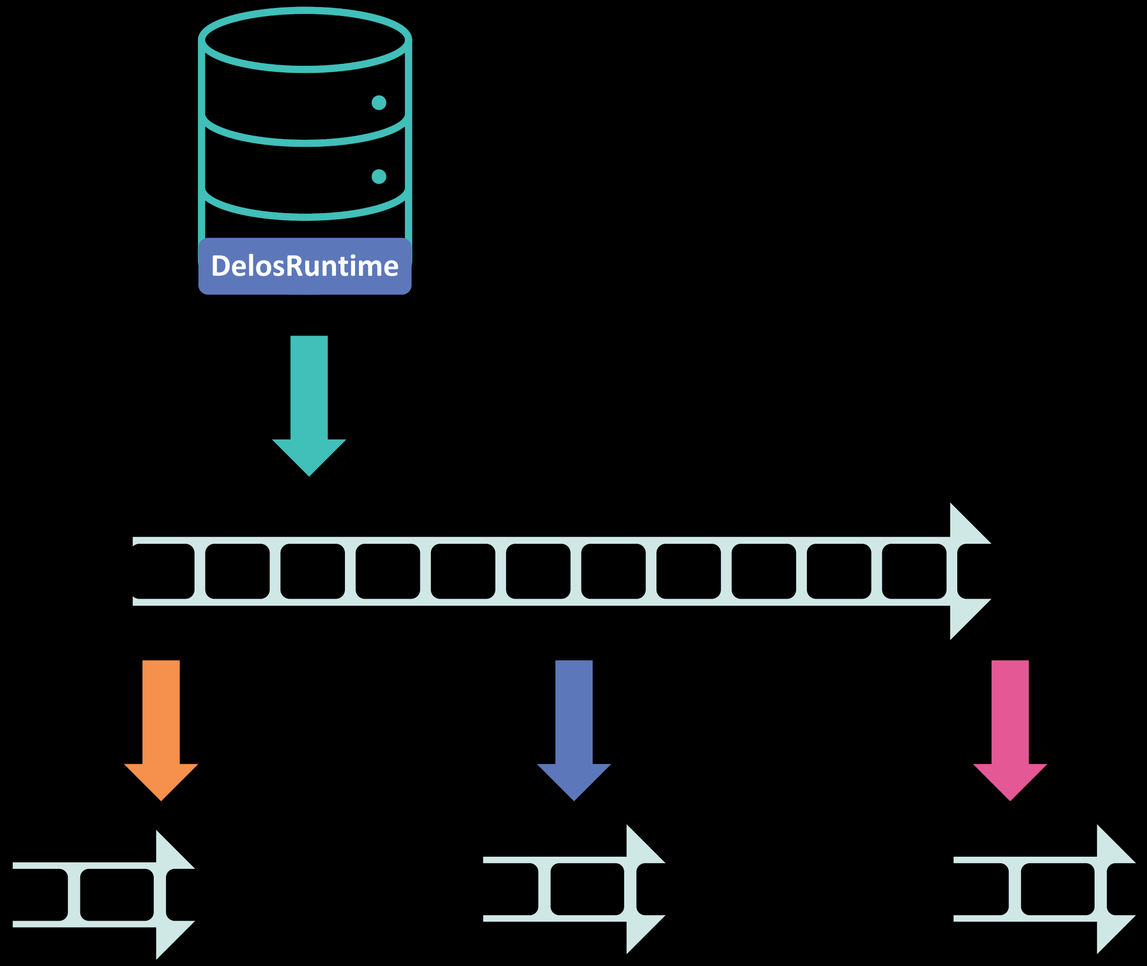
10X

higher throughput via disaggregation

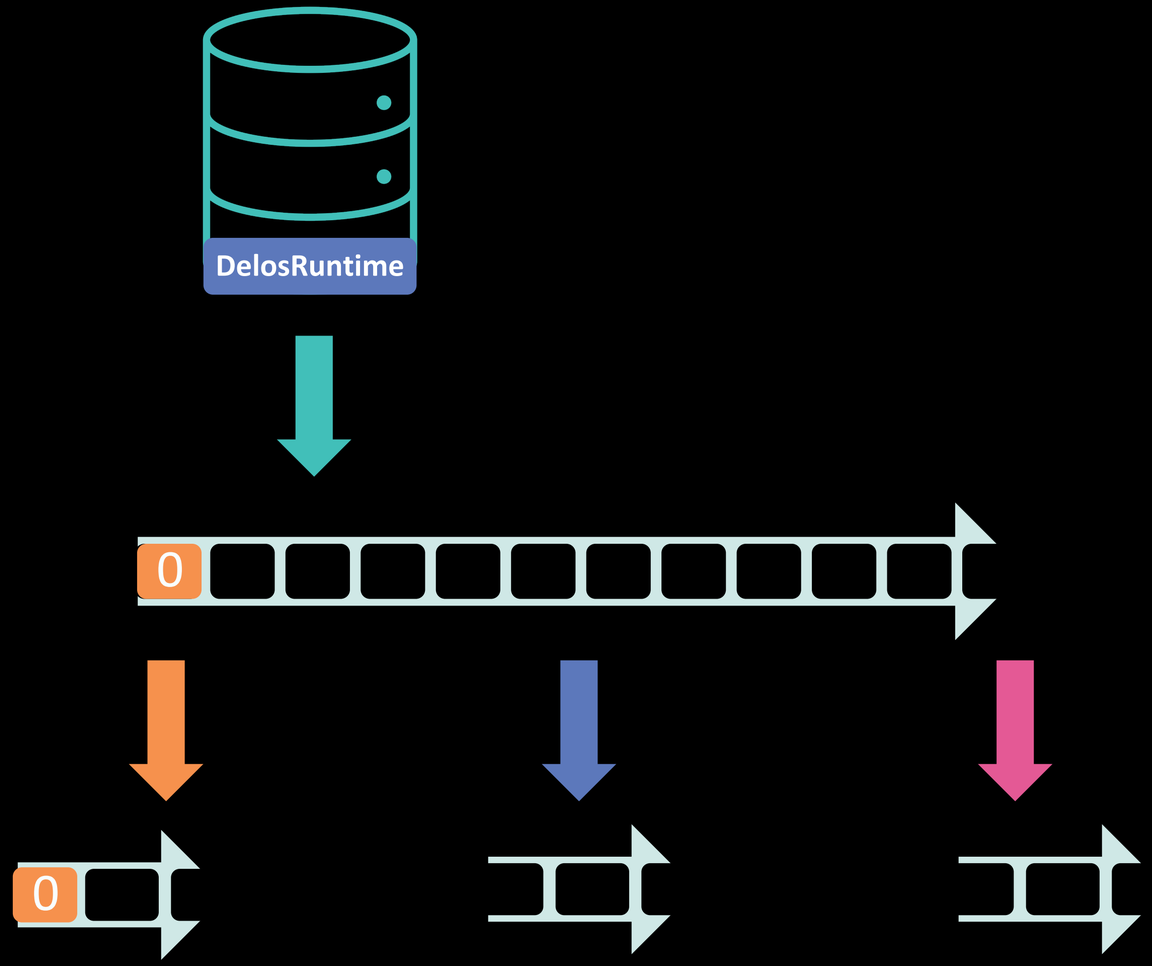
composing Loglets: the StripedLoglet



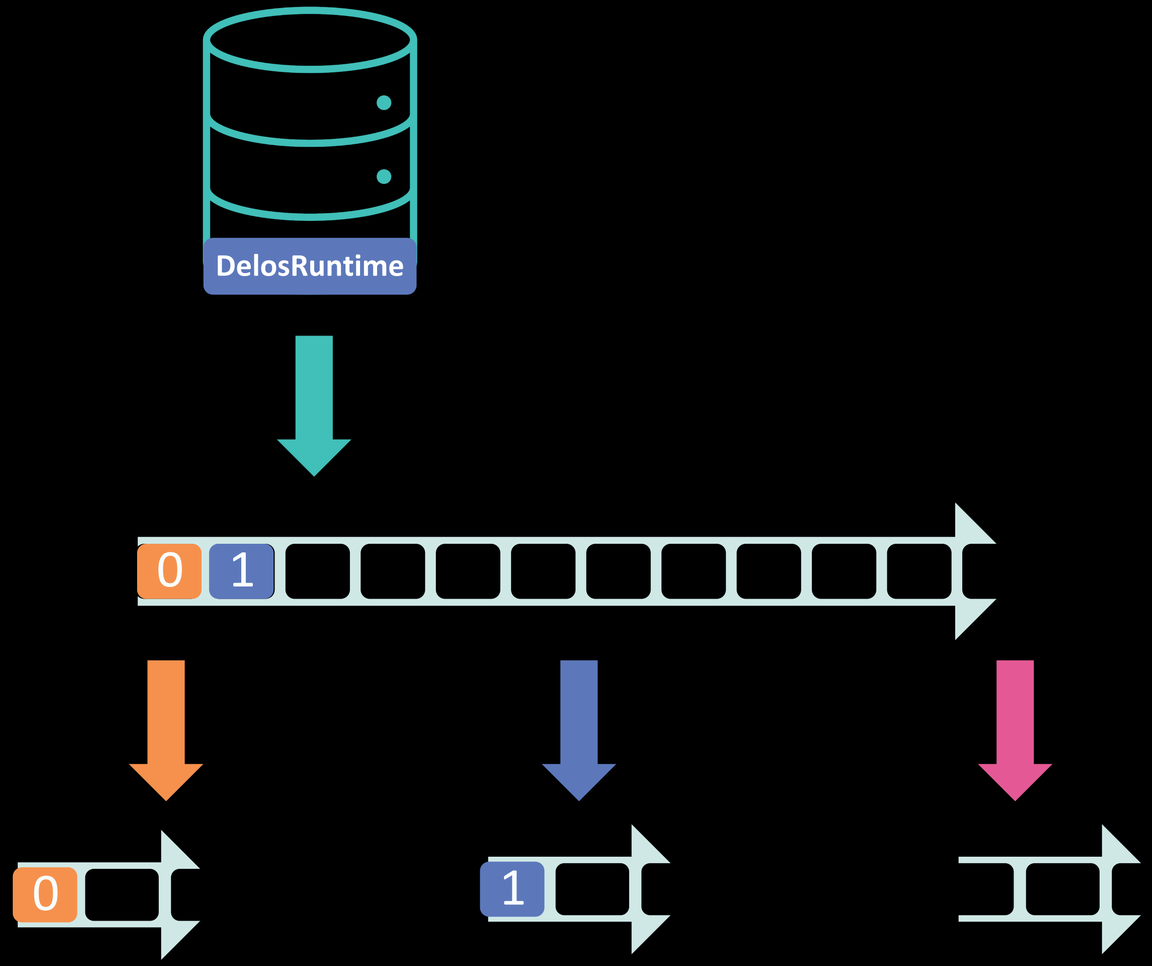
composing Loglets: the StripedLoglet



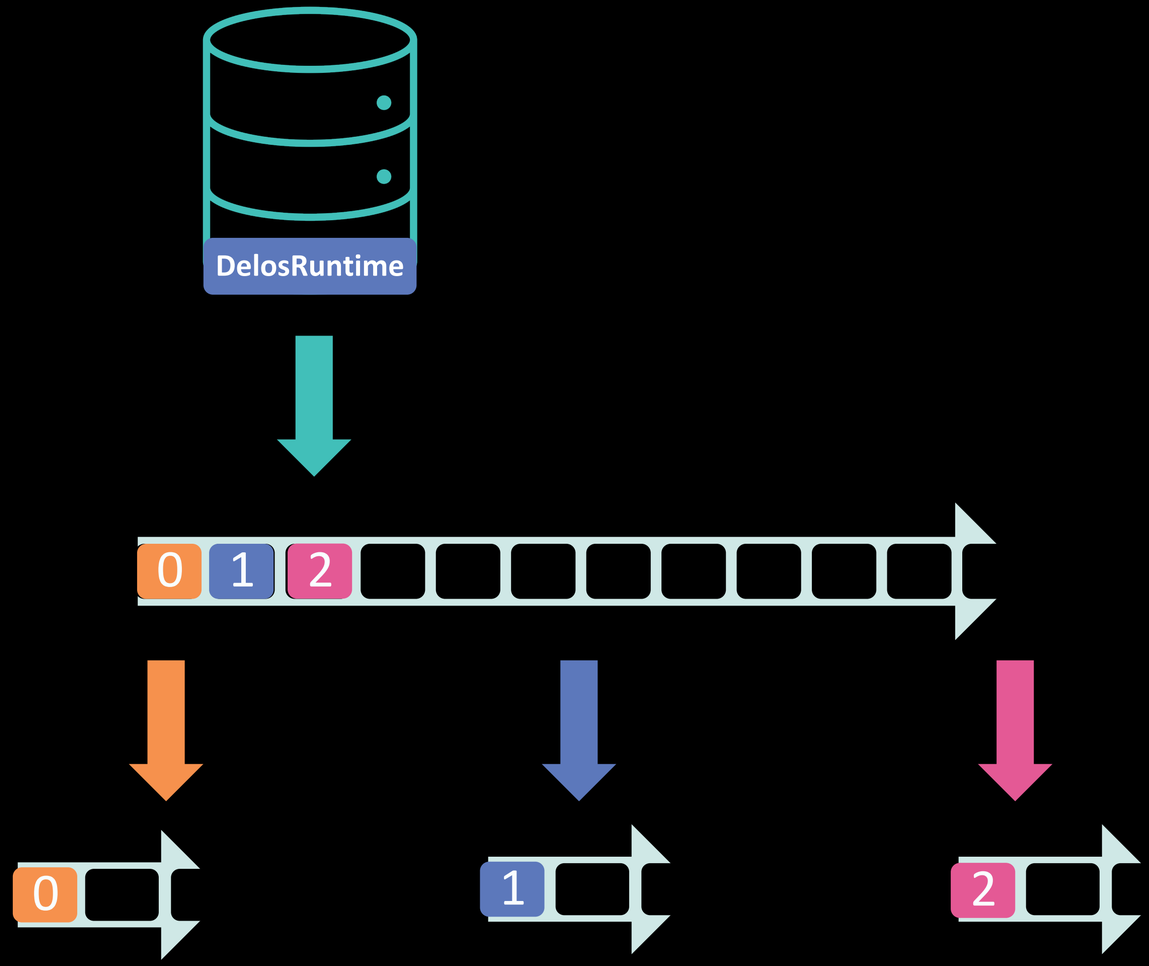
composing Loglets: the StripedLoglet



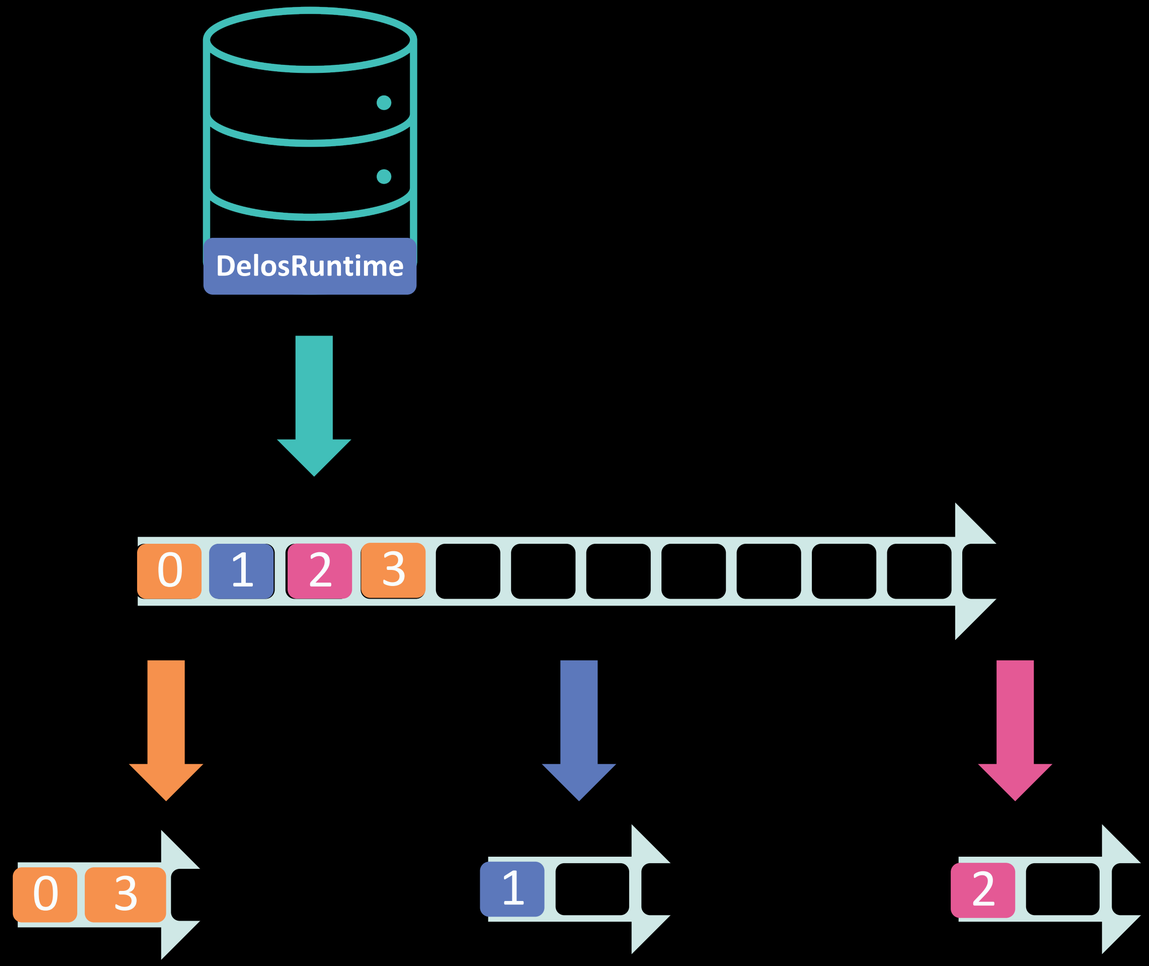
composing Loglets: the StripedLoglet



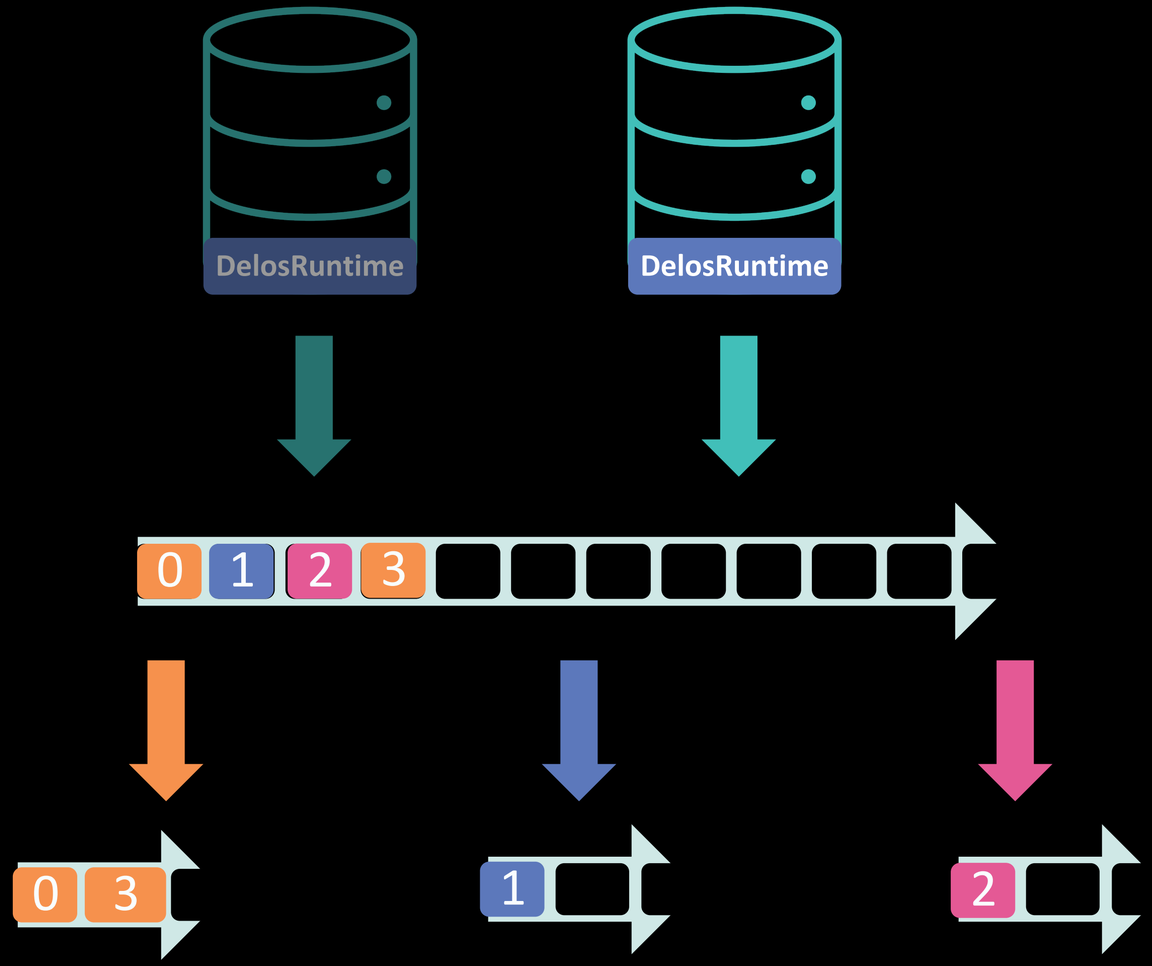
composing Loglets: the StripedLoglet



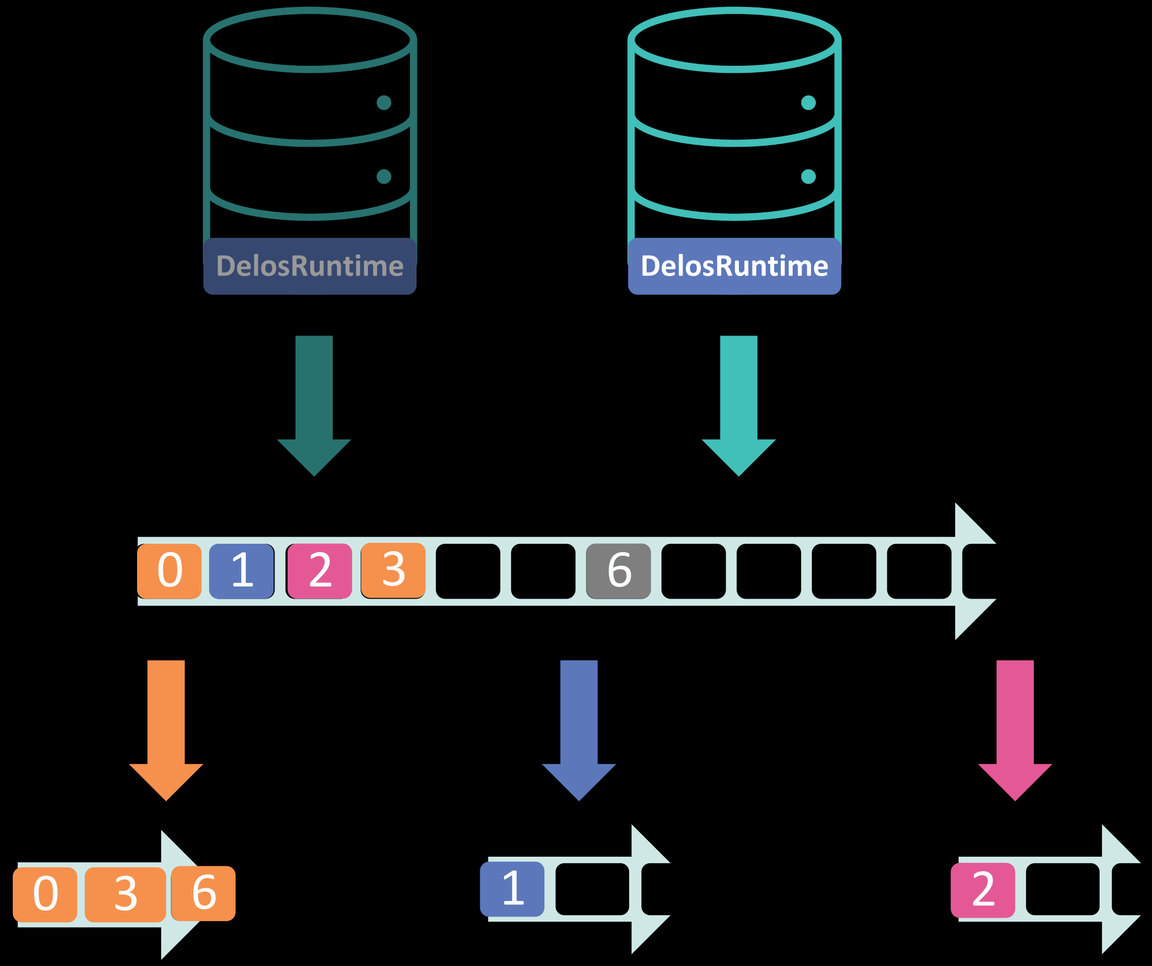
composing Loglets: the StripedLoglet



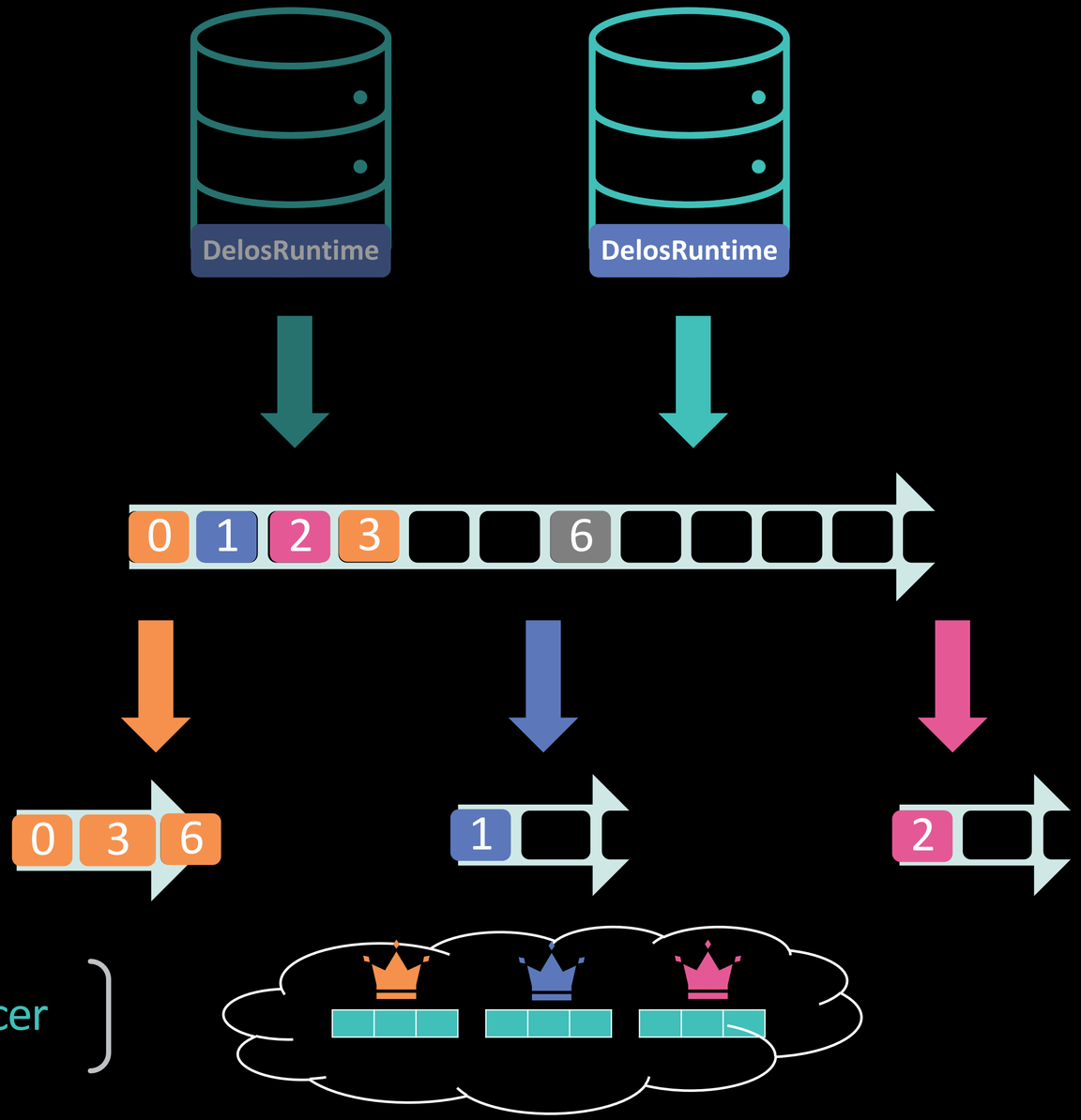
composing Loglets: the StripedLoglet



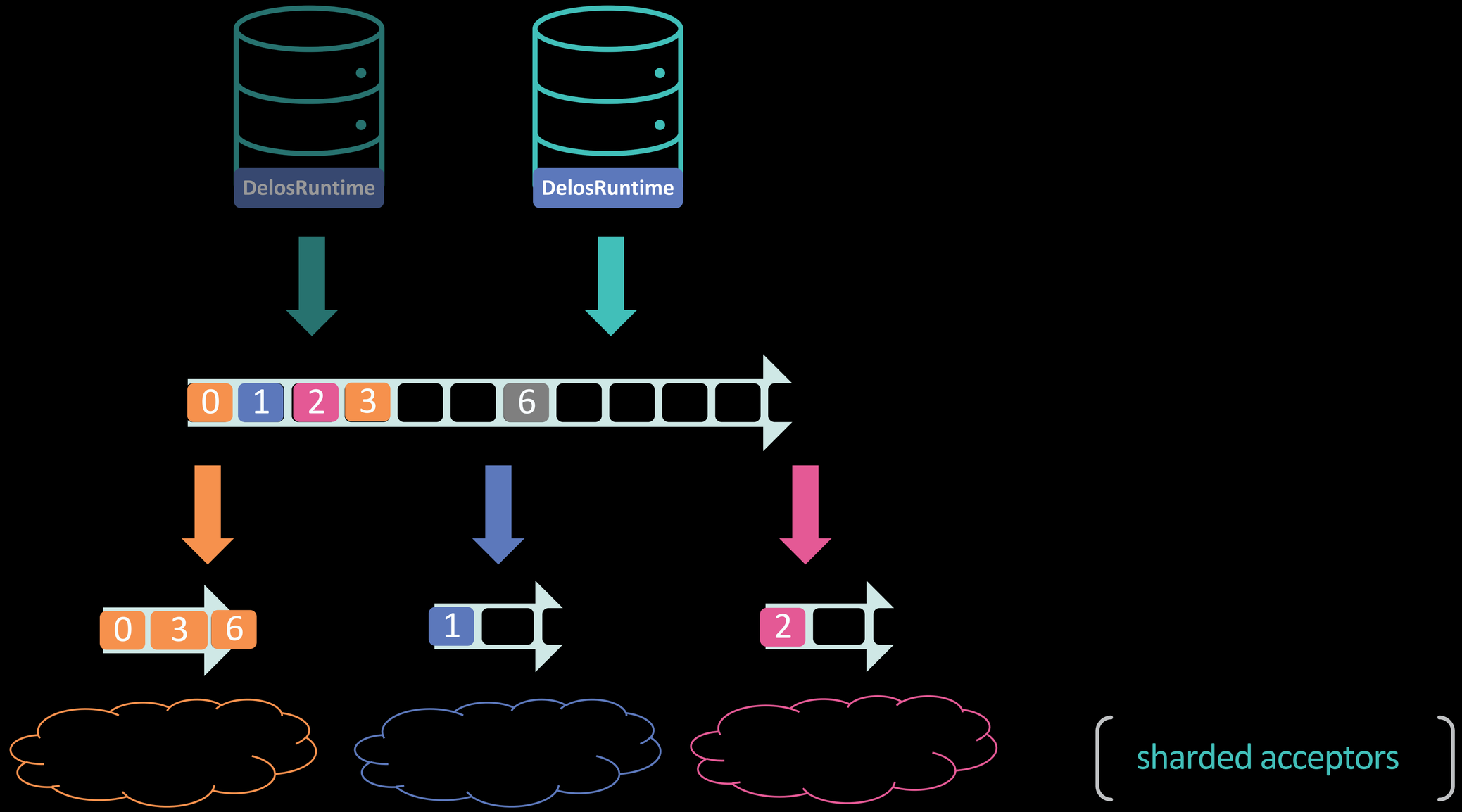
composing Loglets: the StripedLoglet



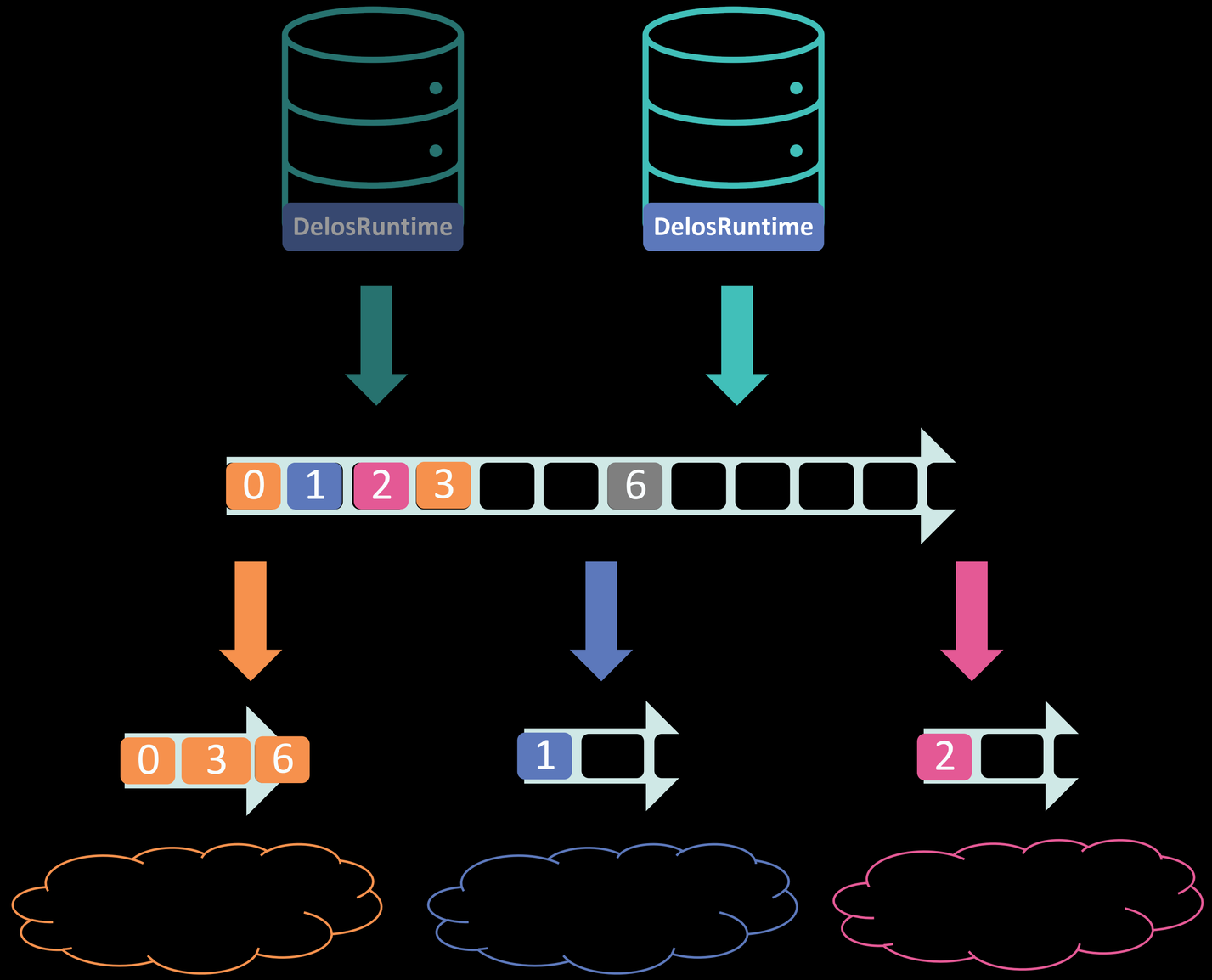
composing Loglets: the StripedLoglet



composing Loglets: the StripedLoglet



composing Loglets: the StripedLoglet

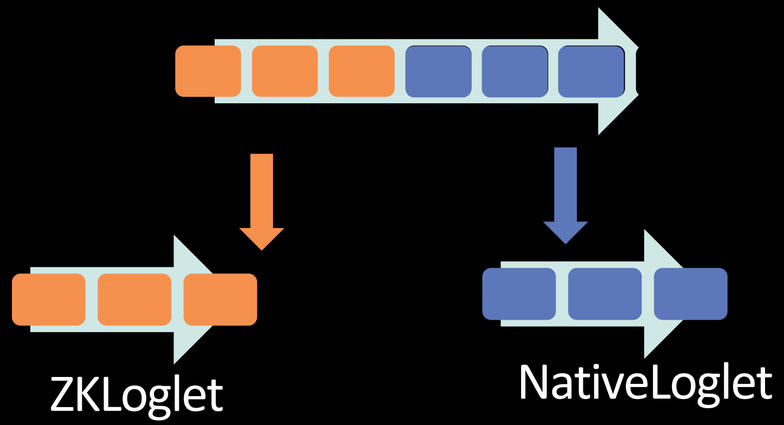


1M+

1KB appends/s with 30 stripes

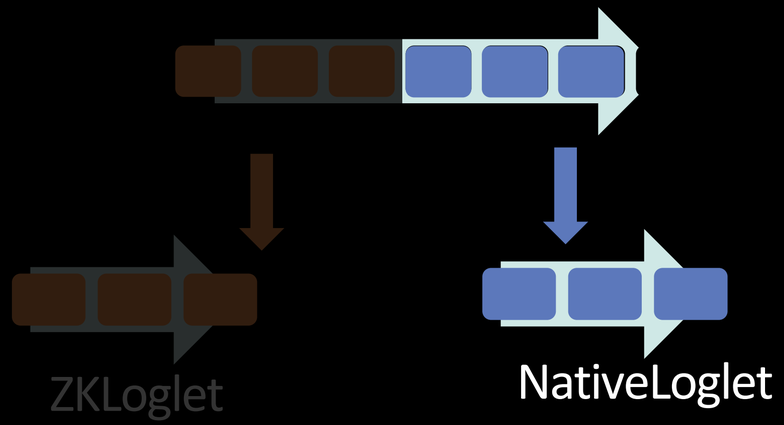
sharded acceptors

trimming the VirtualLog

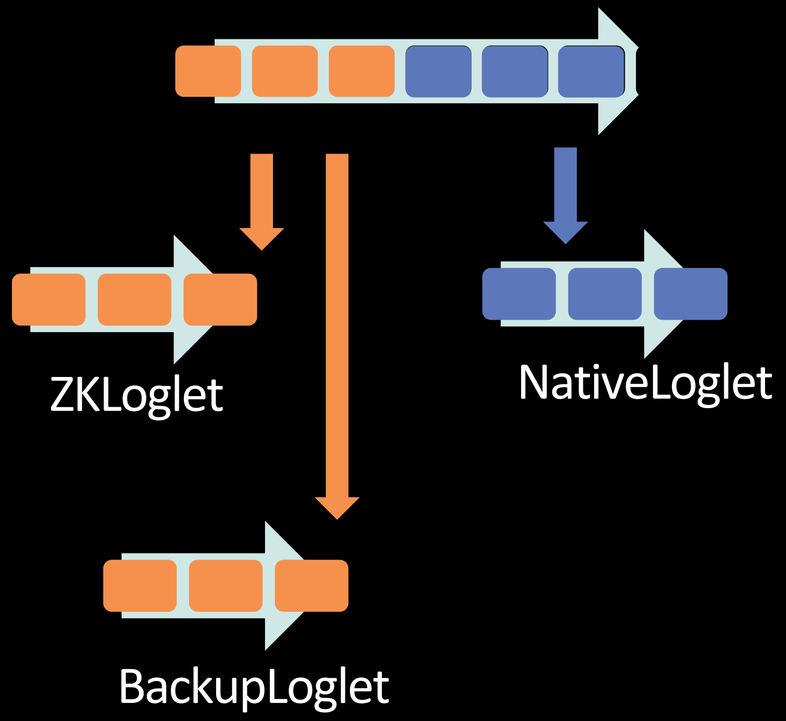


trimming the VirtualLog

trim cold segments



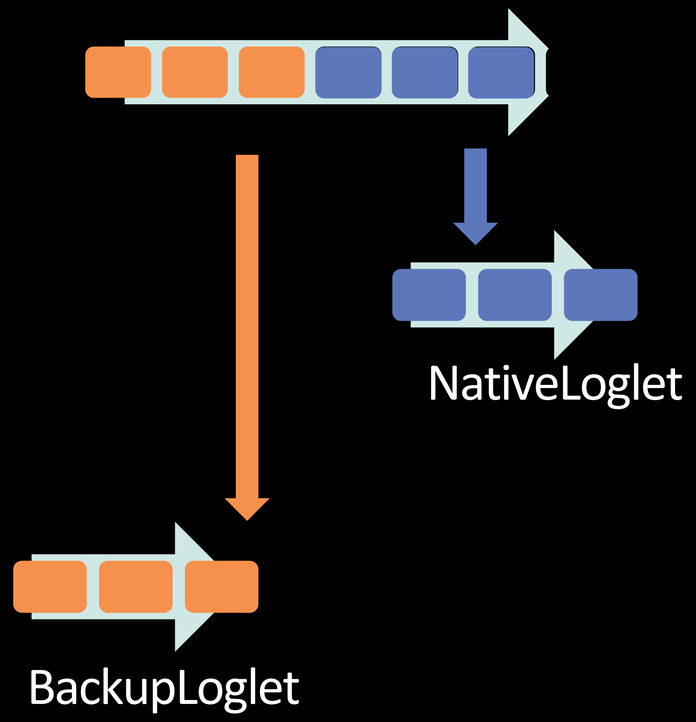
trimming the VirtualLog



trim cold segments

remap cold segments
- InfiniteLog → PiT restore
- more durability

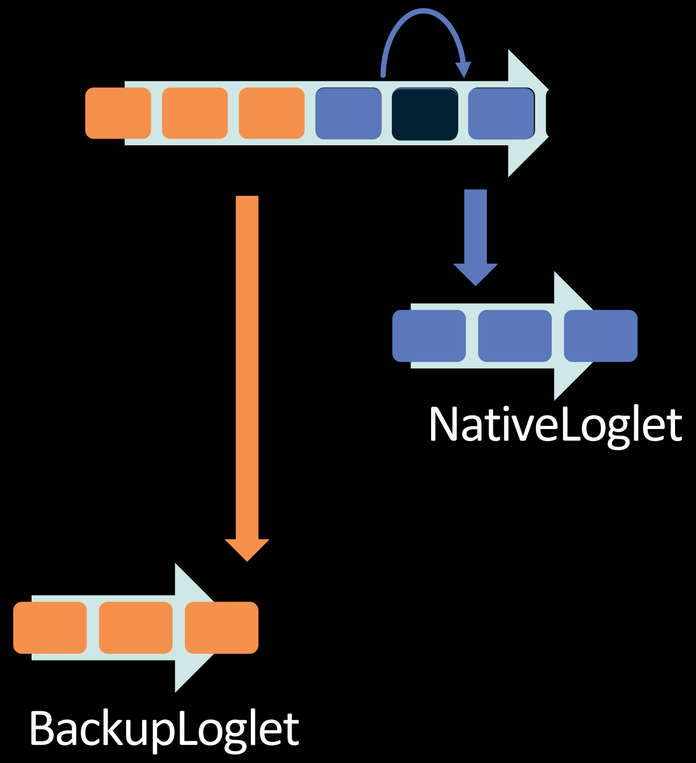
trimming the VirtualLog



trim cold segments

remap cold segments
- InfiniteLog → PiT restore
- more durability

trimming the VirtualLog



trim cold segments

remap cold segments
- InfiniteLog → PiT restore
- more durability

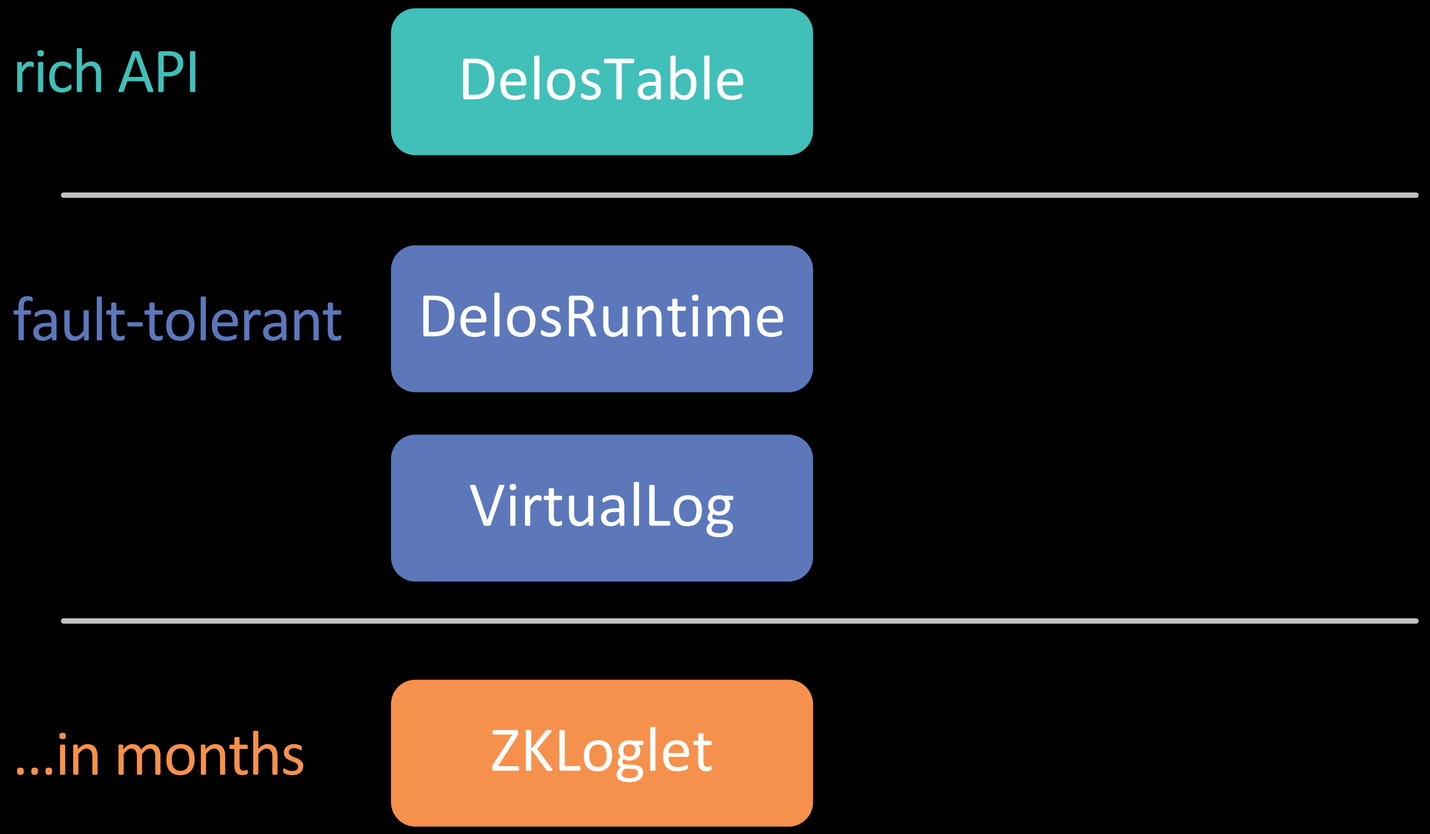
remap single slots
- delete poison pill entries
- less durability

Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*

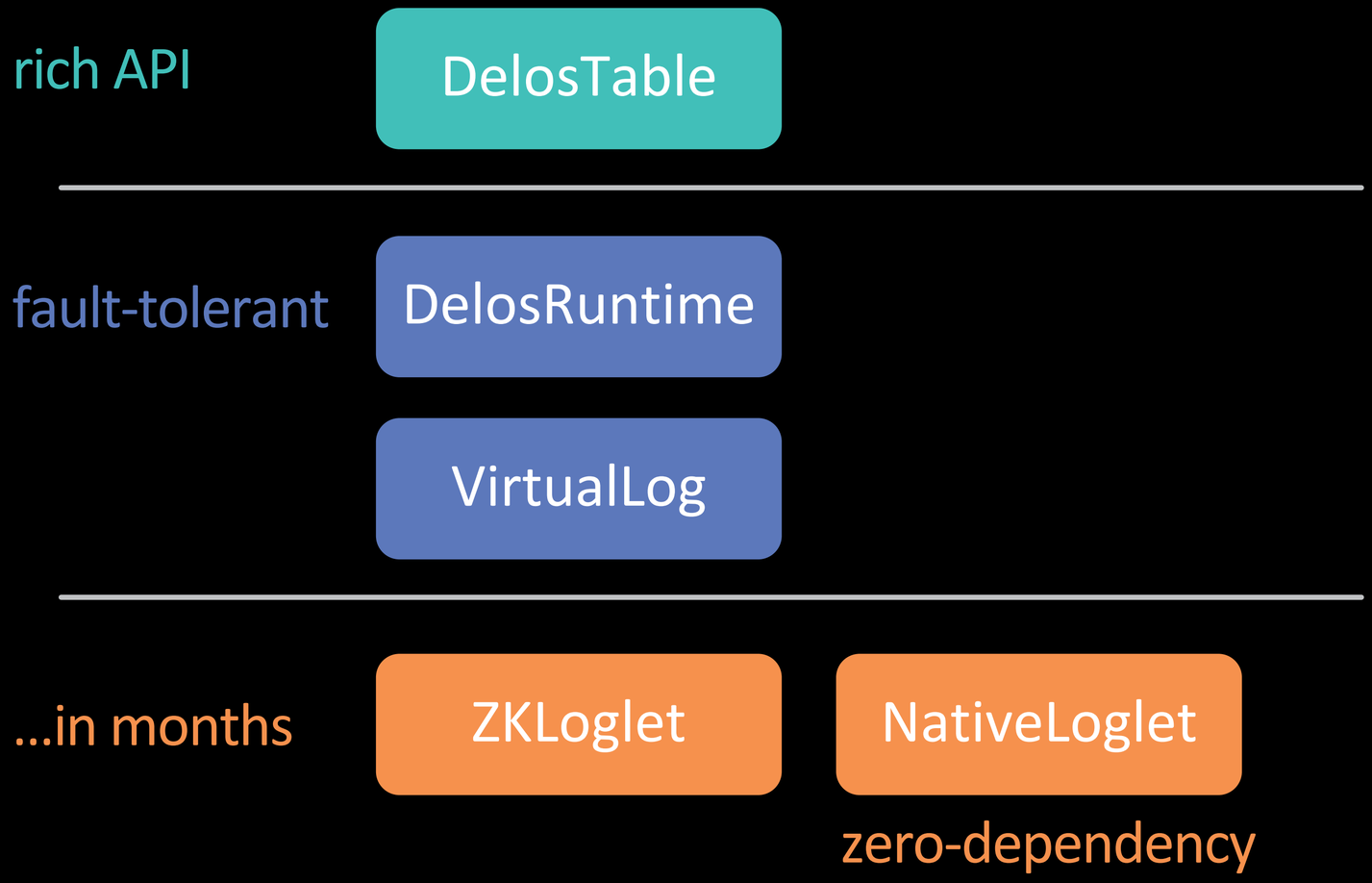
Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*



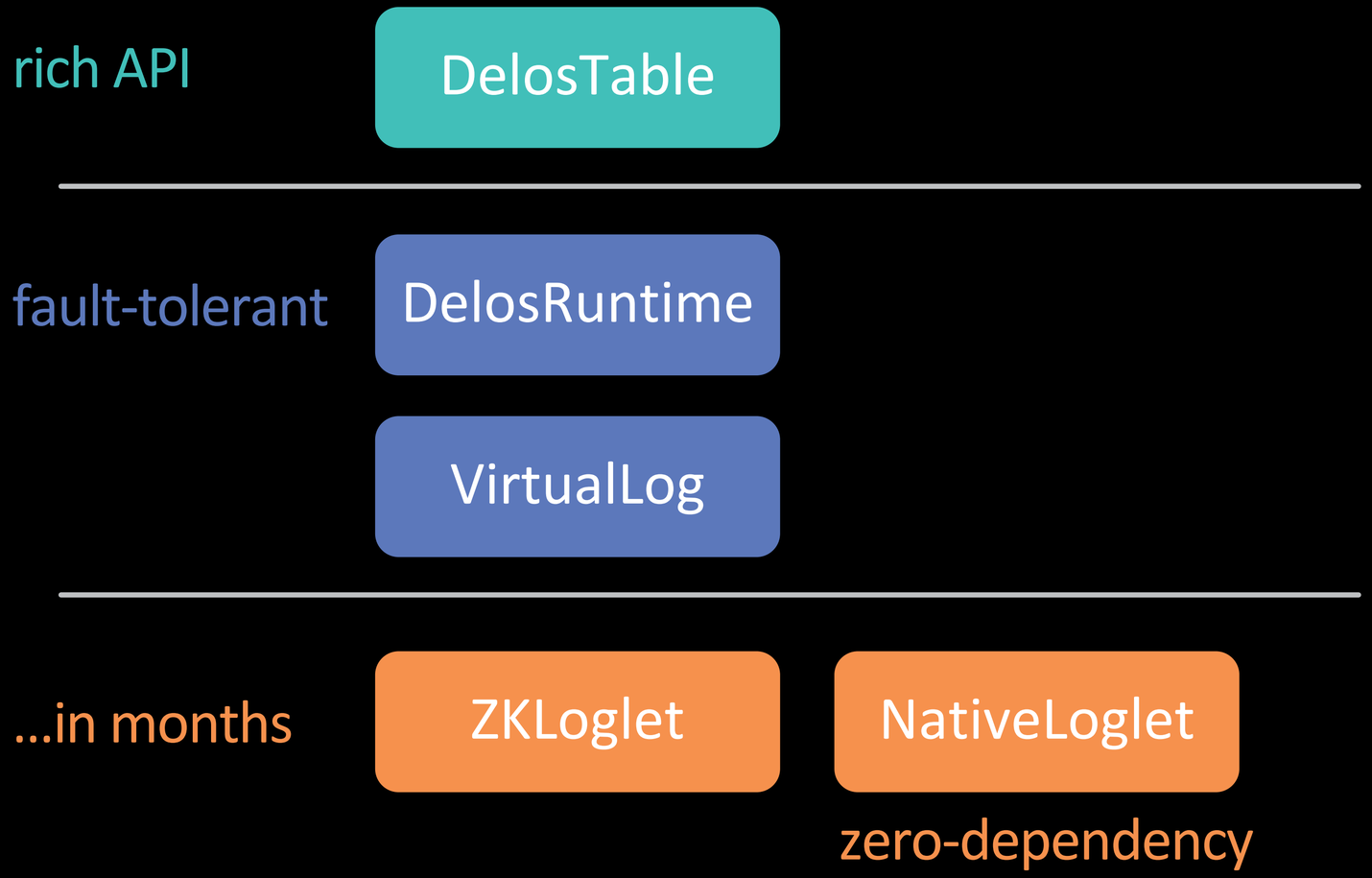
Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*



Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*

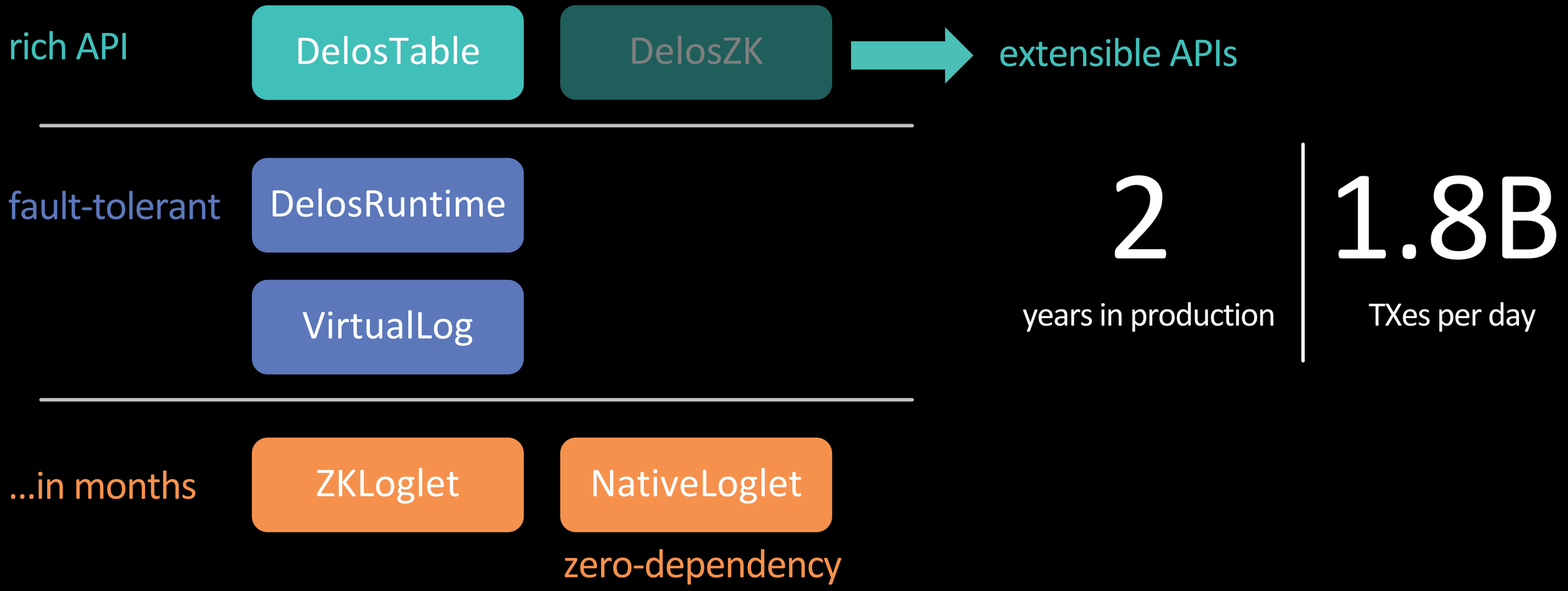


2
years in production

1.8B
TXes per day

Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*

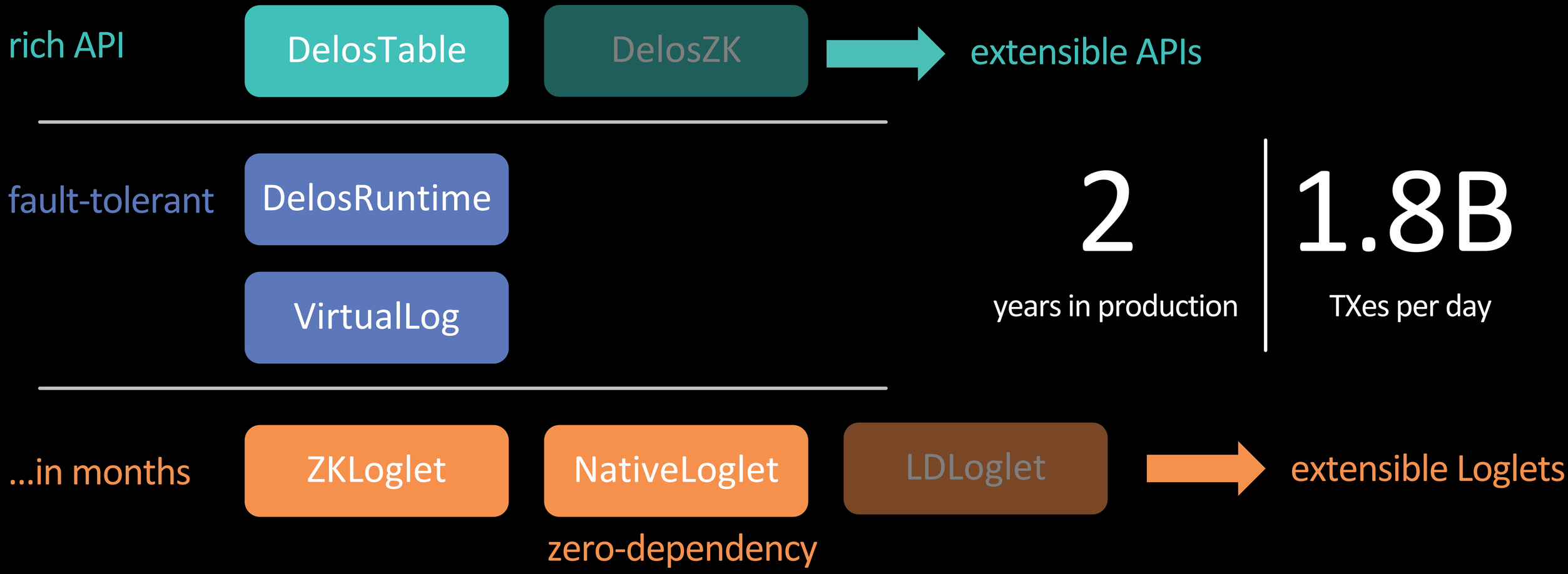


2
years in production

1.8B
TXes per day

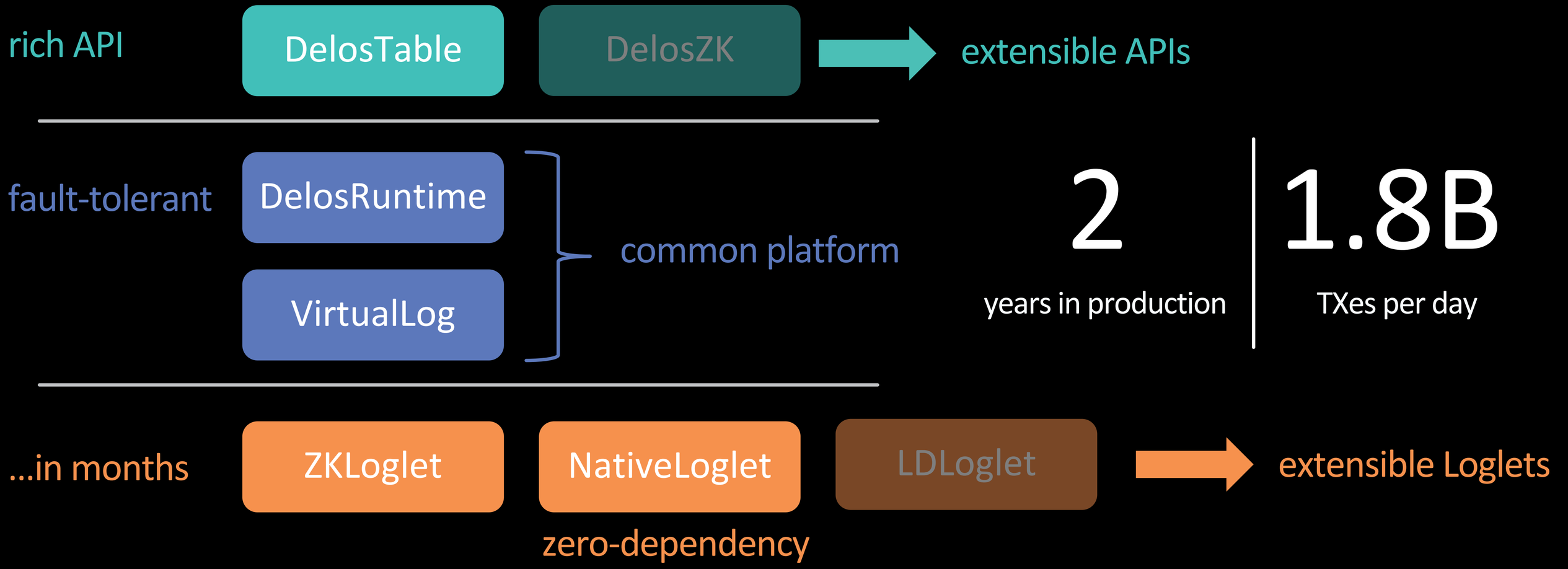
Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*



Delos as a platform

original goal: can we build a zero-dependency, fault-tolerant system with a rich API… *in months?*



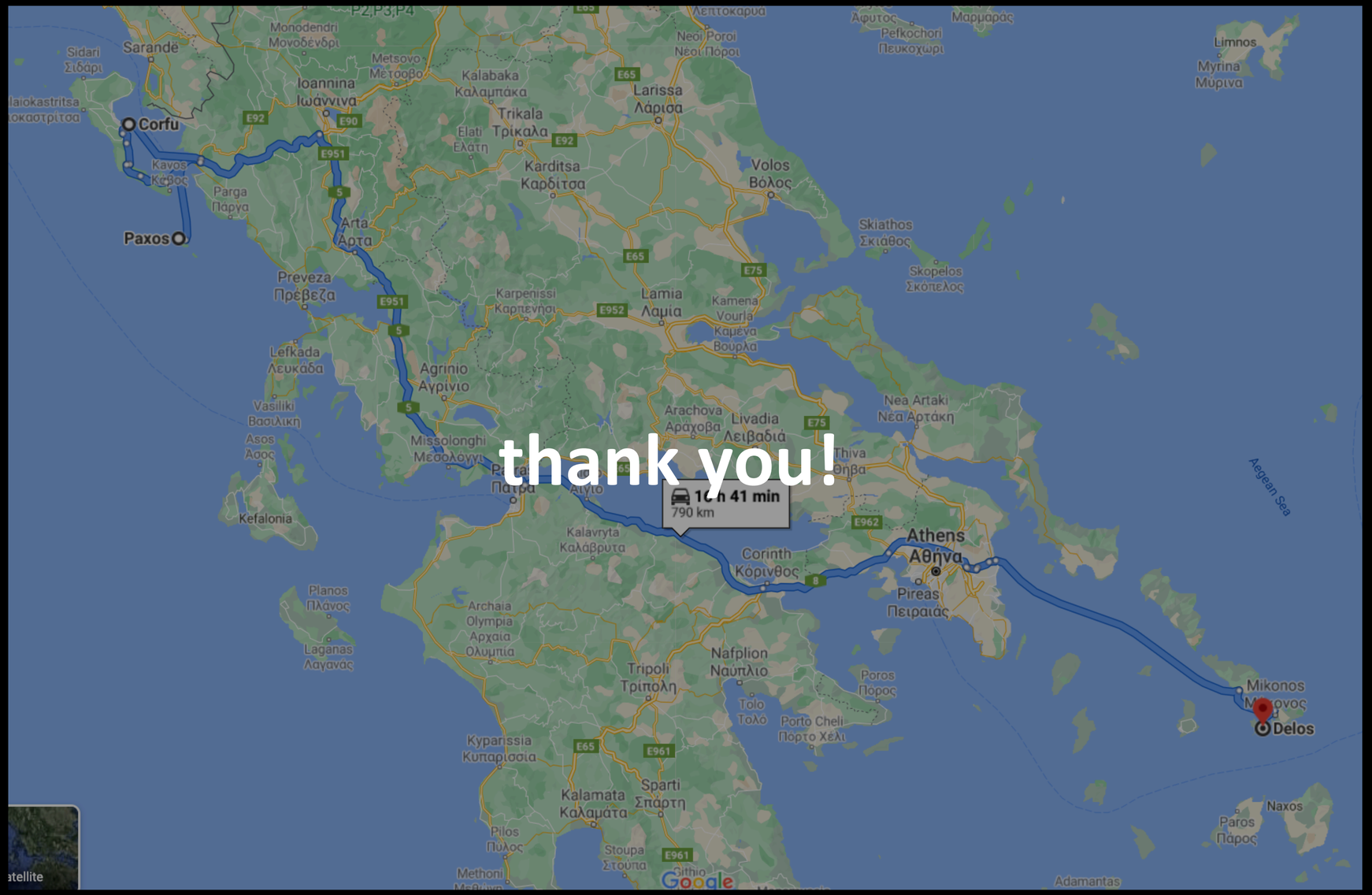
conclusion

Delos is a new storage system at the bottom of the Facebook stack

virtualizing consensus allowed us to develop and deploy new protocols

production benefits immediately from new research…

…new research can reach production quickly



contact: mbalakrishnan at fb.com