vCorfu

A Cloud-Scale Object Store on a Shared Log

Michael Wei, Amy Tai, Christopher J. Rossbach, Ittai Abraham, Maithem Munshed, Medhavi Dhawan, Jim Stabile, Udi Wieder, Scott Fritchie, Steven Swanson, Michael J. Freedman, Dahlia Malkhi

UCSD CSE Computer Science and Engineering

PRINCETON UNIVERSITY

TEXAS The University of Texas at Austin

vmware

Michael Wei

NSDI 2017
March 27th, 2017

Background

Shared Logs and Consistency

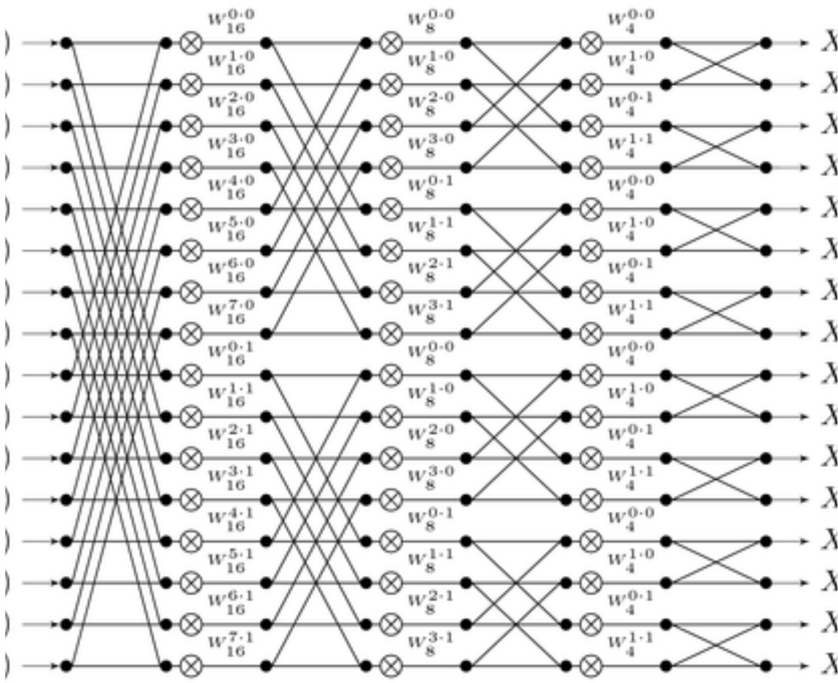
Scalability and Consistency

1 + 1 = 2

1 + 1 = 2

1 + 1 = 2

1 + 1 = ?

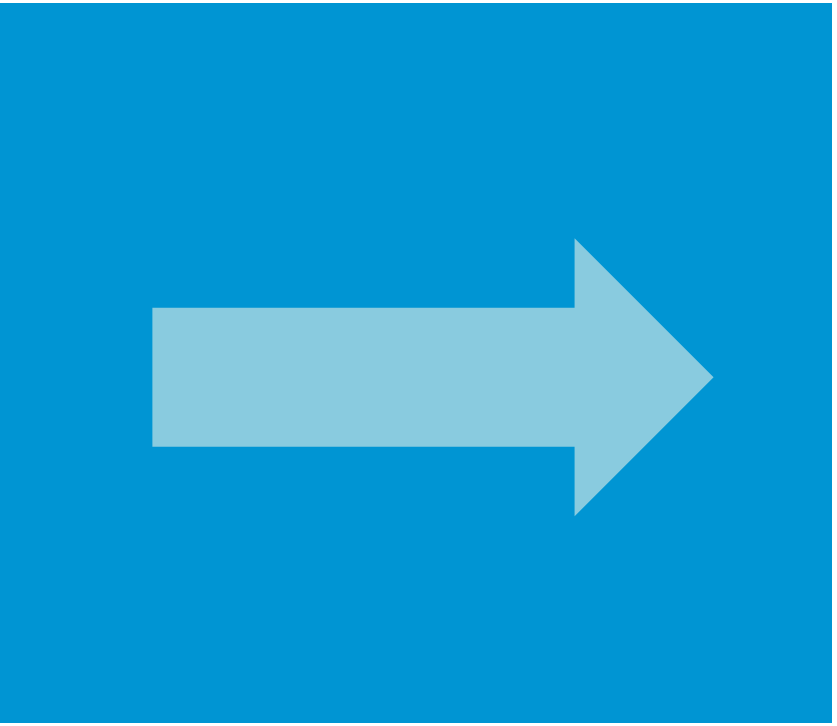


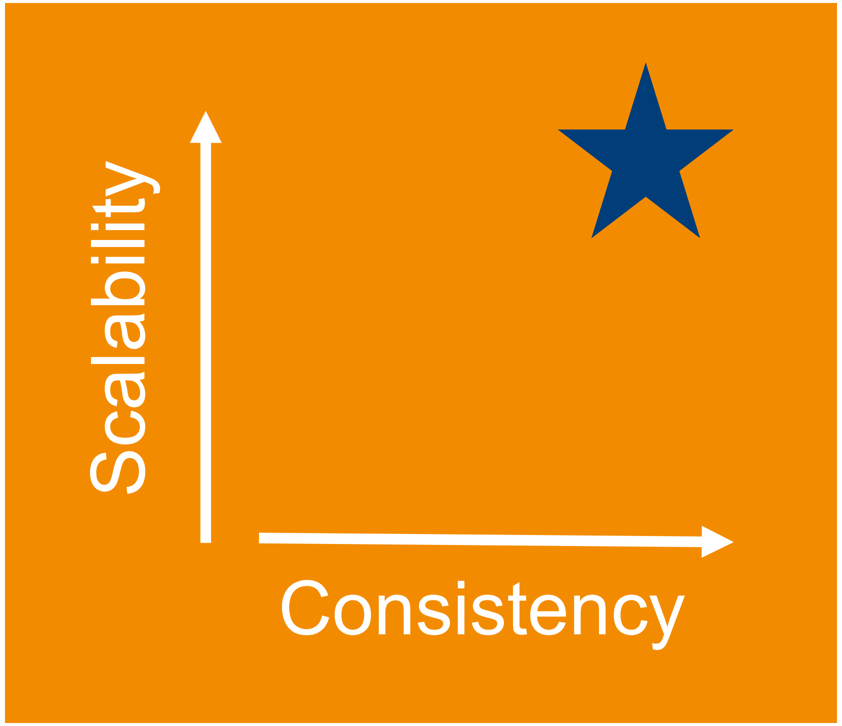
Systems which are scalable…

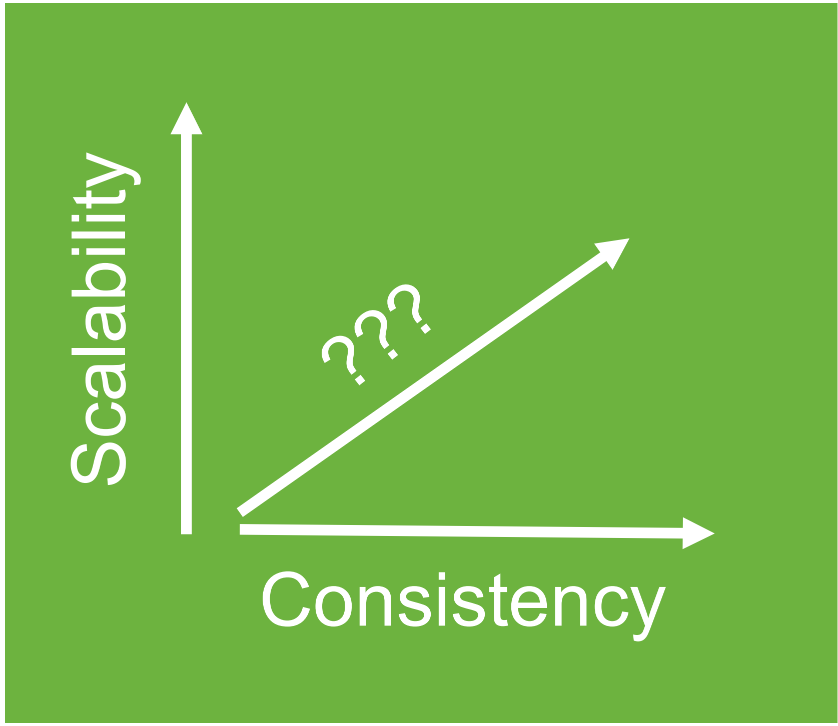
yet consistent…

can be difficult to build

Shared Log Systems







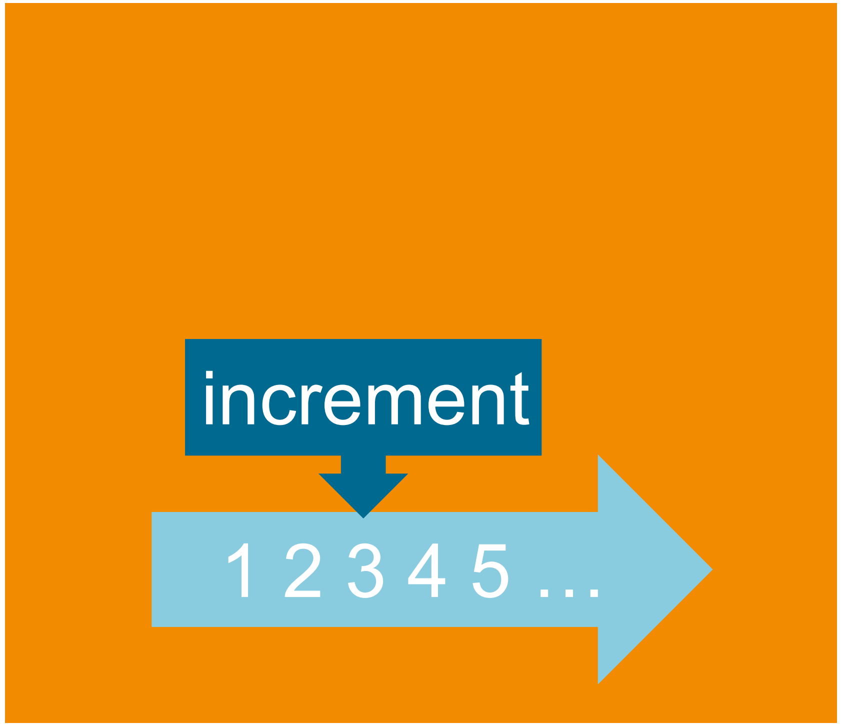
Shared log systems represent a point in the design space…

which provides scalability without compromising consistency…

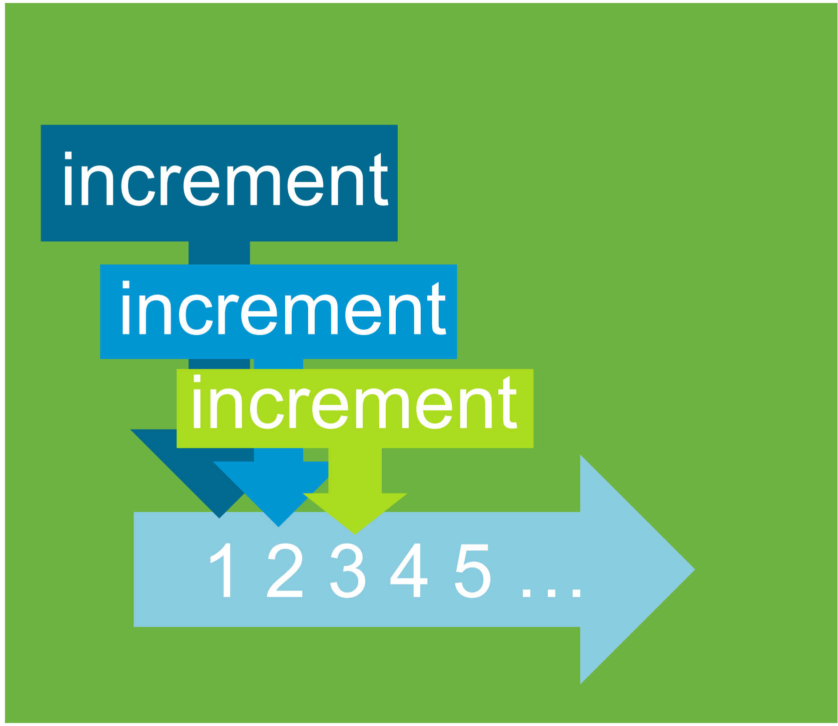
however, these systems make a different set of tradeoffs

Writing vs Reading

While writing to a shared log provides strong consistency and >1/2M appends/s,

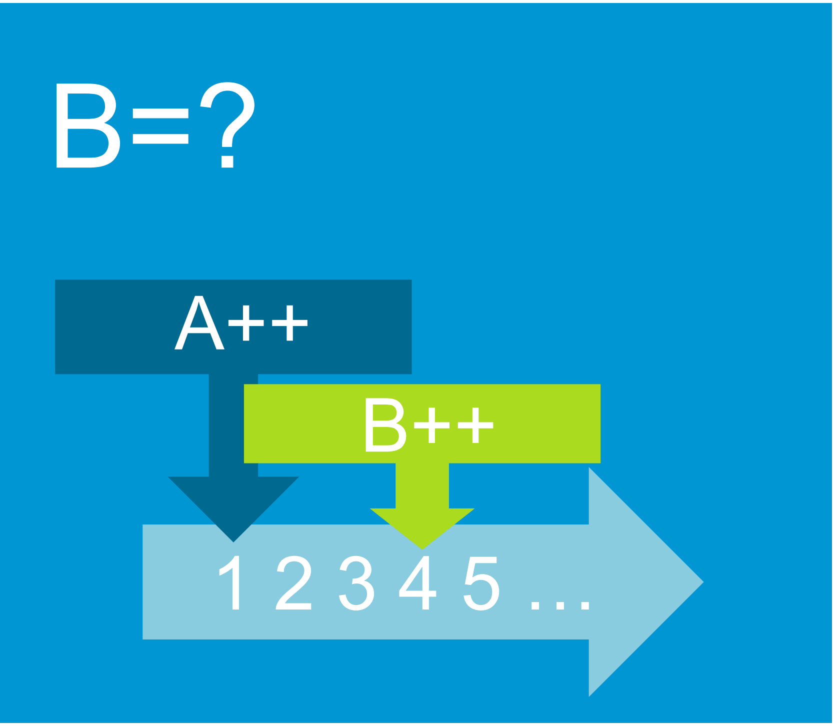


To provide the strongest level of consistency, only updates are logged,

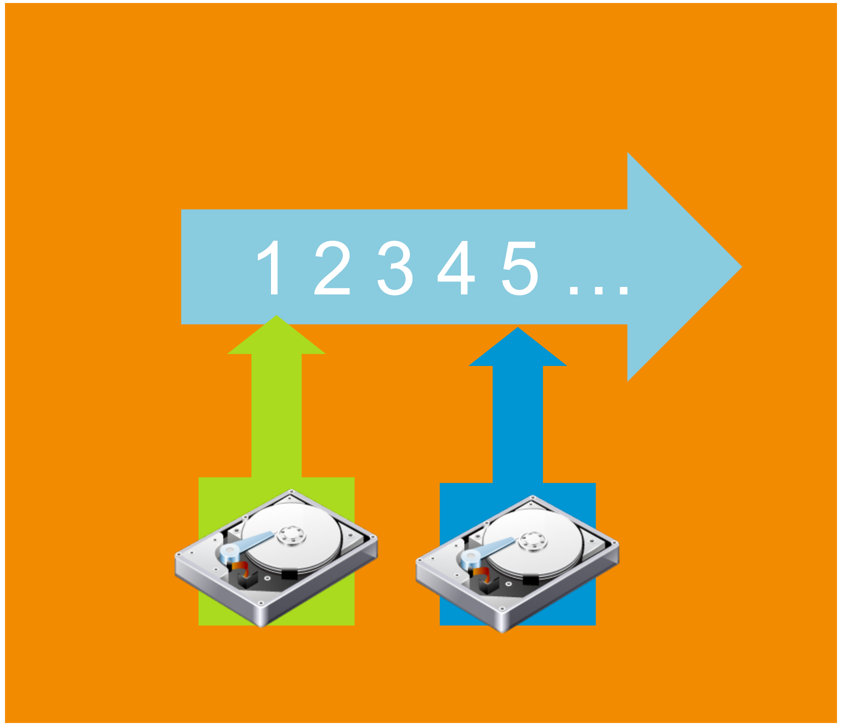


So reads are more expensive, as clients now have to read multiple updates

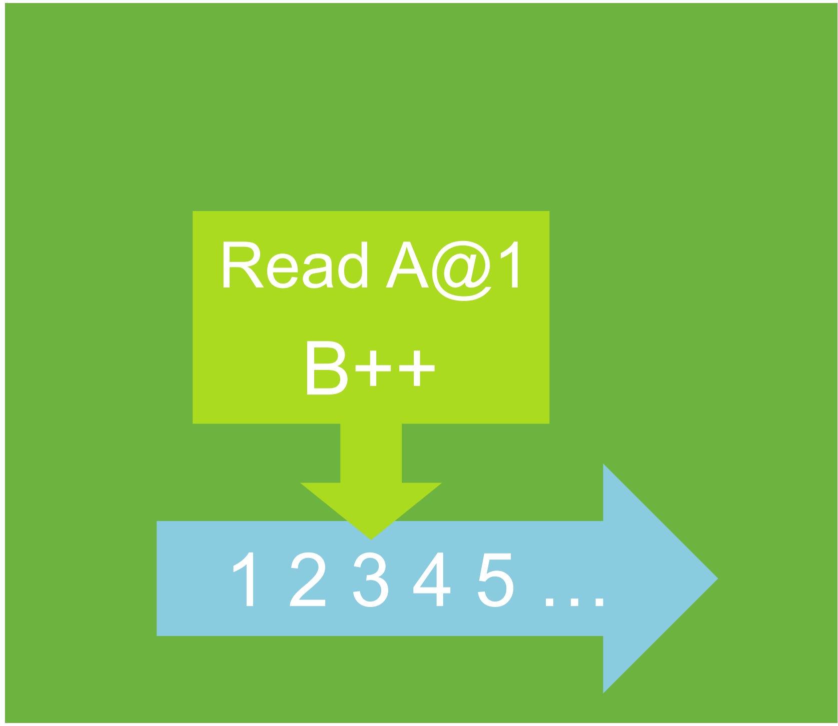
Improving Read Scalability



Clients may read unnecessary updates to service requests

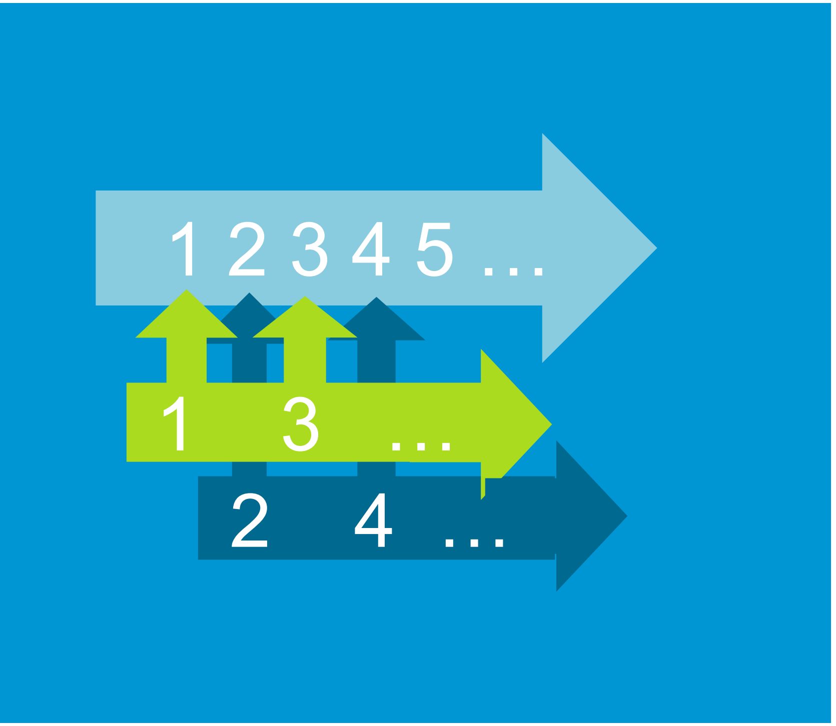


There is no locality, so clients will have to jump around on the log

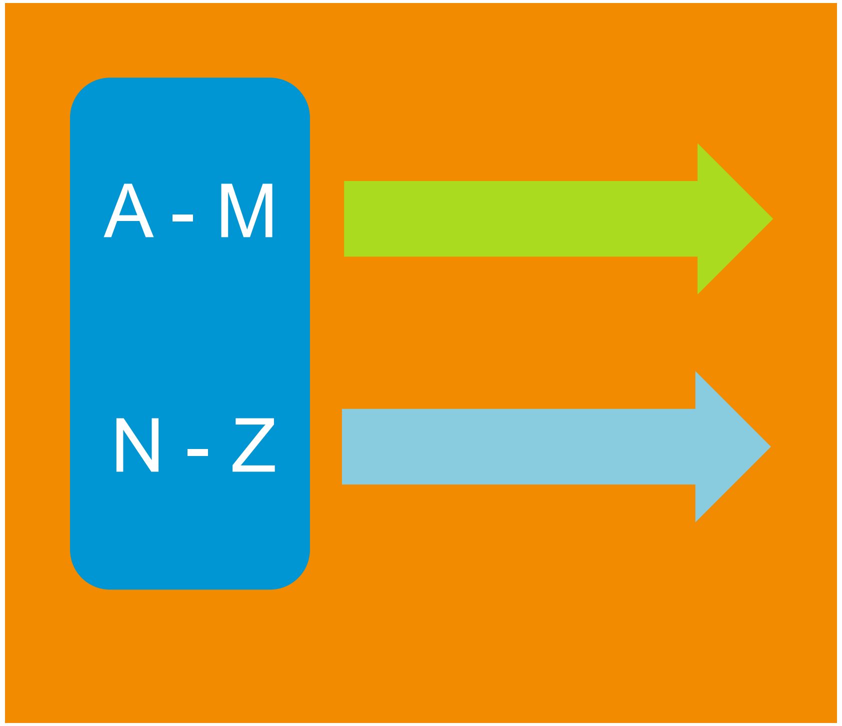


Clients have to do more work to figure out the results of a transaction

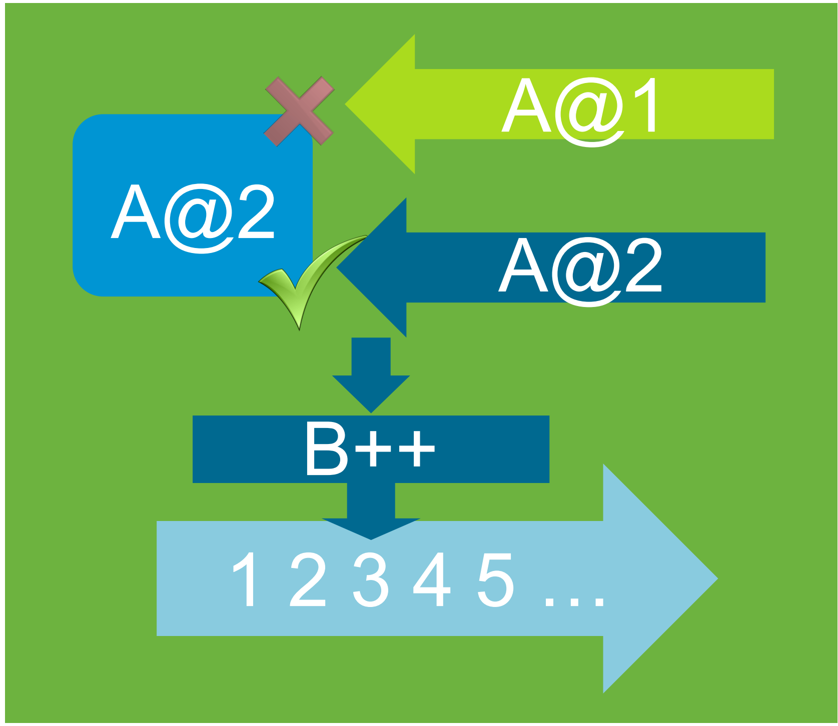
vCorfu addresses read scalability by…



Stream materialization,
which localizes related updates and enables reads without playback

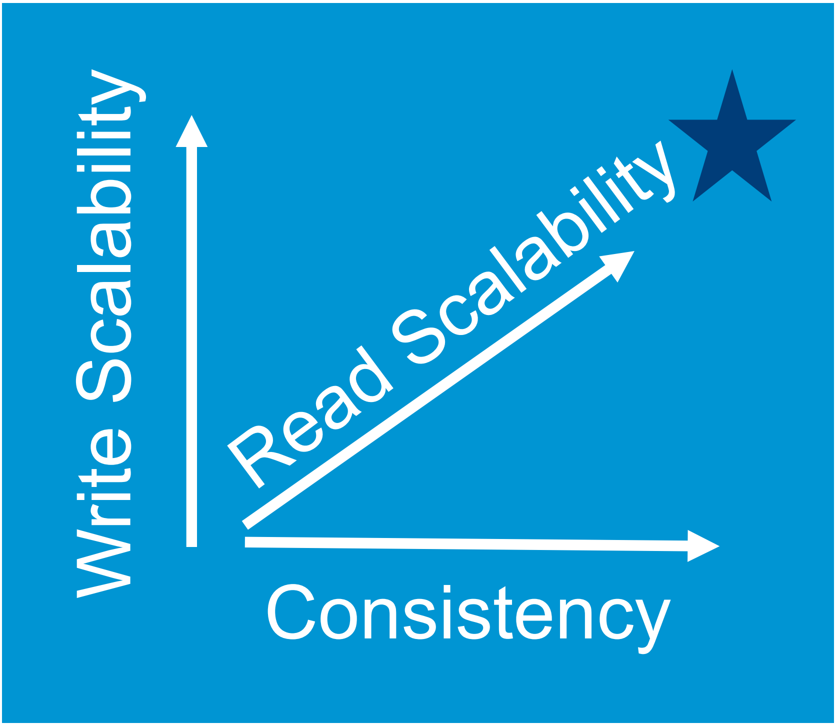


***Composable SMR**,*
which enables large state machines without forcing clients to replicate the entire state machine

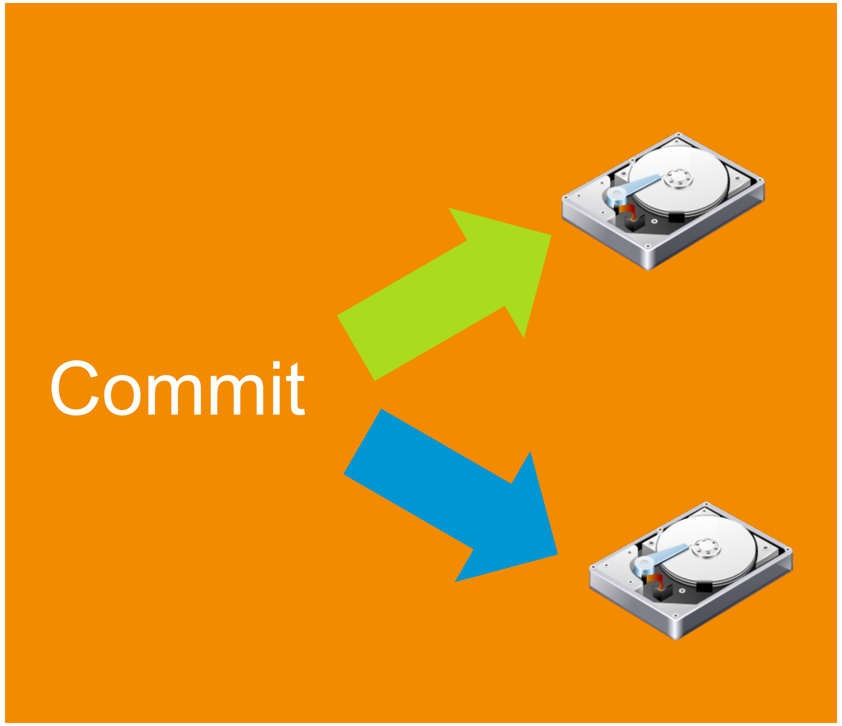


Lightweight Transaction Resolution,
which eliminates the need for clients to determine whether transactions in a log were aborted

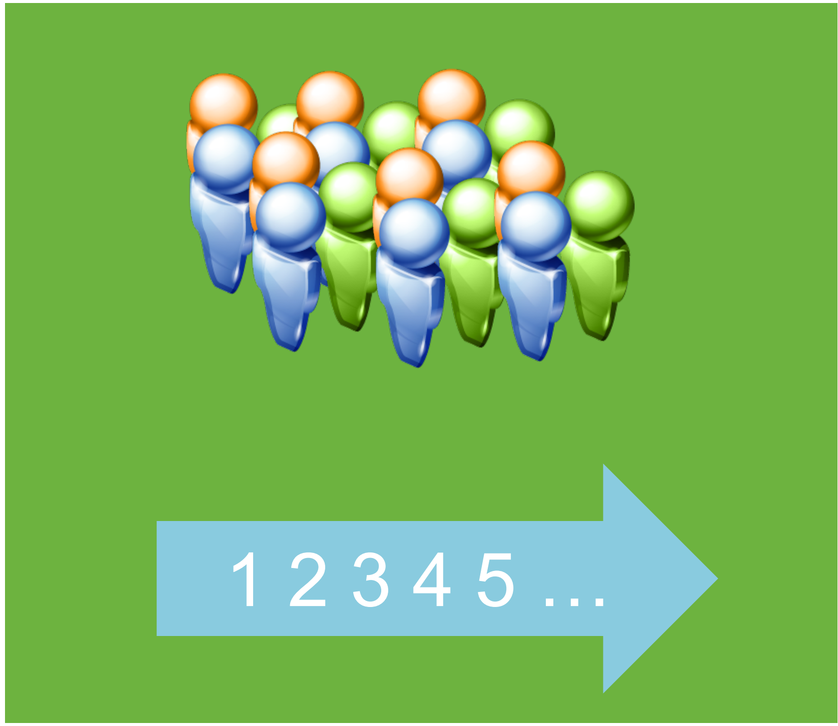
vCorfu Offers Another Point in the Design Space



Different point in the design space



Better read scalability, but at a penalty to writes



We can now service more clients without consuming the entire log

…and we will show

That we can now scale shared log systems to cloud-scale data sets

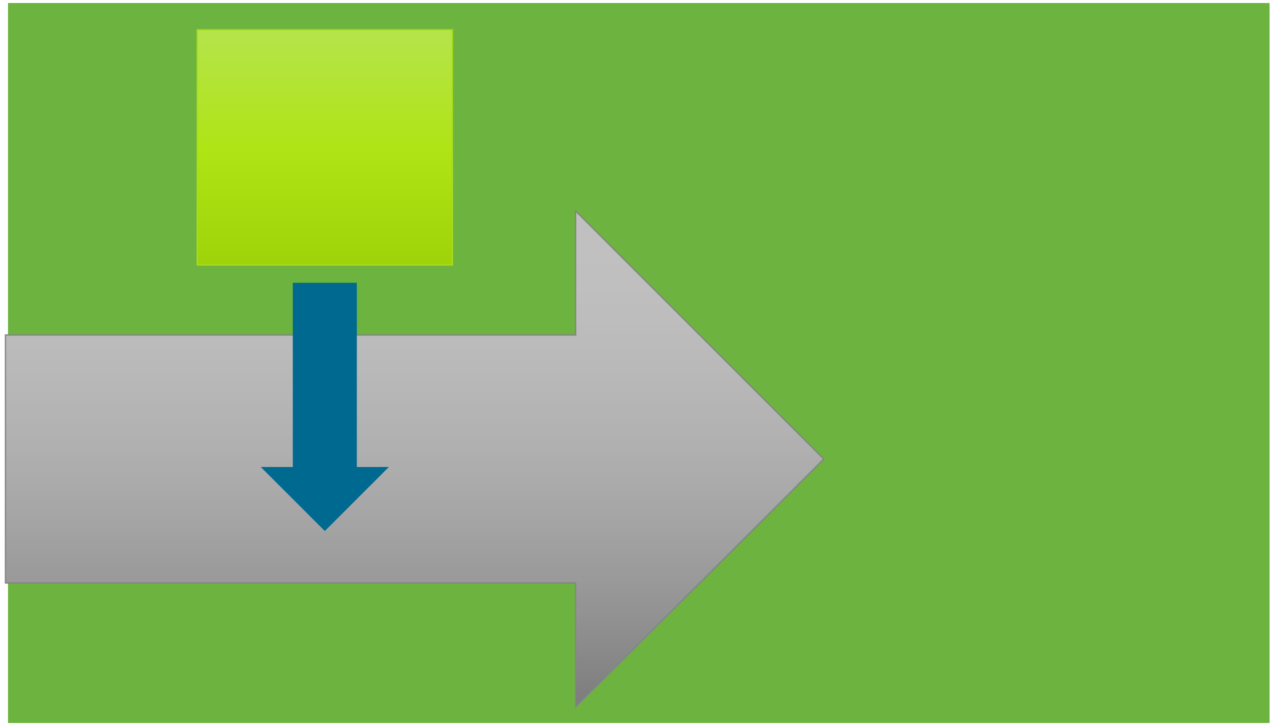
Offer comparable performance to, and often outperform state-of-the-art NoSQL systems

While retaining the strong consistency benefits of a shared log

Shared Log Systems

Interface and Approach

Shared Log [1] Basic Operations

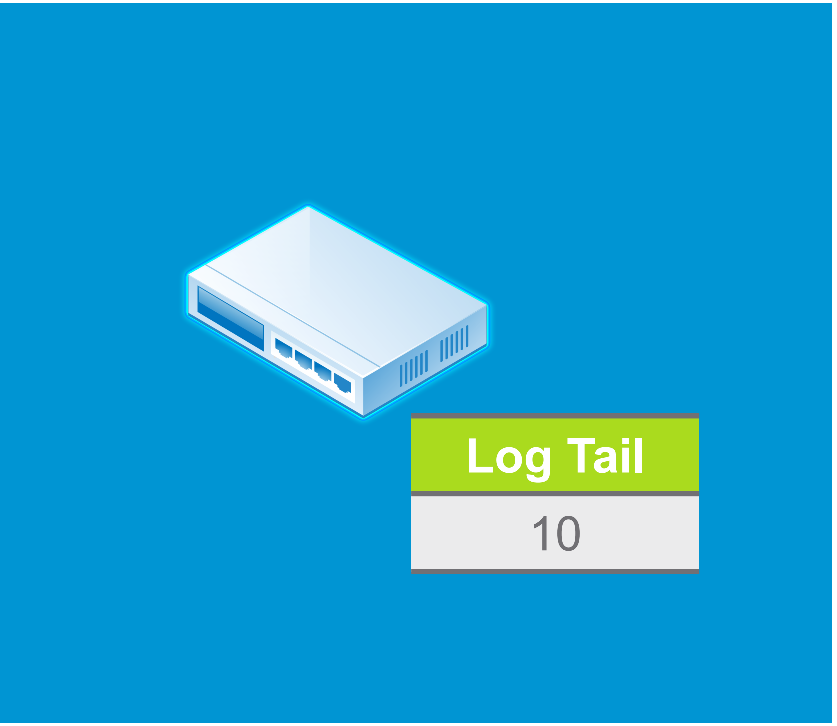


Read(address):
Read an entry from the log

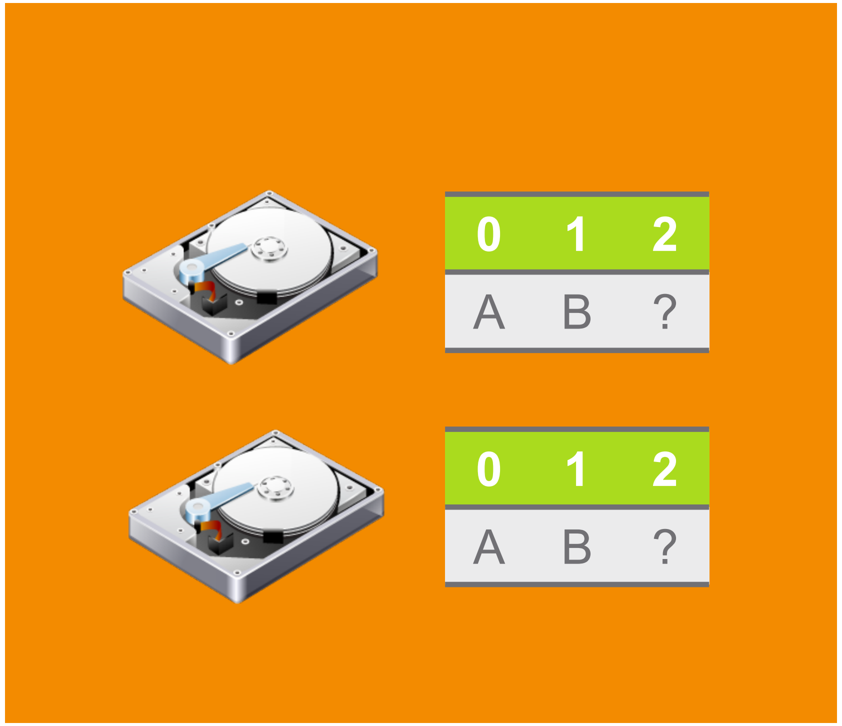
Append(entry):
Append an entry to the log and return the address it was written at

[1] **CORFU: A Shared Log Design for Flash Clusters.** *(NSDI '12)*

Shared Log Systems are Composed of…



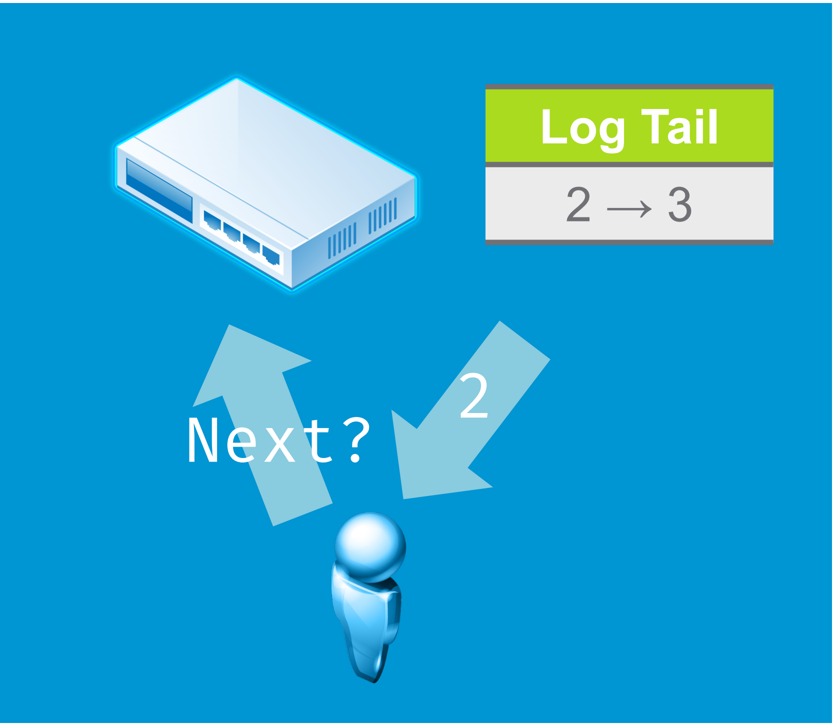
Sequencer,
which issues addresses in a log



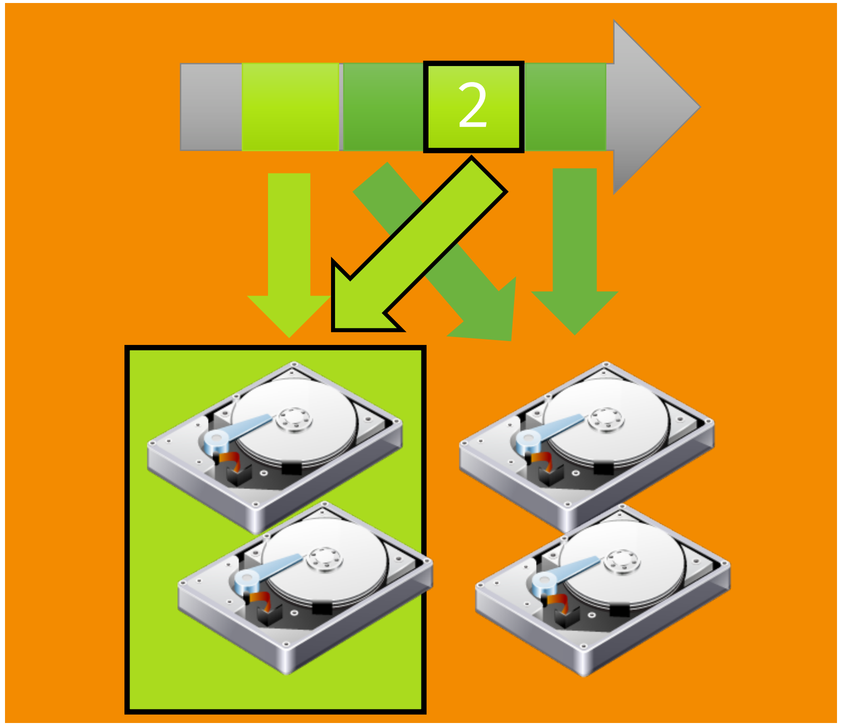
***Log replicas**,*
which store data in the log

Layout,
which maps addresses in the log to log replicas

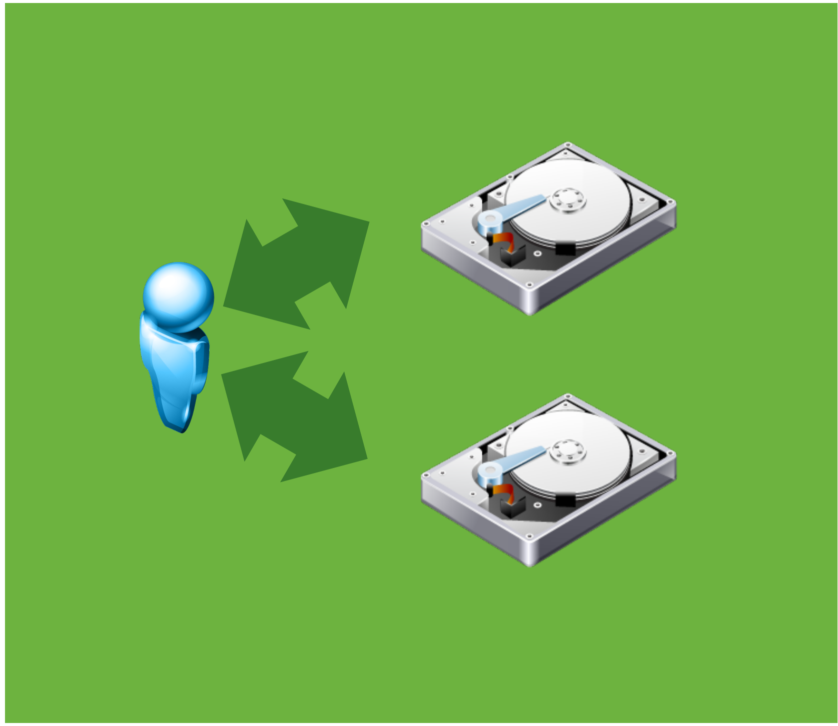
To Append to a Shared Log, Clients…



First contact the sequencer, which issues an address

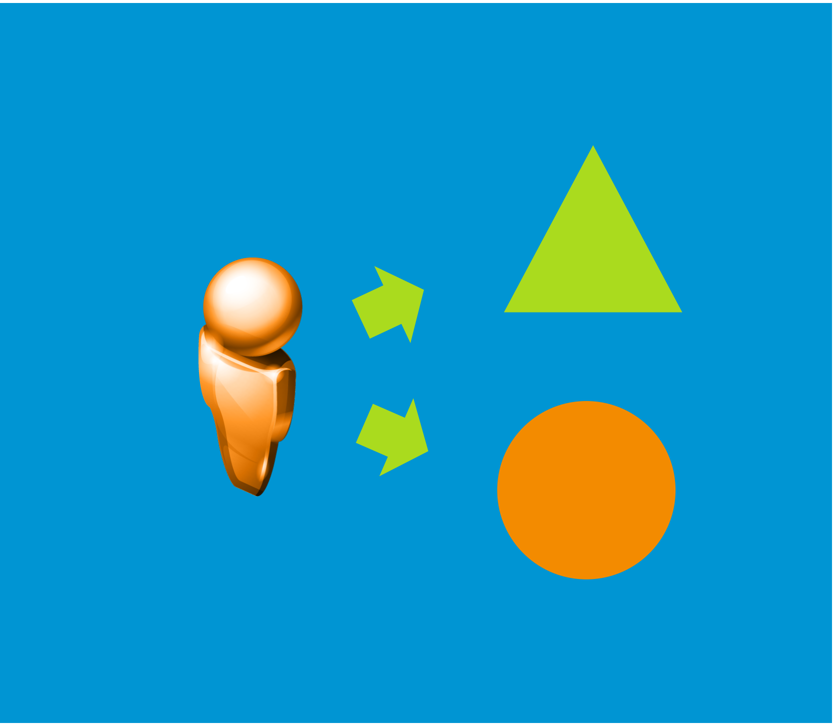


Using the layout, determine which log replicas to write to

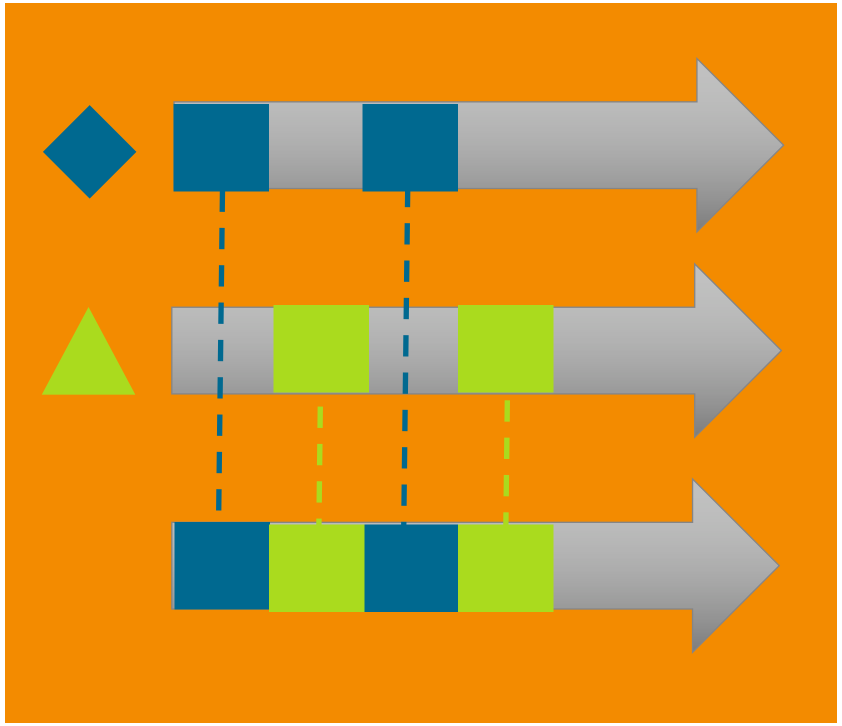


Perform a write using the address given by the sequencer

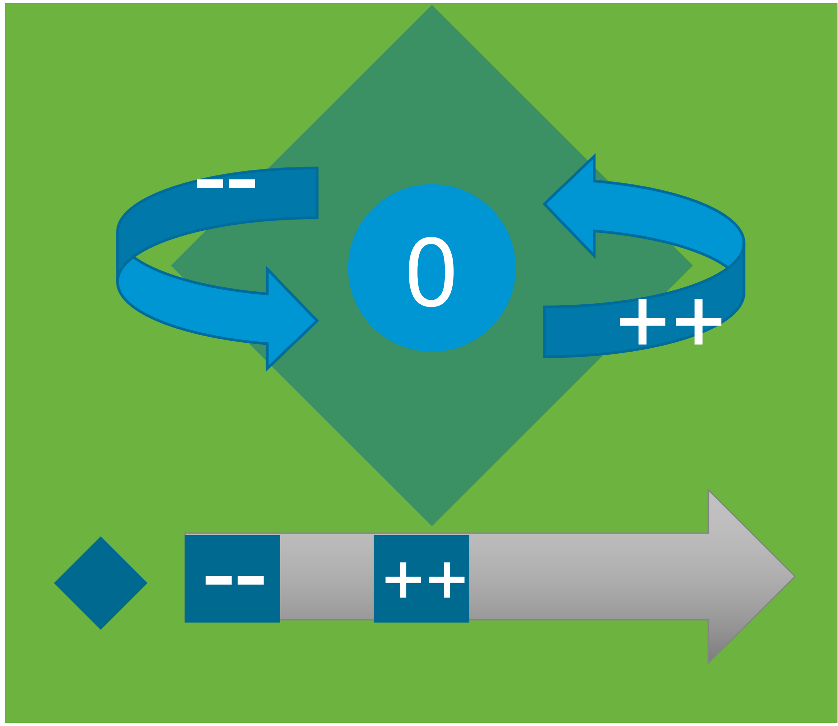
We Take the Tango [2] Approach…



Clients don’t interact with the log directly, rather, they interact with objects



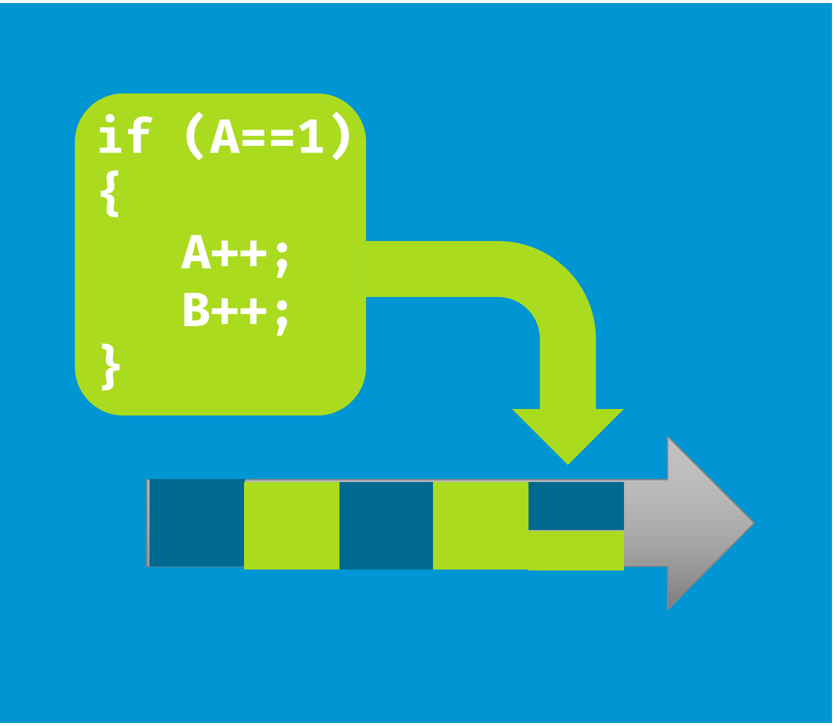
Objects are stored in virtualized logs called streams



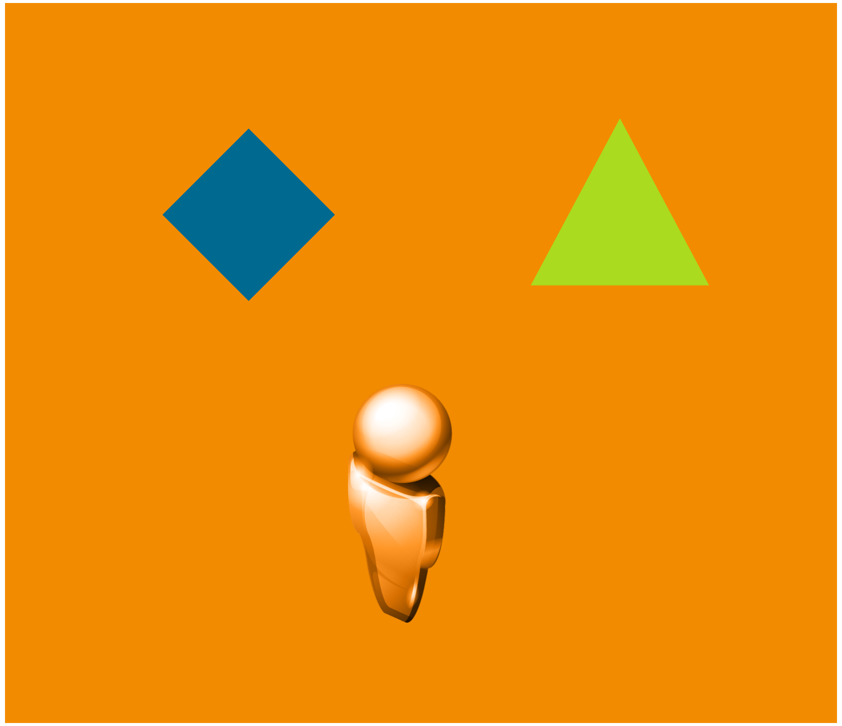
Entries in the stream represent updates to the object state

[2] **Tango: Distributed Data Structures over a Shared Log.** *(SOSP ’13)*

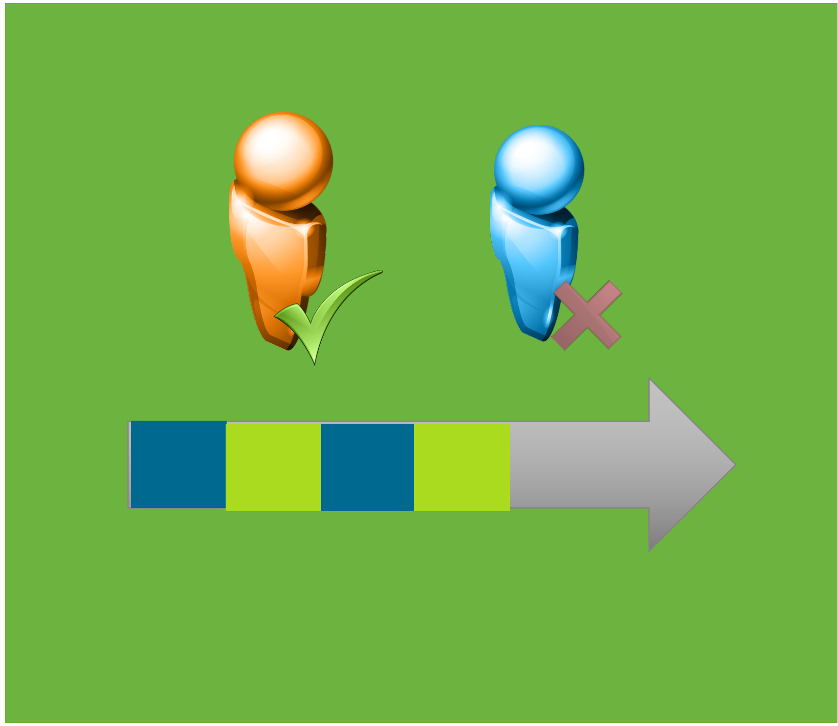
Including Support For Transactions…



The system leverages the log to provide rich support for transactions



Transactions execute optimistically on the client's in-memory views



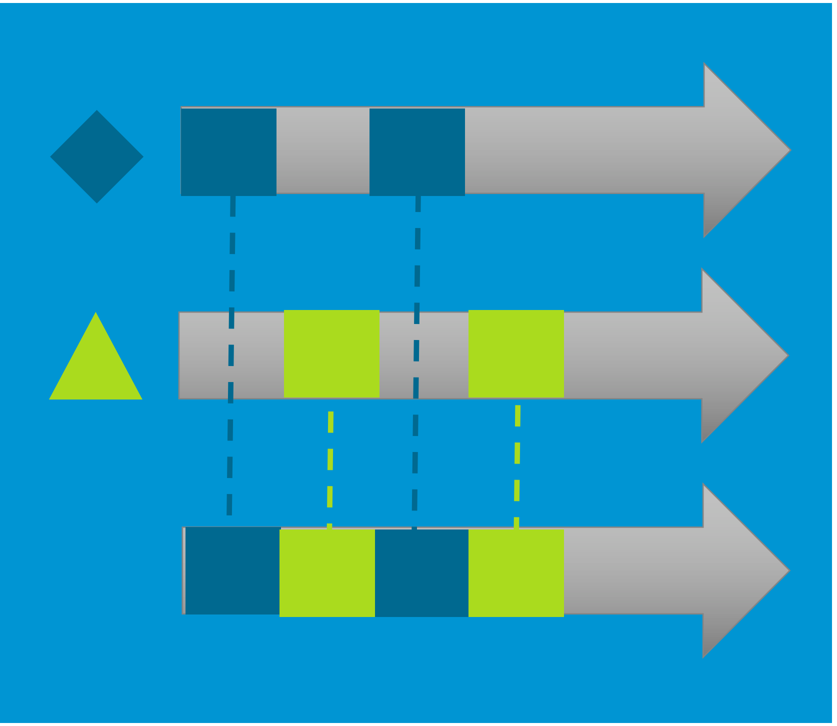
And the log serves as the ground truth in case of conflicts

vCorfu Stream Store

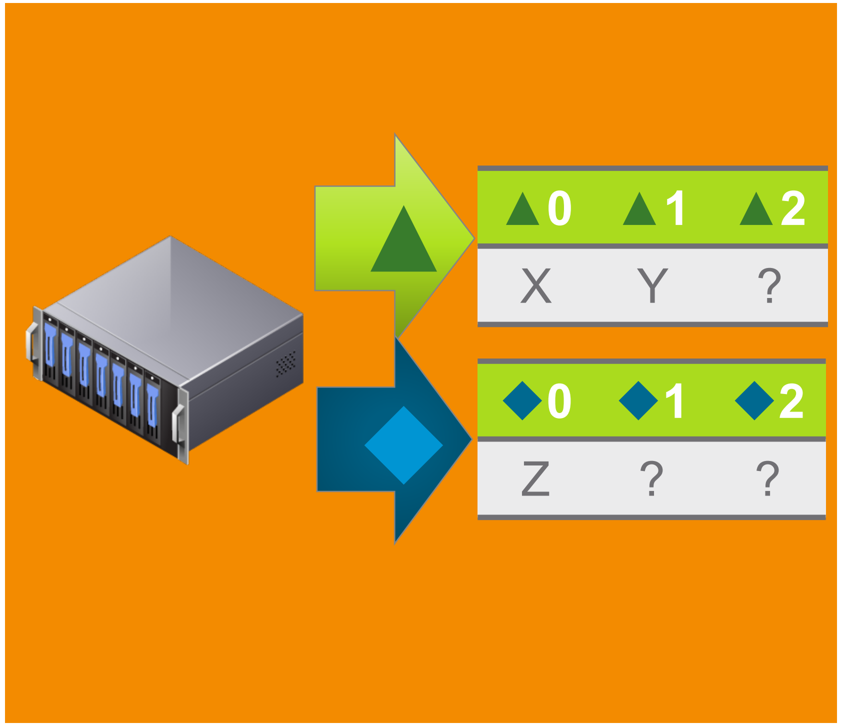
Architecture and Design

vmware®

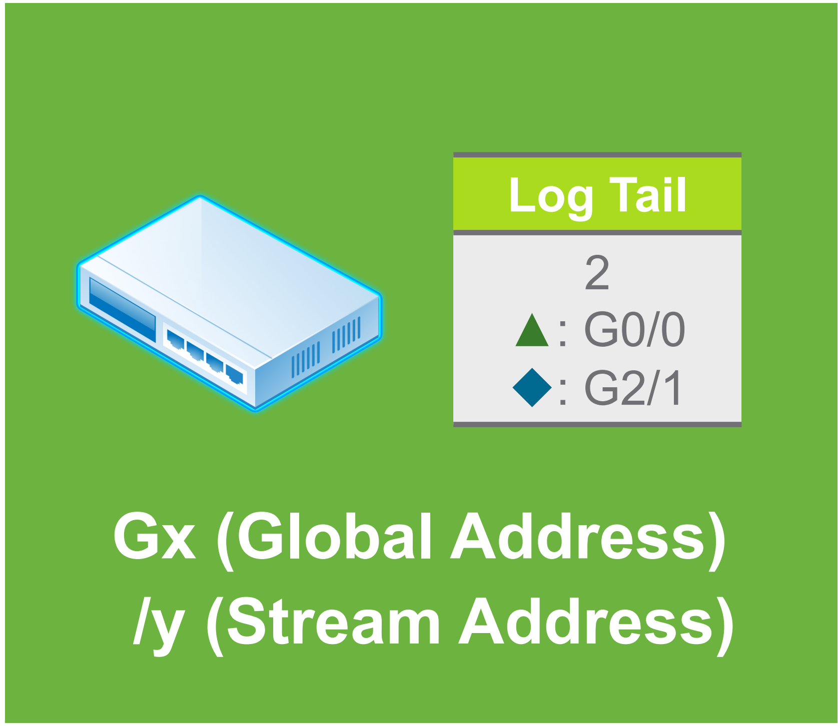
Materialized Streams



In vCorfu, a fundamental building block is a materialized stream

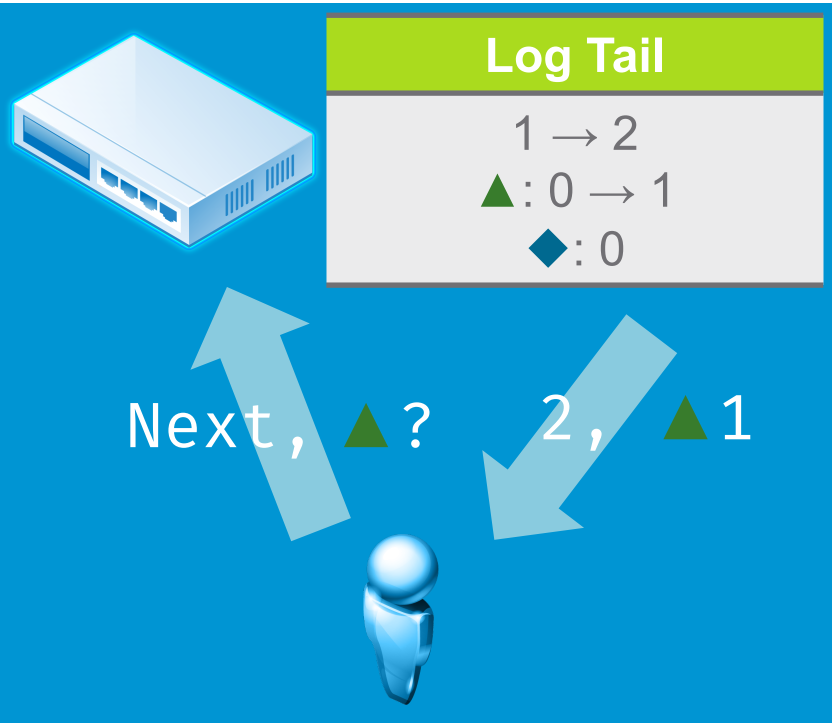


Stream replicas implement the storage for a materialized stream



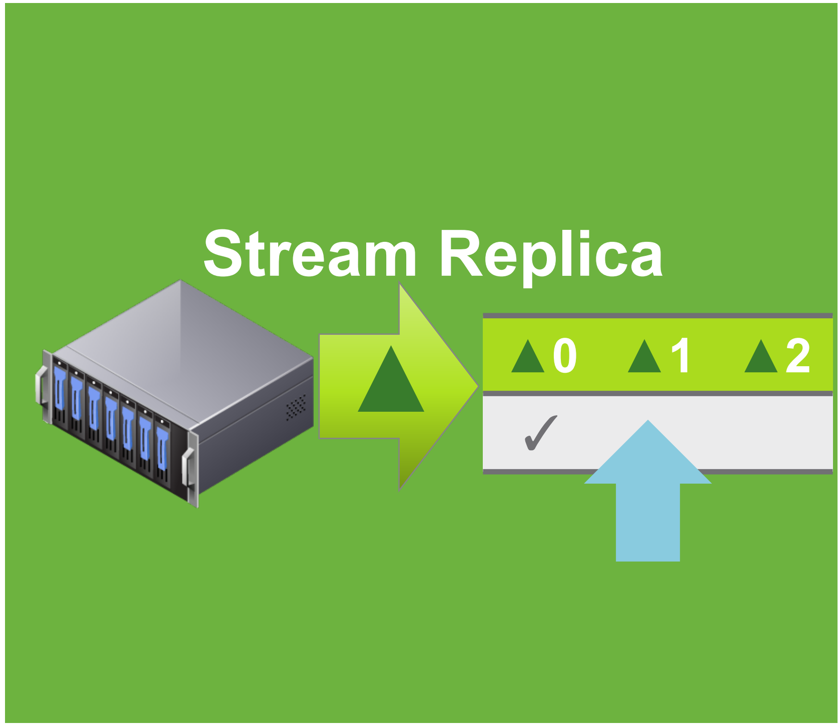
The vCorfu sequencer keeps track of the global tail as well as stream tails (global, stream)

Materializing Streams



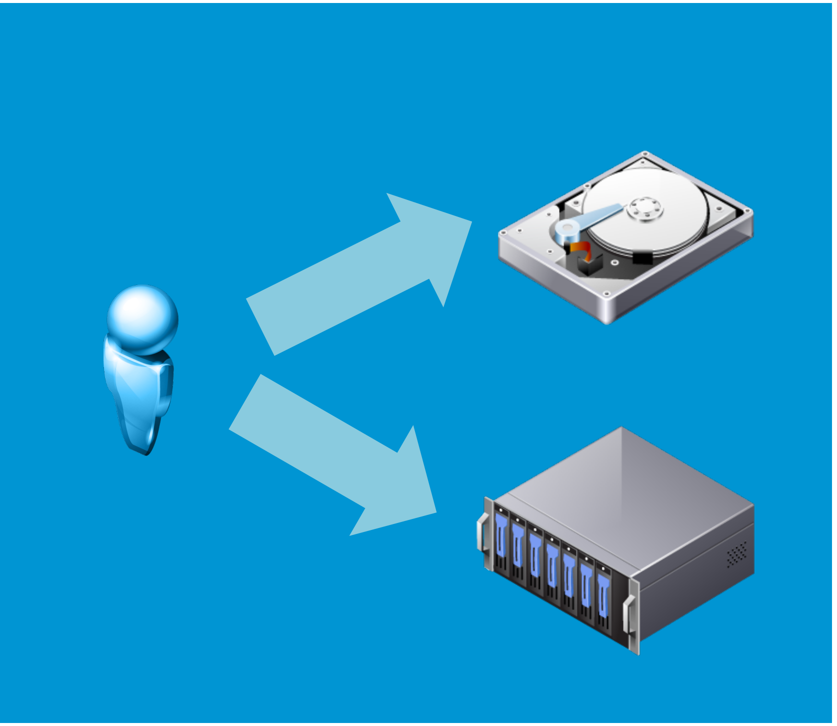
Sequencer issues global address (2) and stream address (▲1)

Write to log replica using the global address (2)

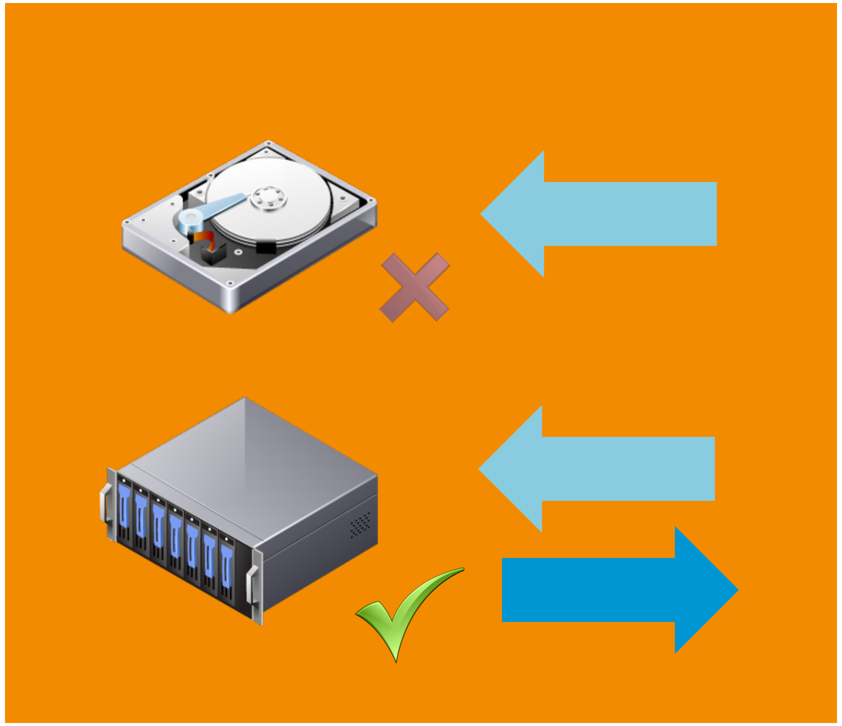


Write to stream replica using the stream ID (▲) and stream address (1)

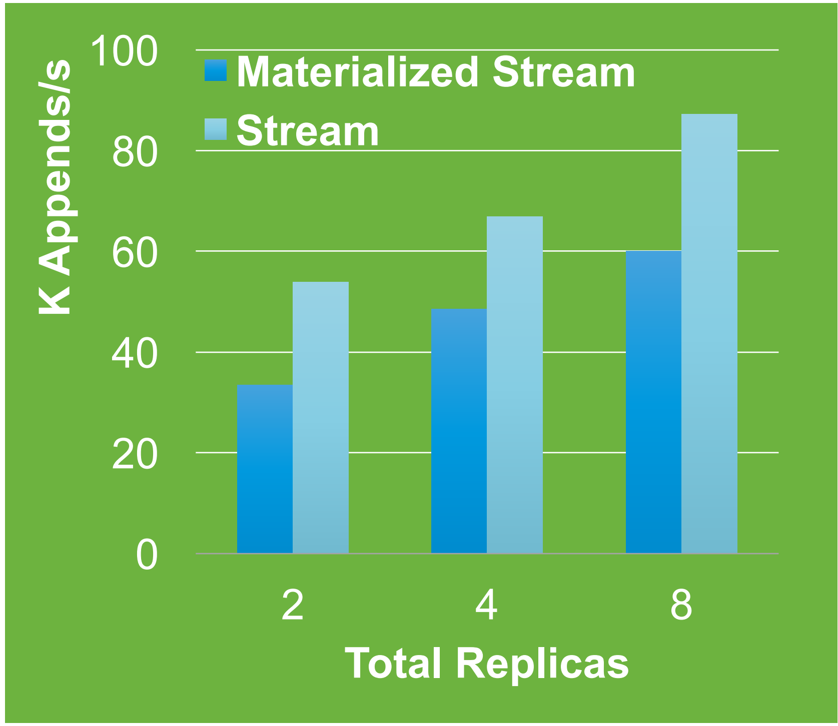
Materializing Streams



Client must commit data to every log and stream replica



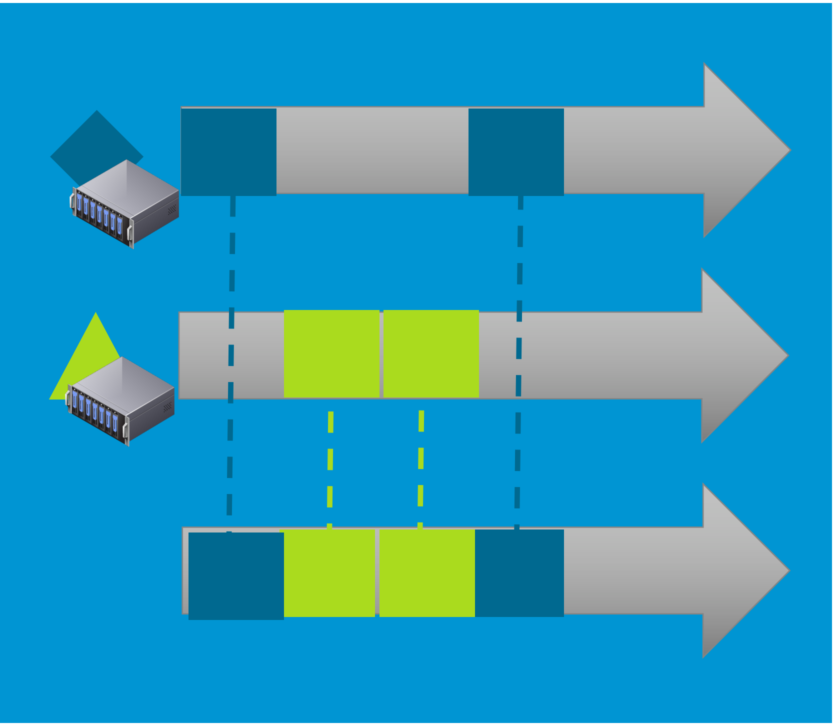
Log replicas and stream replicas only serve committed data

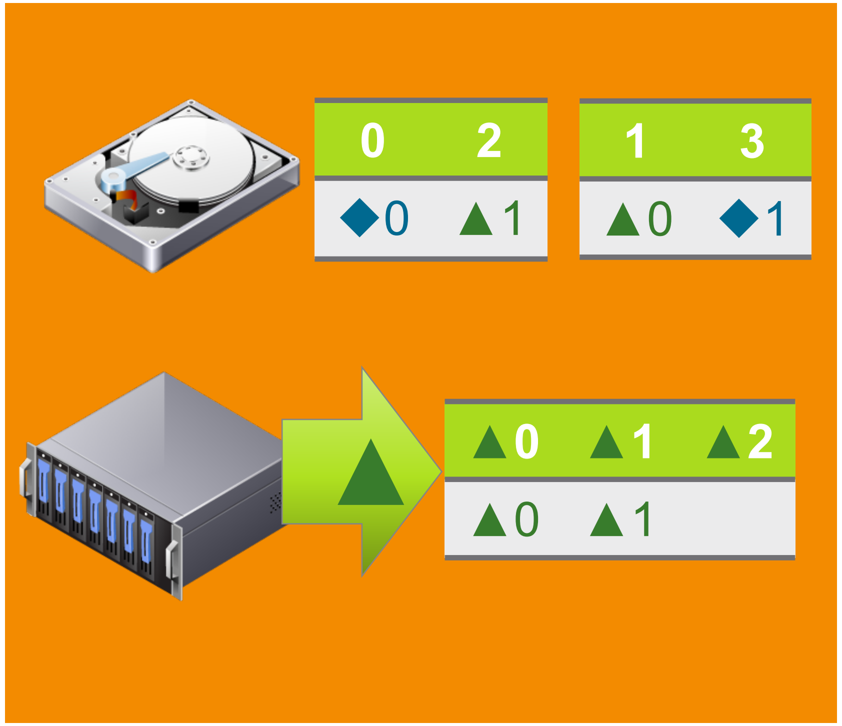


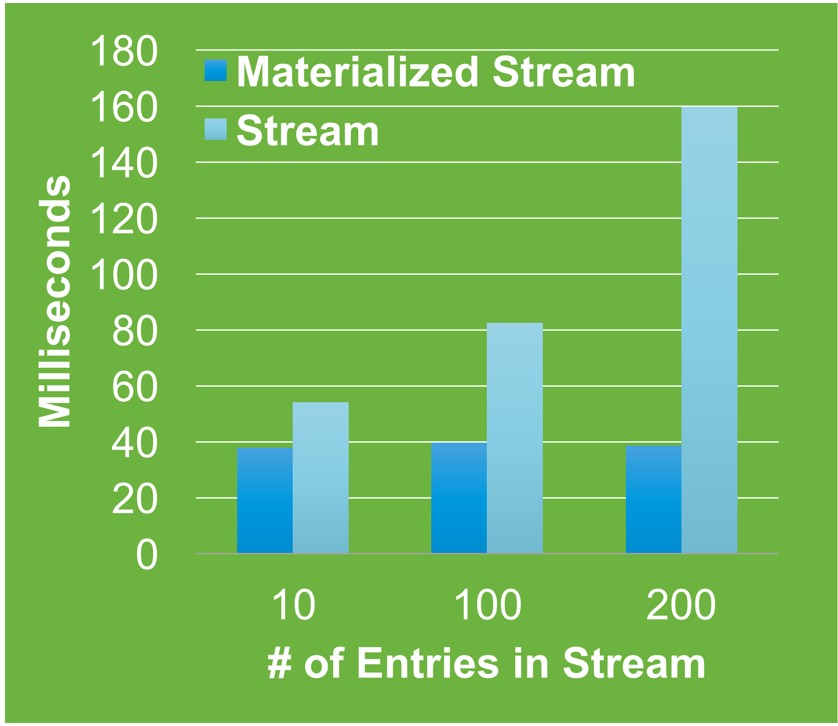
Extra commit message reduces append throughput

vmware®

Reading From a Materialized Stream





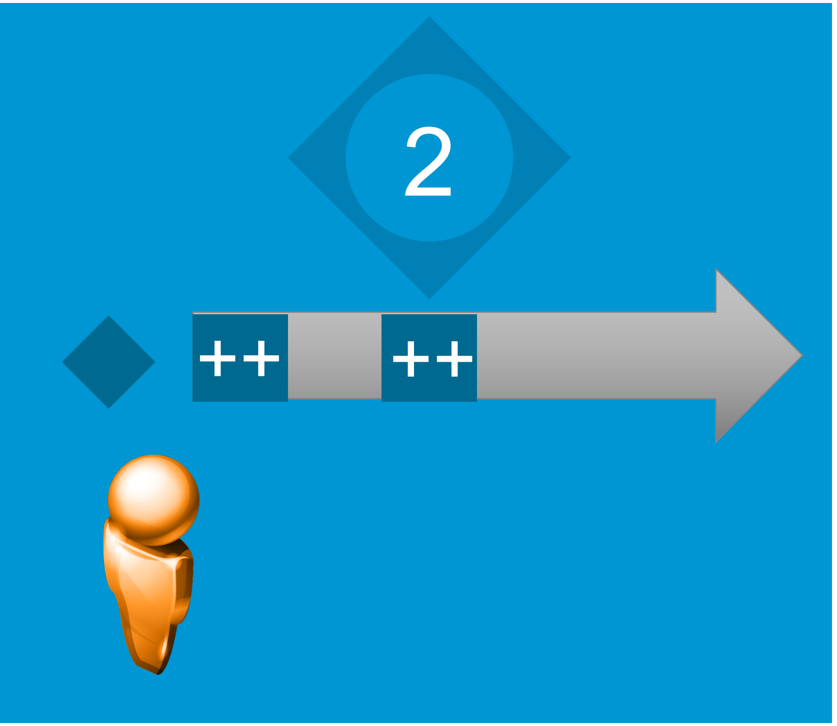


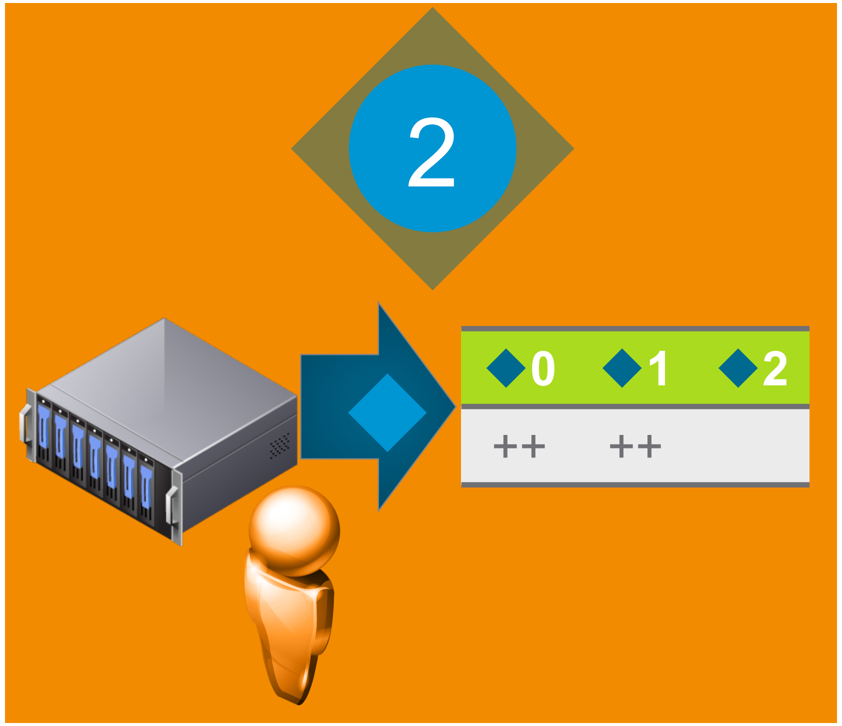
Stream replicas contain all updates for a given stream

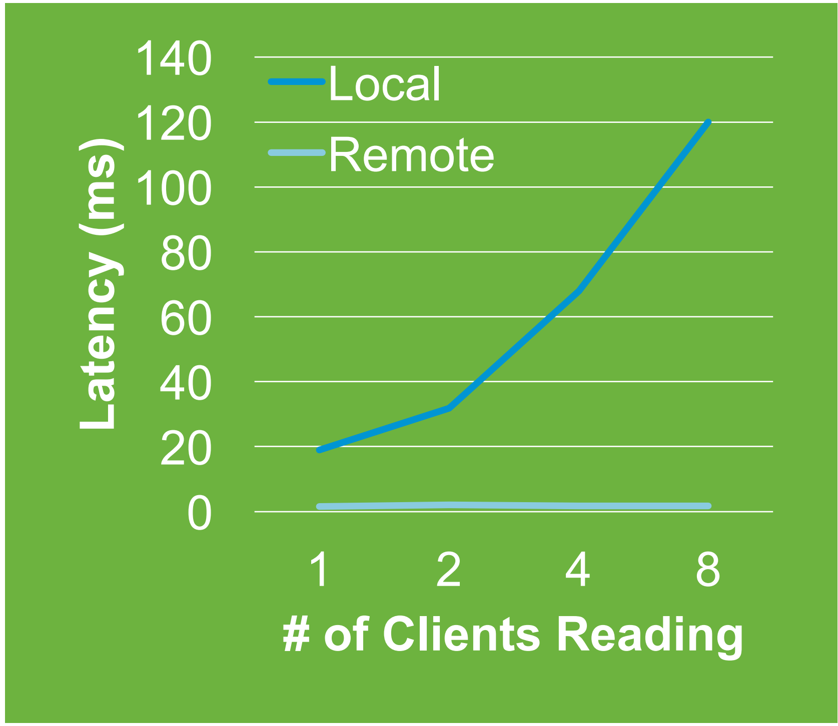
This enables reading a stream by contacting only one replica

Not having to jump from replica to replica greatly improves read performance

Local / Remote Views





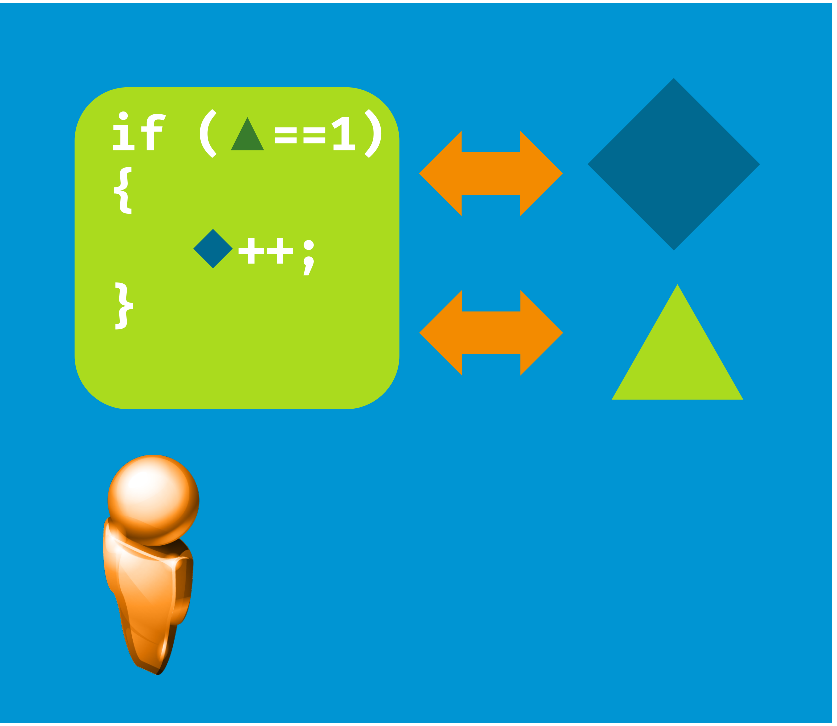


Local views enable clients to obtain in-memory objects by following updates

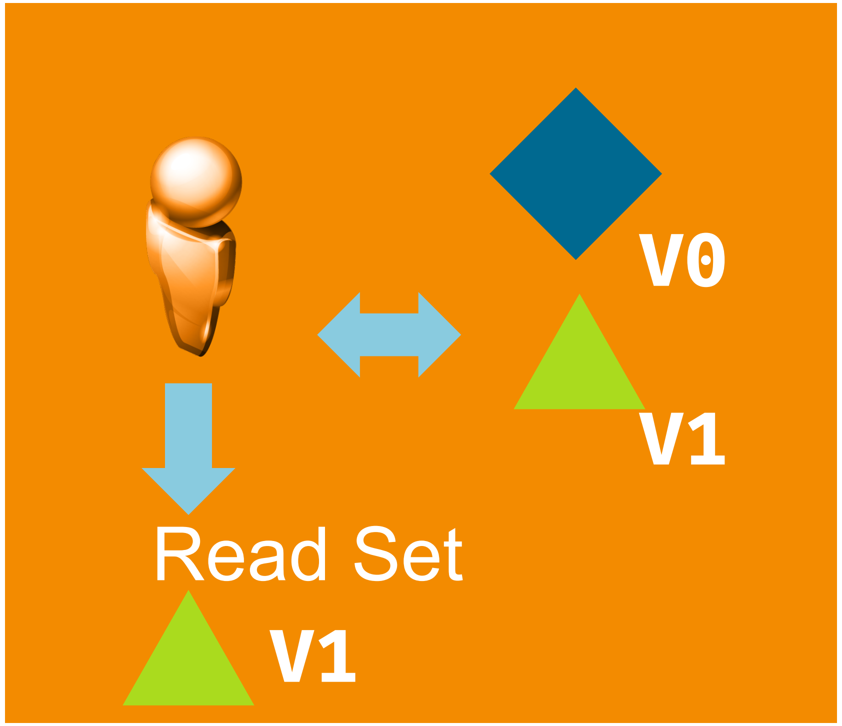
Remote views enable to delegate playback to stream replicas

Remote views keep latency constant with a heavily modified object and many clients reading

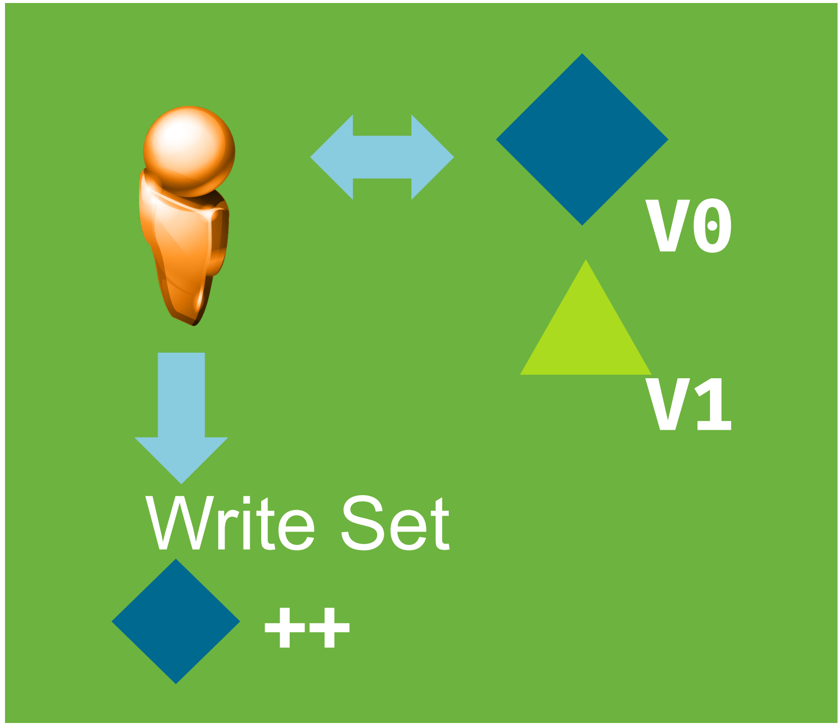
Transactional Execution



We support optimistic transaction execution based on versioned object views

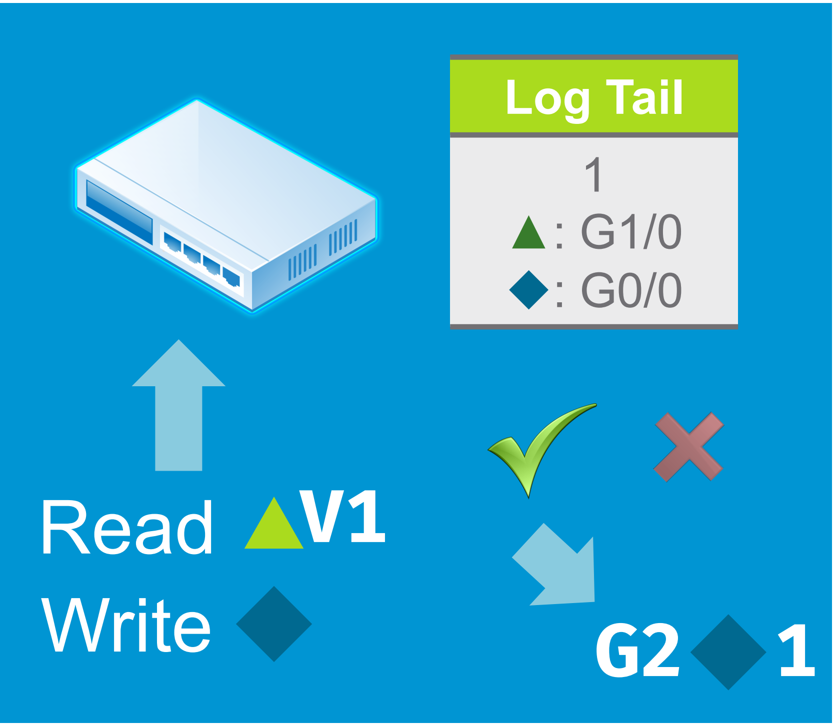


The client tracks the version of each object it accesses

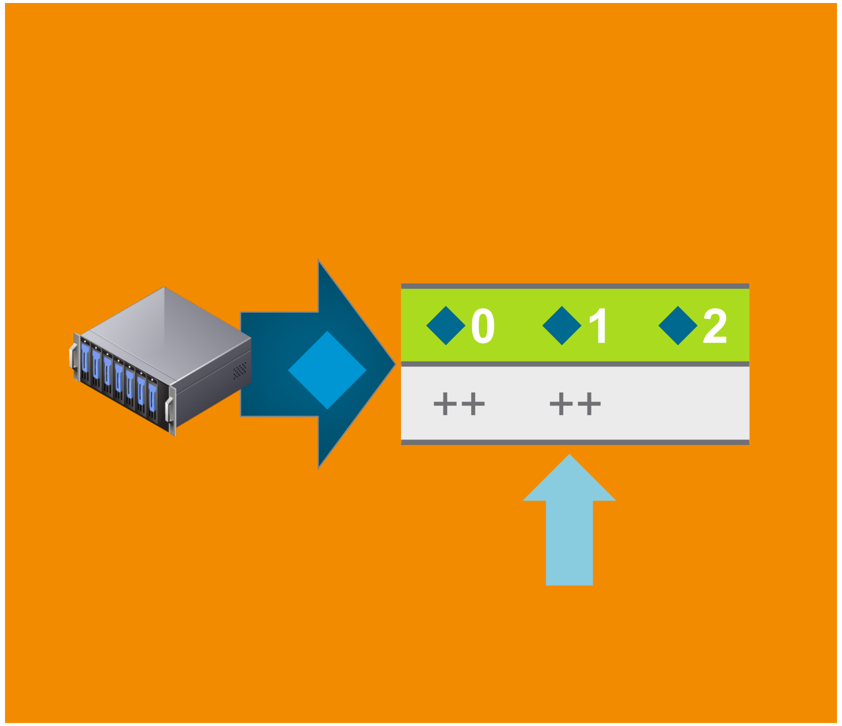


And generates a list of modifications it will make

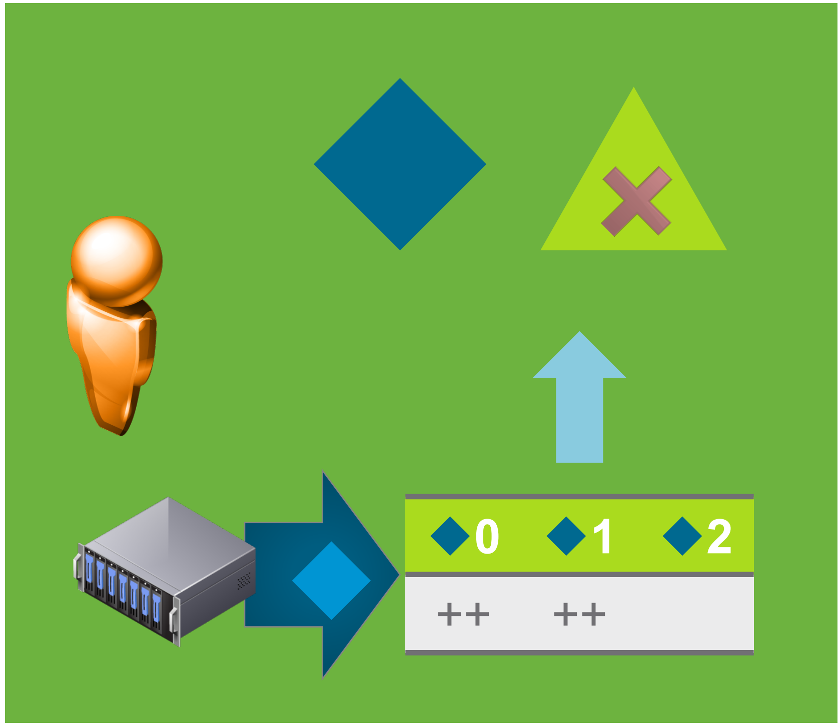
Lightweight Transaction Resolution



Send sequencer version of read and write set, address issued if read set versions are equal

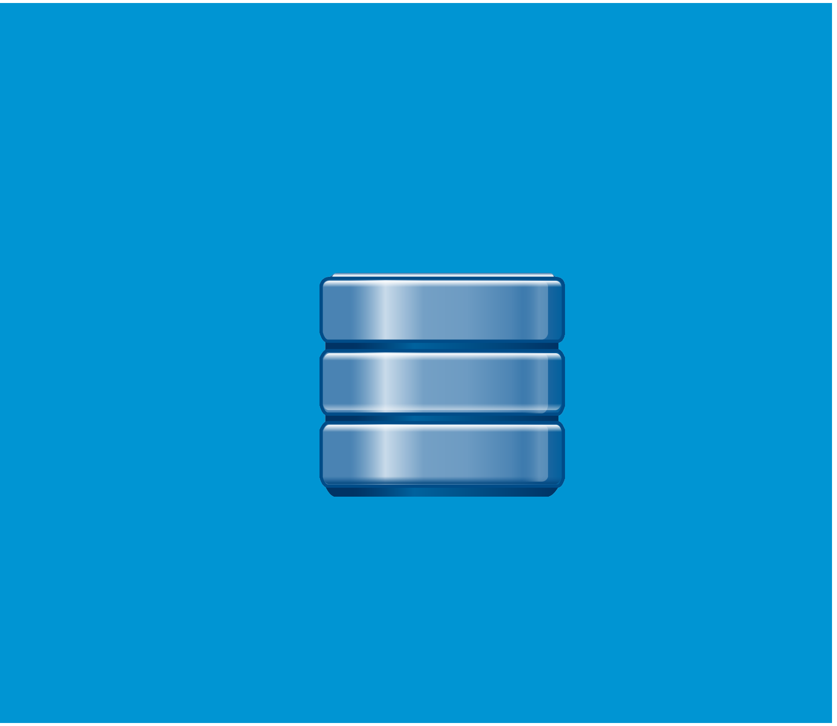


This enables only the write set to be written, since we know that the read set will not have changed

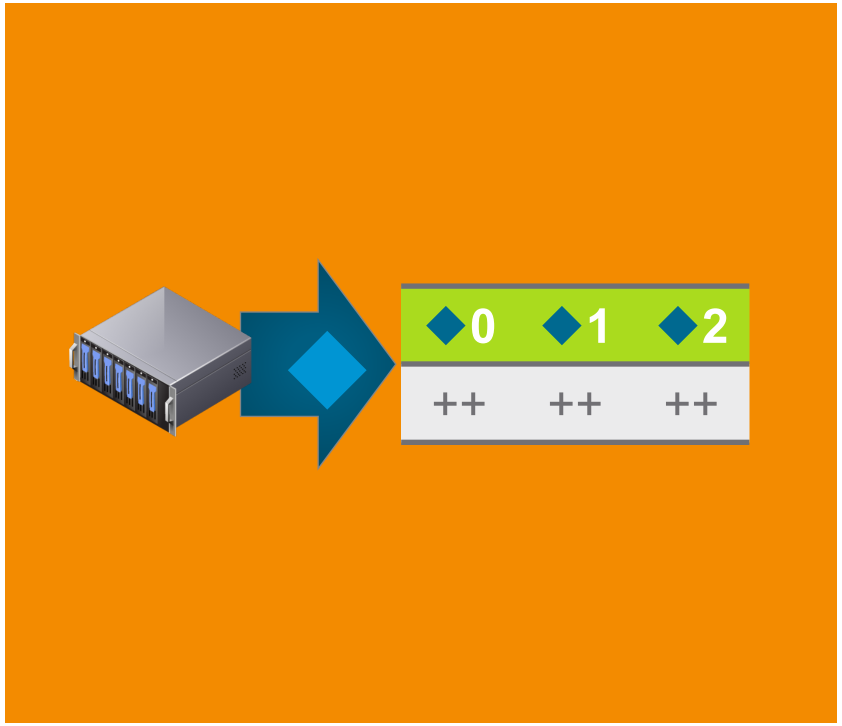


And a client encountering this entry does not need to determine the read set

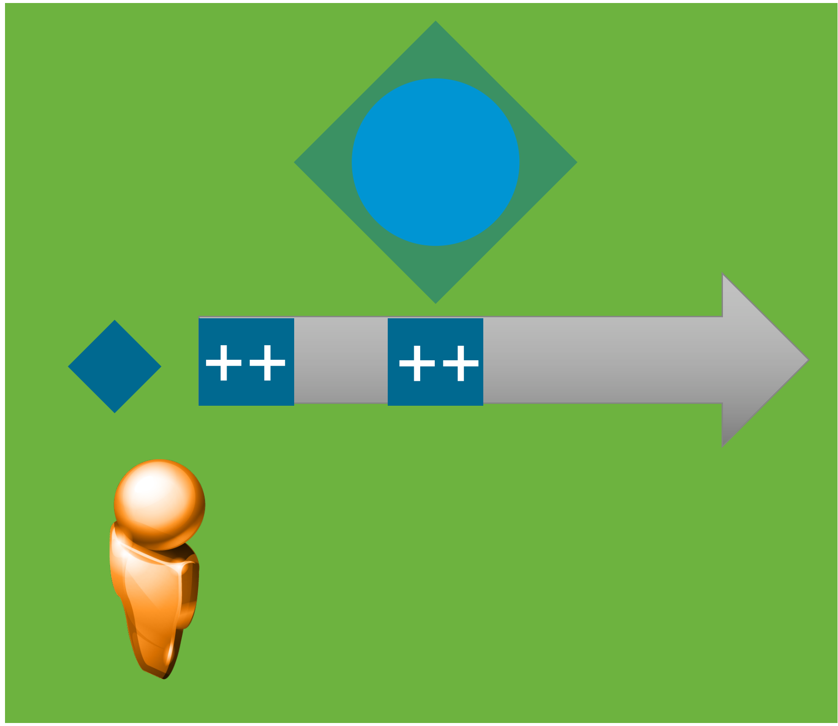
Large State Machines



Objects can contain large amounts of state, which pose a difficulty for SMR

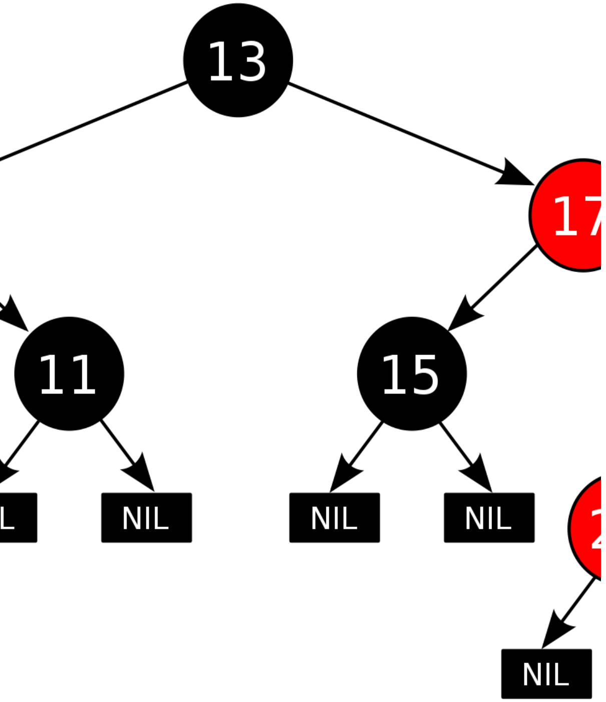


They pose a burden on the log because they contain many updates

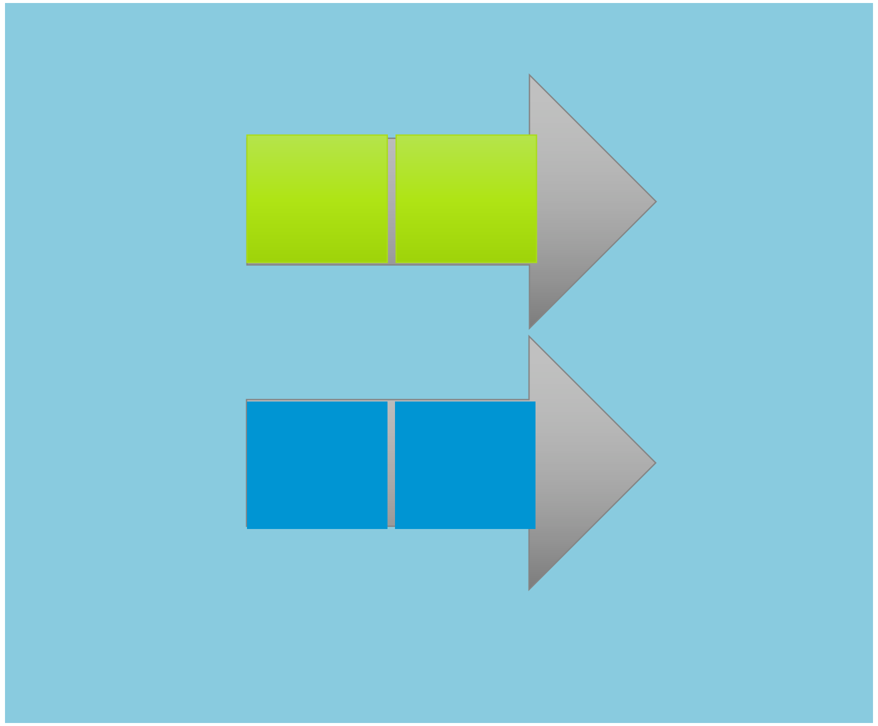


They pose a burden for a client, which has to play all these updates in memory

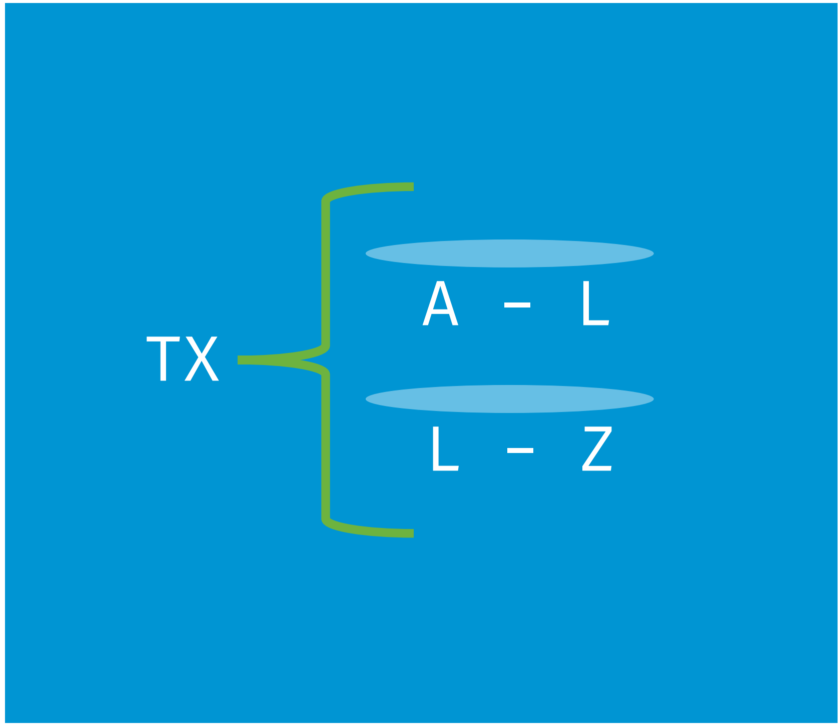
CSMR: Composing vCorfu Objects



vCorfu objects can be composed of other vCorfu objects with a pointer

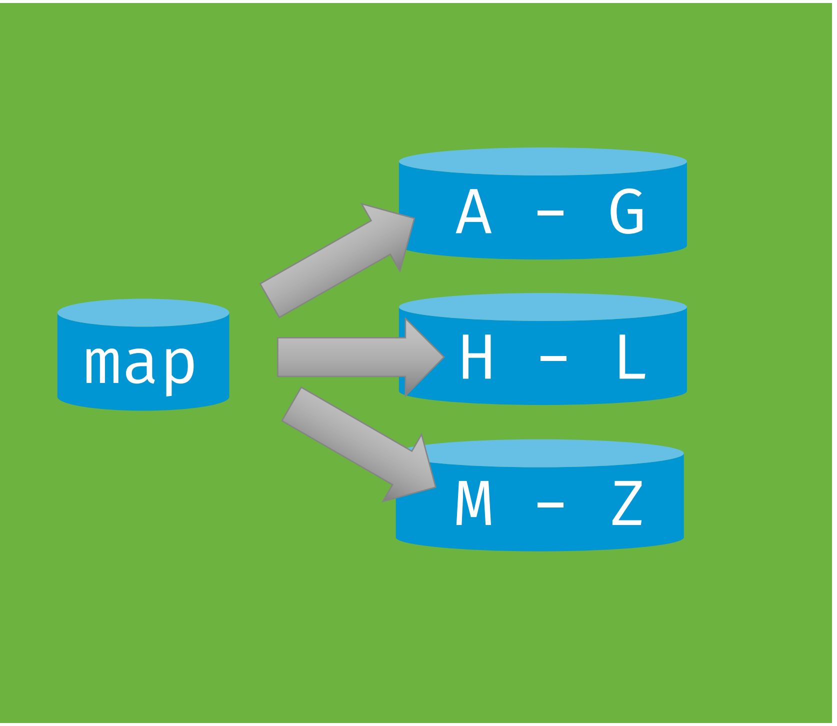


Reduces playback burden by naturally dividing objects

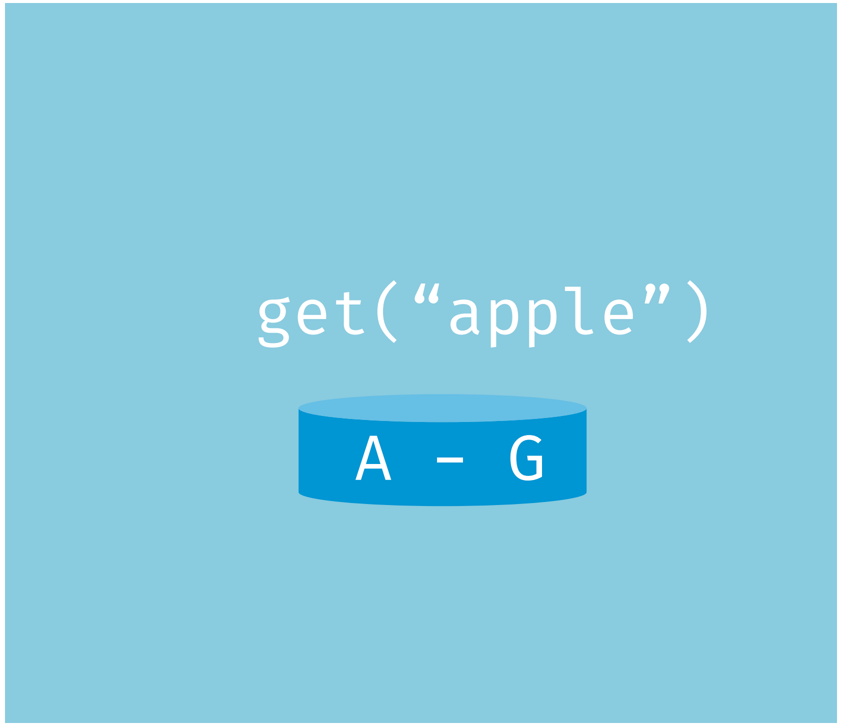


Leverages transactional features of vCorfu

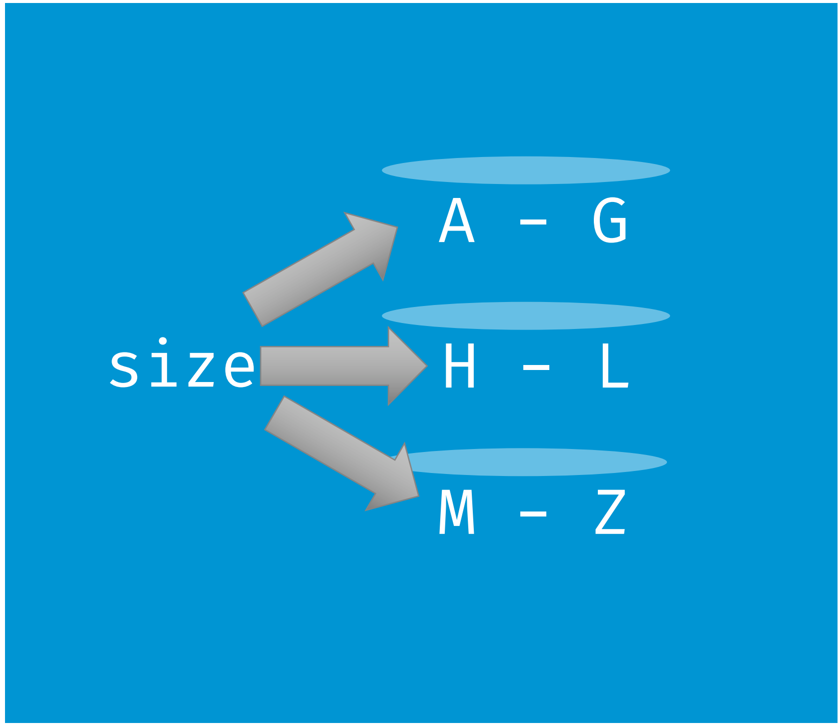
CSMR Example: Map



Instead of a single map, compose a map from multiple buckets

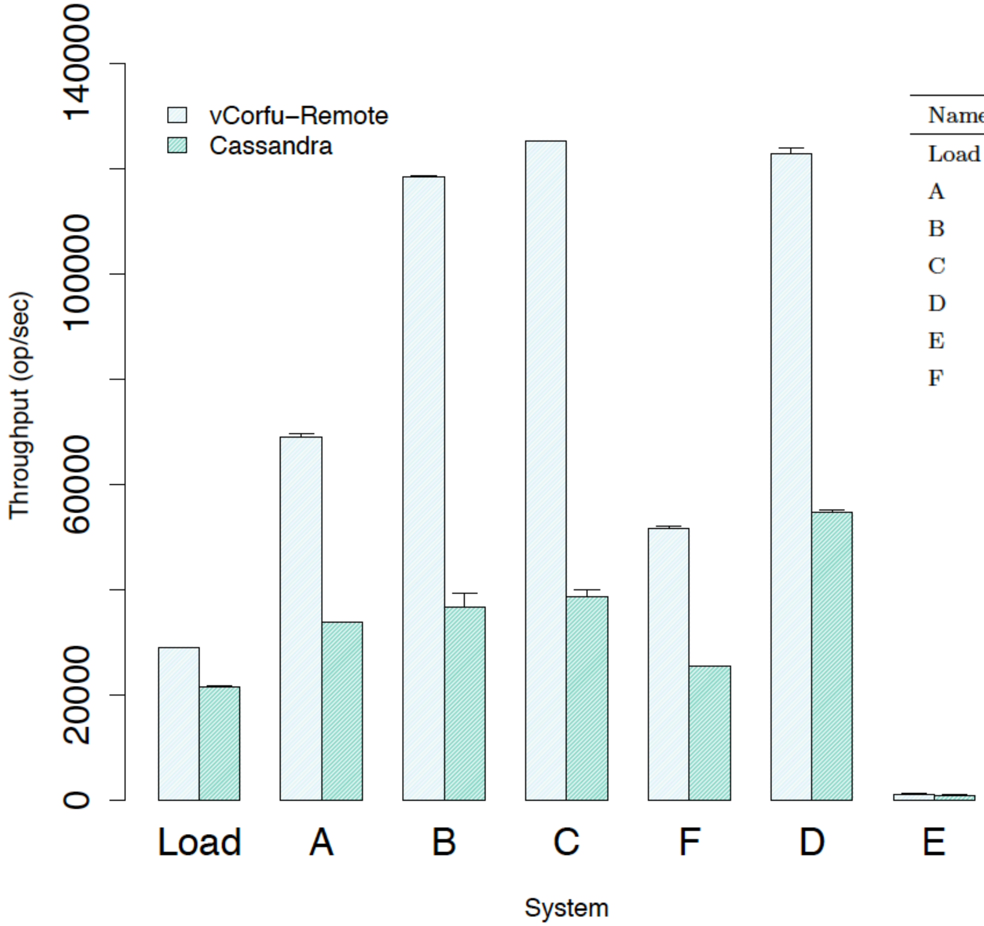


Most operations only need to access a single bucket



Certain operations, like clear() or size() are more expensive with CSMR

vCorfu vs. Cassandra YCSB

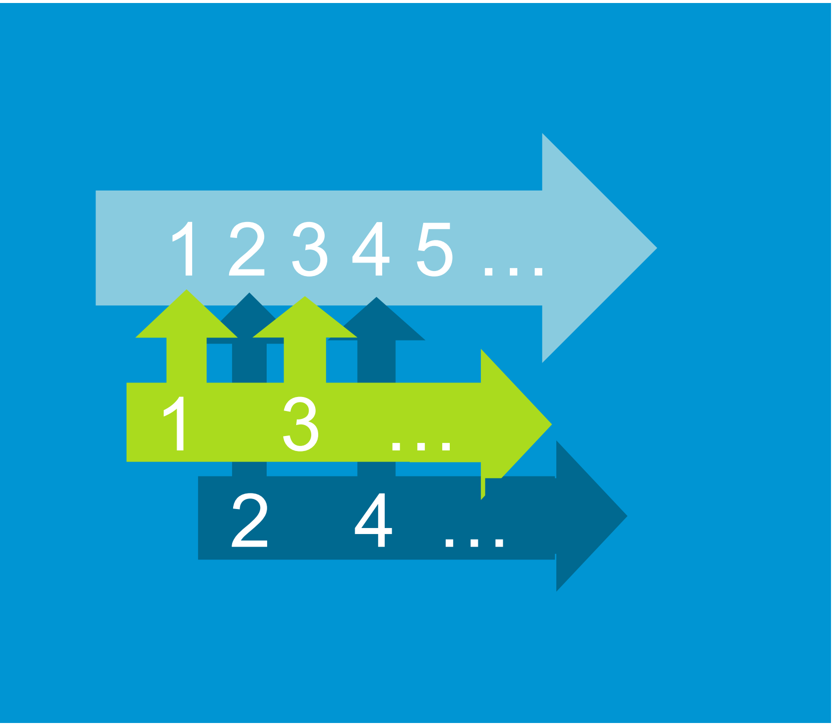


Name	Workload	Description
Load	100% Insert	Propogating Database
A	50% Read/50% Update	Session Store
B	95% Read/5% Update	Photo Tagging
C	100% Read	User Profile Cache
D	95% Read/5% Insert	User Status Updates
E	95% Scan/5% Insert	Threaded Conversations
F	50% Read/50% Read-Modify	User Database

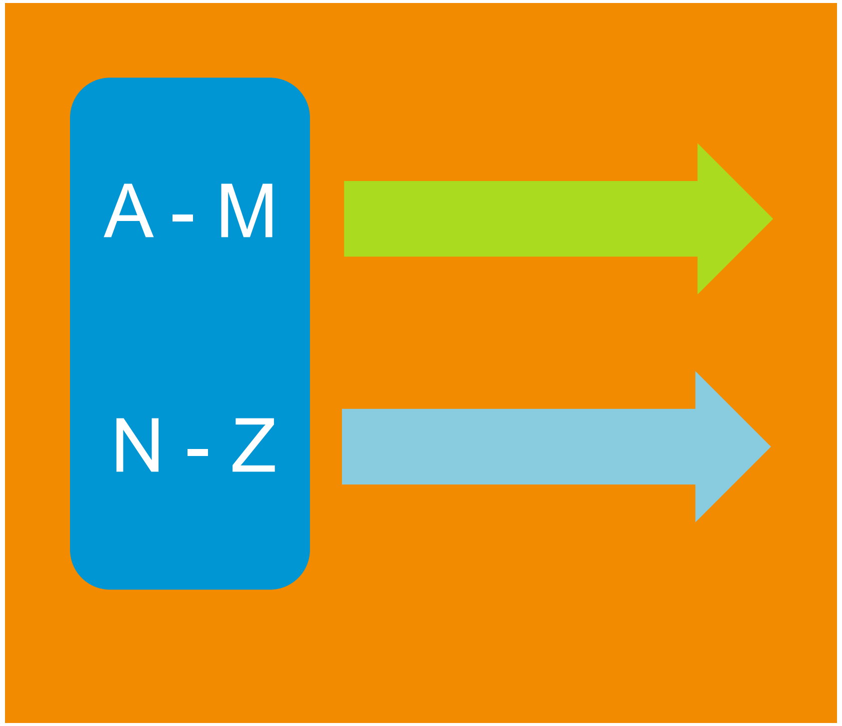
Conclusion

vCorfu Benefits

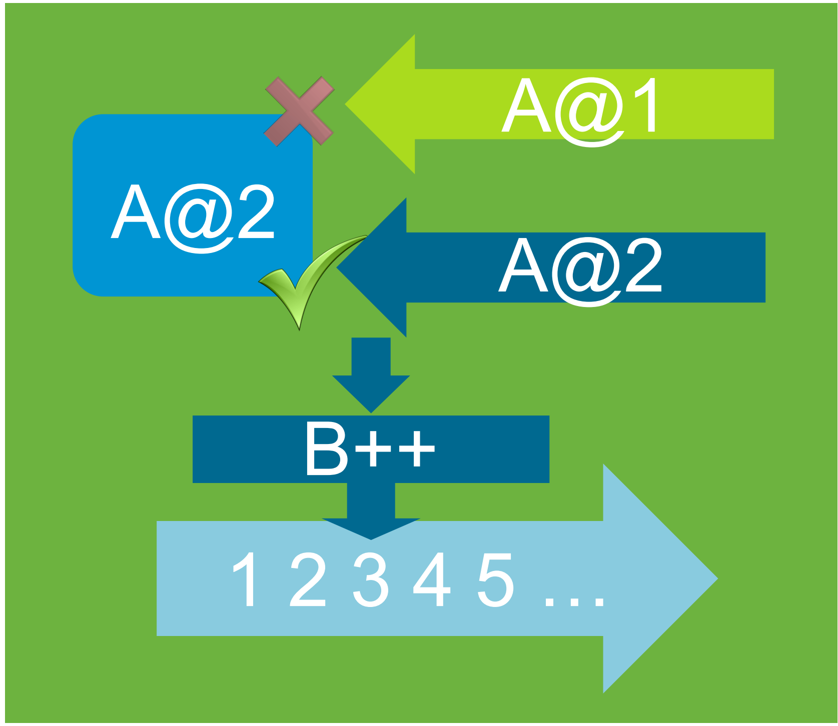
vCorfu addresses the read burden by…



Stream materialization, which localizes related updates and enables reads without playback



Composable SMR, which enables large state machines without forcing clients to replicate the entire state machine



Lightweight Transaction Resolution, which eliminates the need for clients to determine whether transactions in a log were aborted

vmware

Special thanks to the Corfu Team: Past and Present

Original Corfu Paper
Mahesh Balakrishnan
Dahlia Malkhi
Vijayan Prabhakaran
Ted Wobber
John D. Davis

Tango
Ming Wu
Sriram Rao
Tao Zou
Aviad Zuck

Replex
Amy Tai
Michael J. Freedman
Ittai Abraham

VMware NSX
Maithem Munshed
Zeeshan Lokhandwala
Medhavi Dhawan
Jim Stabile
Kapil Goyal
Guprit Johal
Konstain Spriov
James Chang
Jim Yang
Kevin James
Anny Manzanil
Ragnar Edholm

vCorfu
Christopher J. Rossbach
Udi Wieder
Scott Fritchie

Corfu is Available on GitHub

CorfuDB / **CorfuDB**

Unwatch 36 | Unstar 161 | Fork 38

Code | Issues 43 | Pull requests 6 | Projects 1 | Wiki | Pulse | Graphs | Settings

A cluster consistency platform

Edit

New Add topics

1,602 commits | 13 branches | 0 releases | 18 contributors

gitter join chat

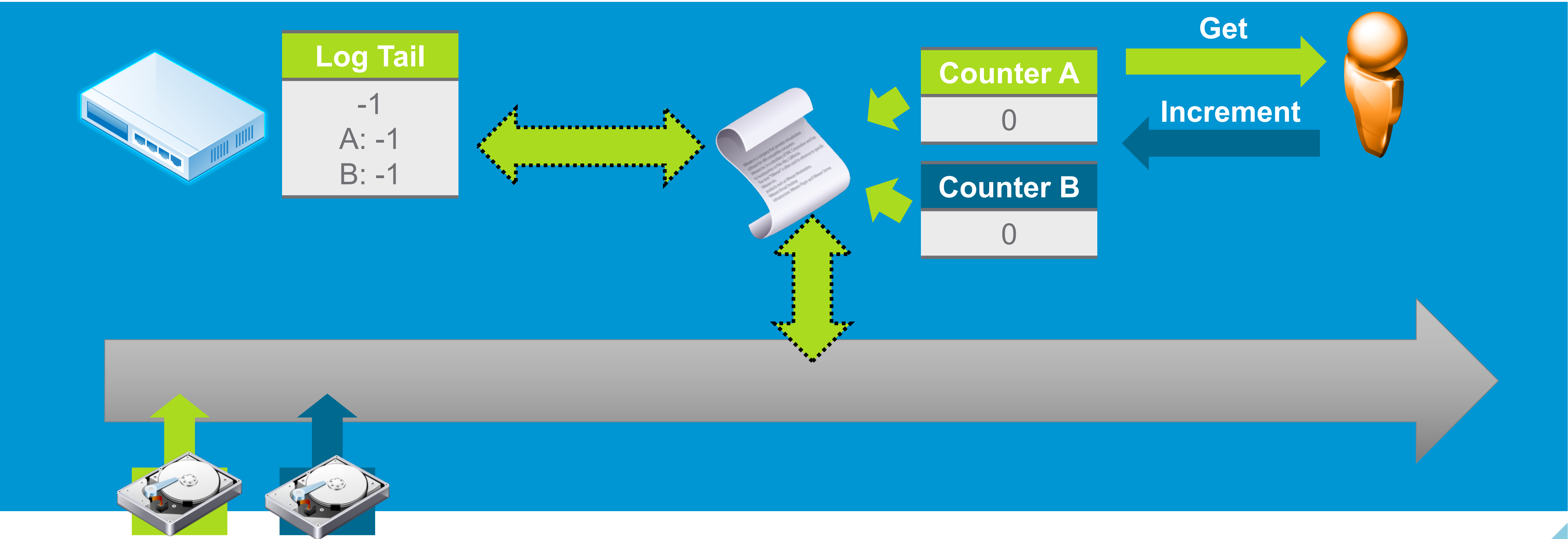
build passing | coverage 55% | Ready 2

Corfu is a consistency platform designed around the abstraction of a shared log. CorfuDB objects are in-memory, highly available data structures providing linearizable read/write operations and strictly serializable transactions. CorfuDB is based on peer-reviewed research, see References.

fin

Questions?

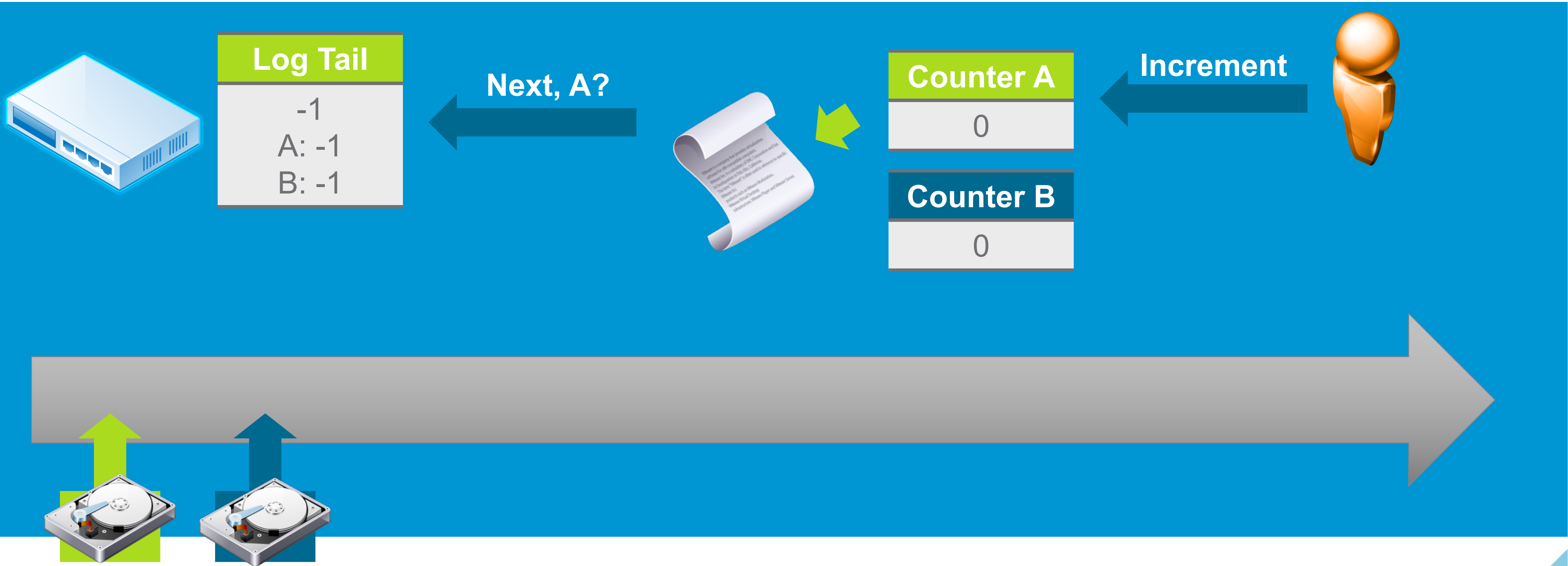
We Take the Tango [1] Approach…



Where clients interact with objects, and a runtime manages interactions with the log. Each object is contained within a stream, which is the set of updates for that object.

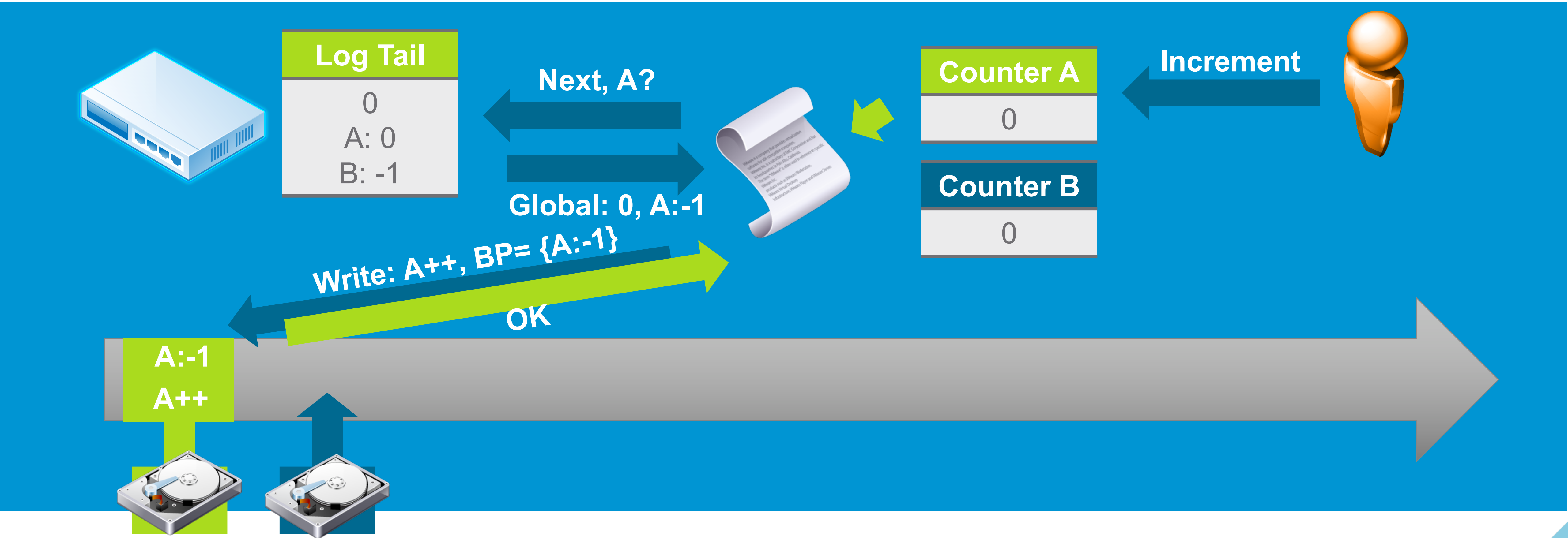
[1] **Tango: Distributed Data Structures over a Shared Log.** Mahesh Balakrishnan, Dahlia Malkhi, Ted Wobber, Ming Wu, Vijayan Prabhakaran, Michael Wei, John D. Davis, Sriram Rao, Tao Zou, Aviad Zuck. *SOSP 2013: The 24th ACM Symposium on Operating Systems Principles.*

Example: Incrementing a Counter



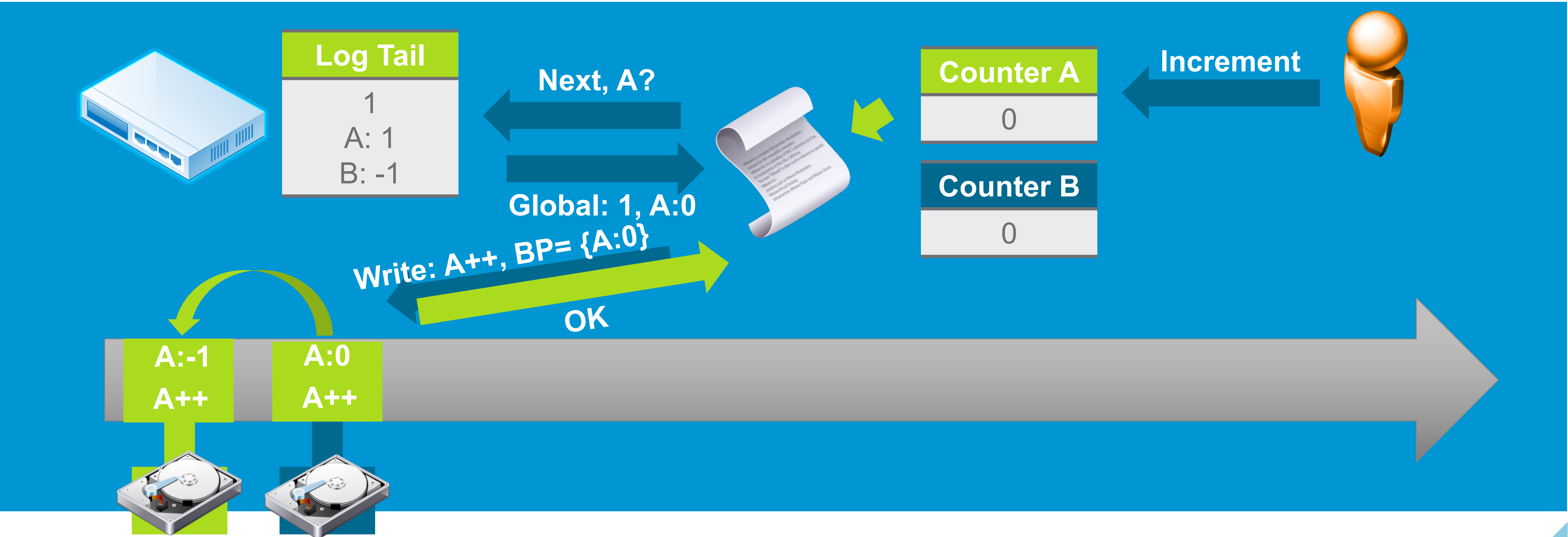
When the client increments the counter, the runtime asks the sequencer for the next address for the stream of the given counter

Example: Incrementing a Counter



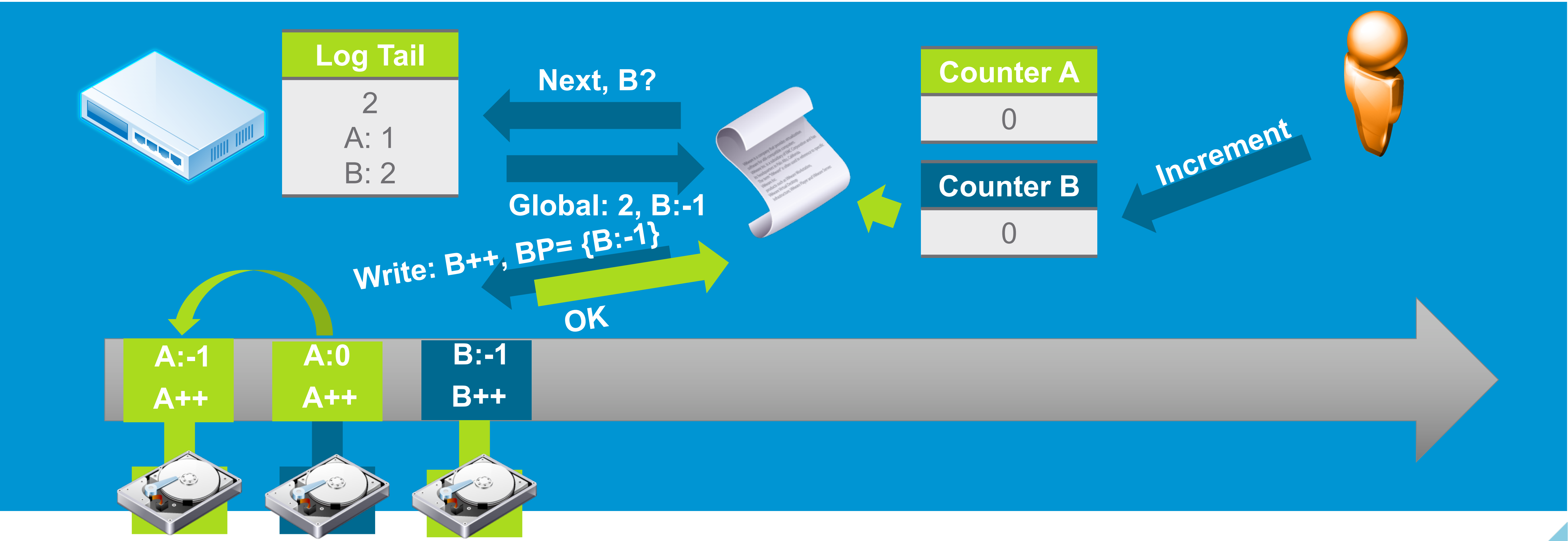
And the runtime can now write a increment record to the log replica, writing the previous stream address given in the record, known as a ***backpointer***.

Example: Incrementing a Counter



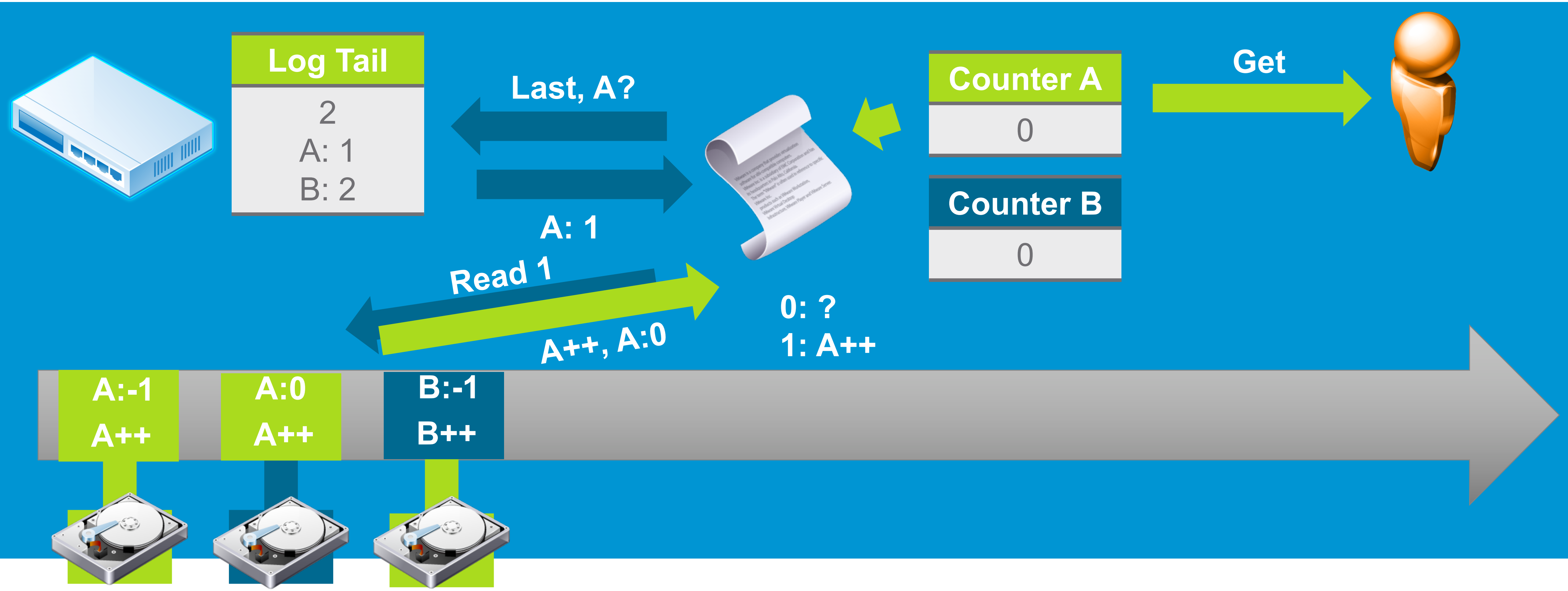
And the runtime can now write a increment record to the log replica, writing the previous stream address given in the record, known as a ***backpointer***.

Example: Incrementing a Counter



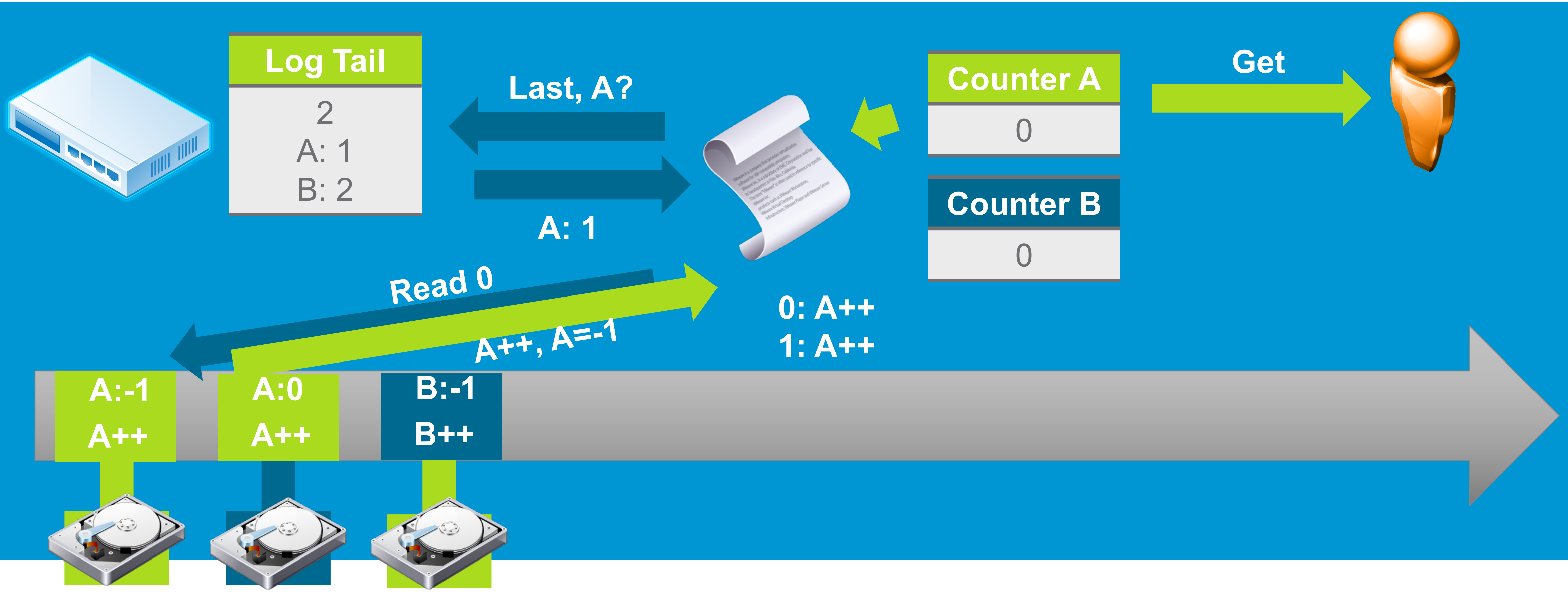
And the runtime can now write a increment record to the log replica, writing the previous stream address given in the record, known as a ***backpointer***.

Example: Reading a Counter



To read, the runtime contacts the sequencer for the latest address issued to stream A.
The client then reads all the updates, traversing the backpointers.

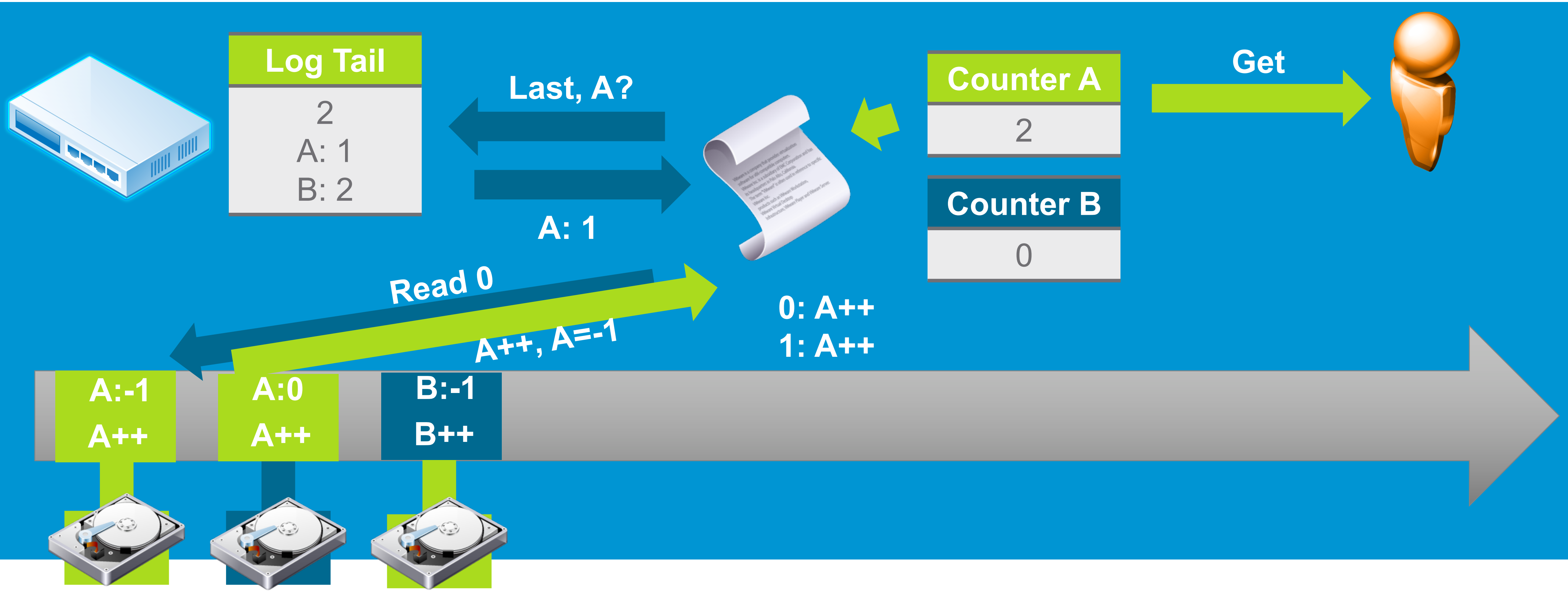
Example: Reading a Counter



The runtime keeps all the updates in memory until the entire stream has been read.

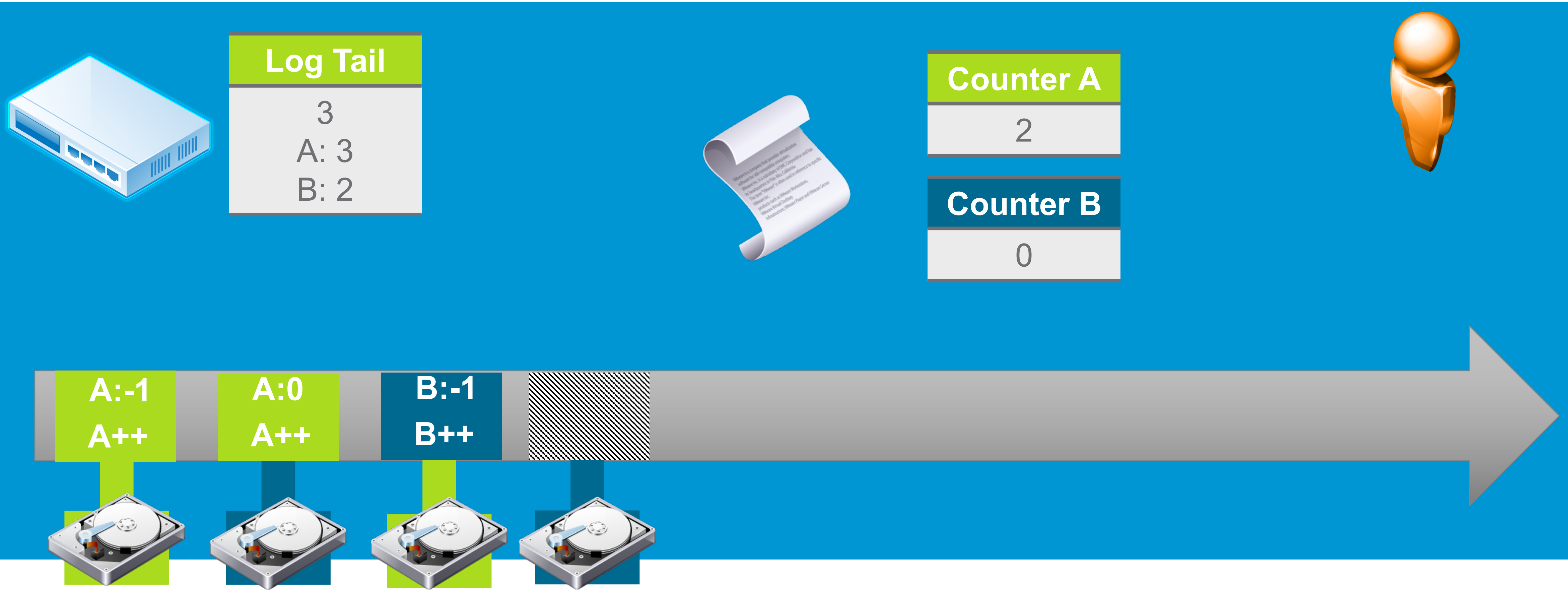
vmware

Example: Reading a Counter



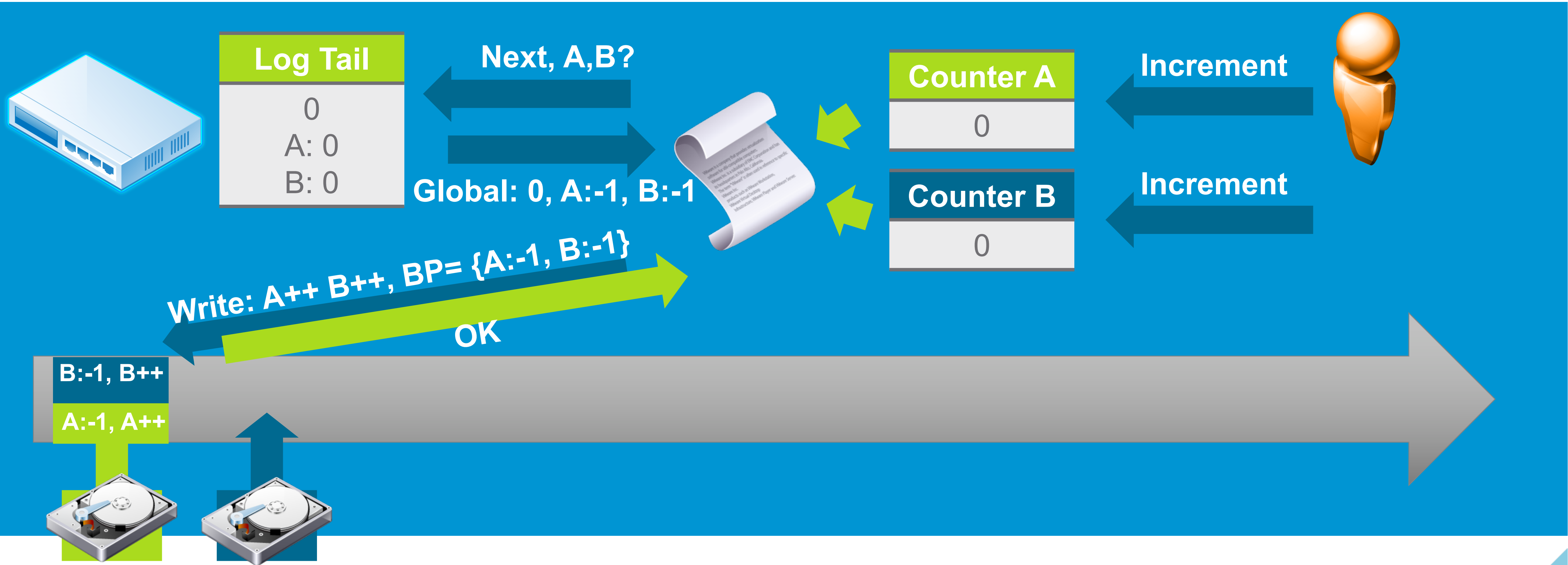
Once the entire stream is read, the runtime applies the updates and returns the new value of the counter to the client.

Example: Holes



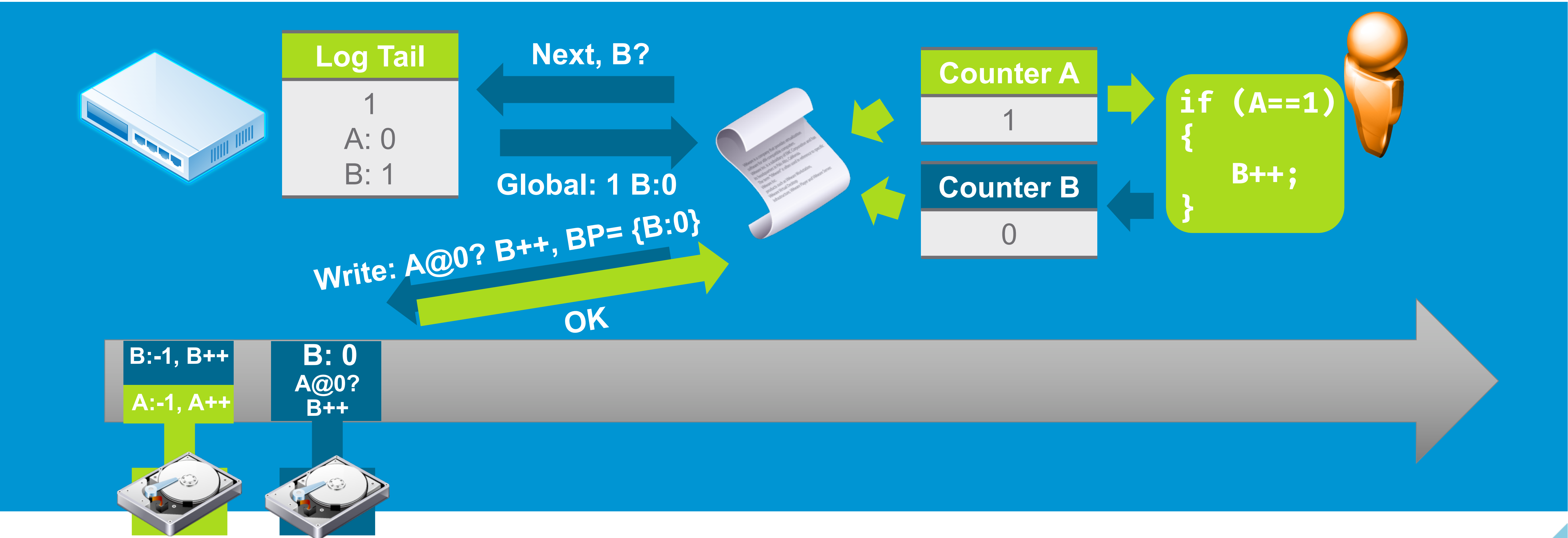
Holes due to failed clients can be a problem – they contain no information about backpointers, and require a linear scan if encountered.

Example: Incrementing Multiple Counters



A multi-put is implemented by generating a single entry which is part of both streams.

Example: Incrementing Multiple Counters... Conditionally



To increment multiple counters conditionally, a transaction is created.
The runtime keeps track of the read set (address or version of read objects) and the write set.
At commit time, an entry with the read set and write set is written.

vCorfu Stream Store

Materializing Streams

Introducing a New Component: The Stream Replica

0	1	2
X	Y	?

Log Replica

A

A0	A1	A2
X	Y	?

B

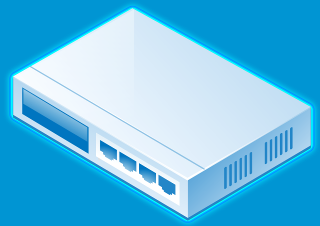
B0	B1	B2
Z	?	?

Stream Replica

In vCorfu, we add an additional component, a stream replica, which stores data indexed not on the log address, but a combination of the stream ID and the address in the stream.

vmware®

Modifying an Existing Component: Sequencer now tracks Stream Addresses



Log Tail

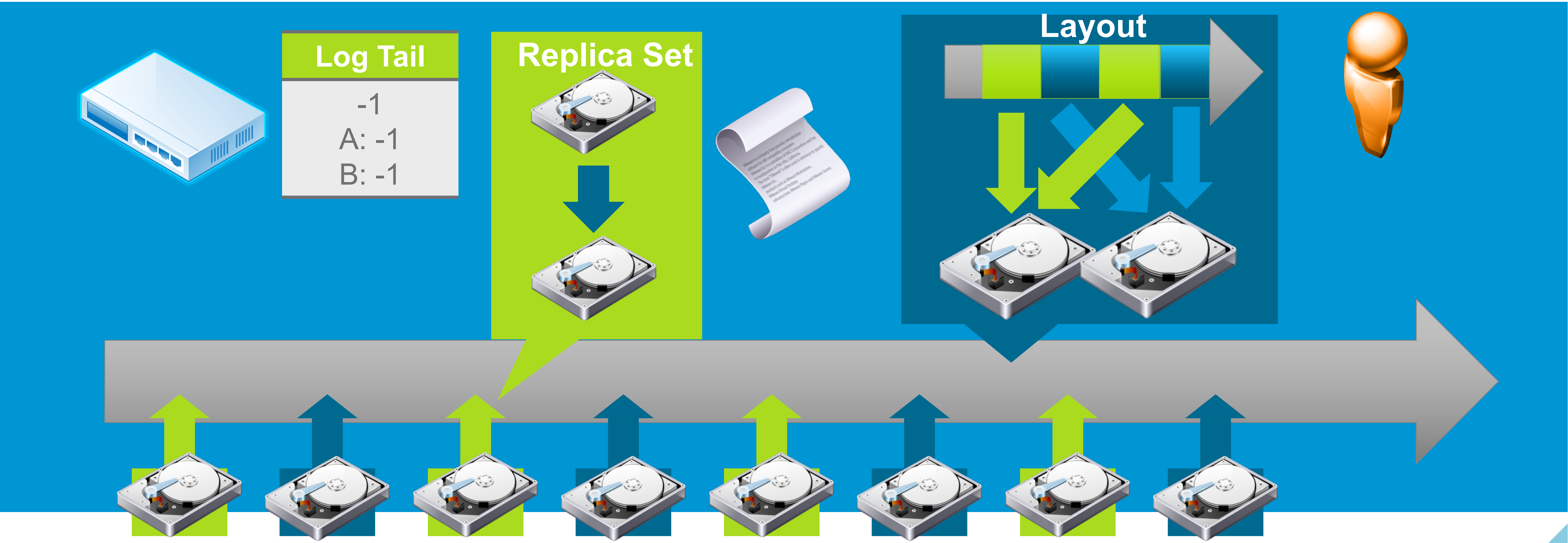
2
A: G0/0
B: G2/2

Gx (Global Address) /y (Stream Address)

Streaming Sequencer

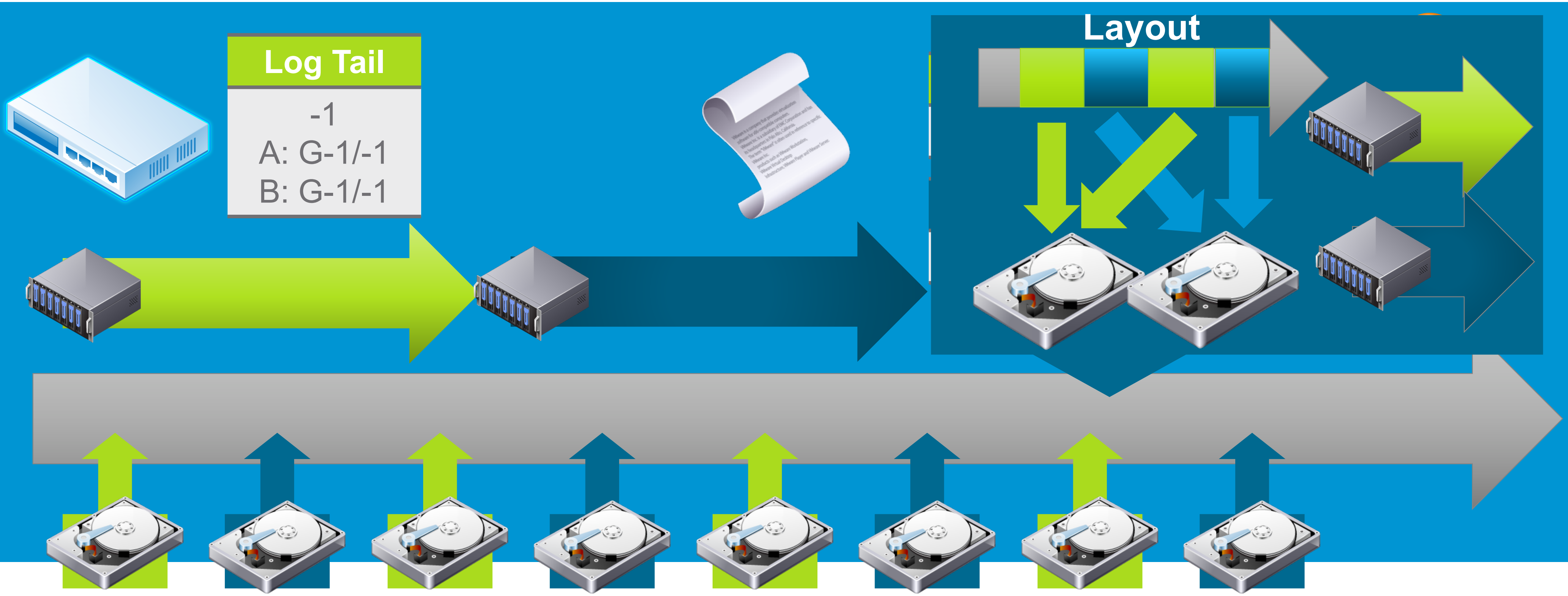
We also make a modification to the sequencer so it tracks the stream addresses used as an index for the stream replicas. This is a small counter with a small amount of state.

Corfu / Tango Replica Sets



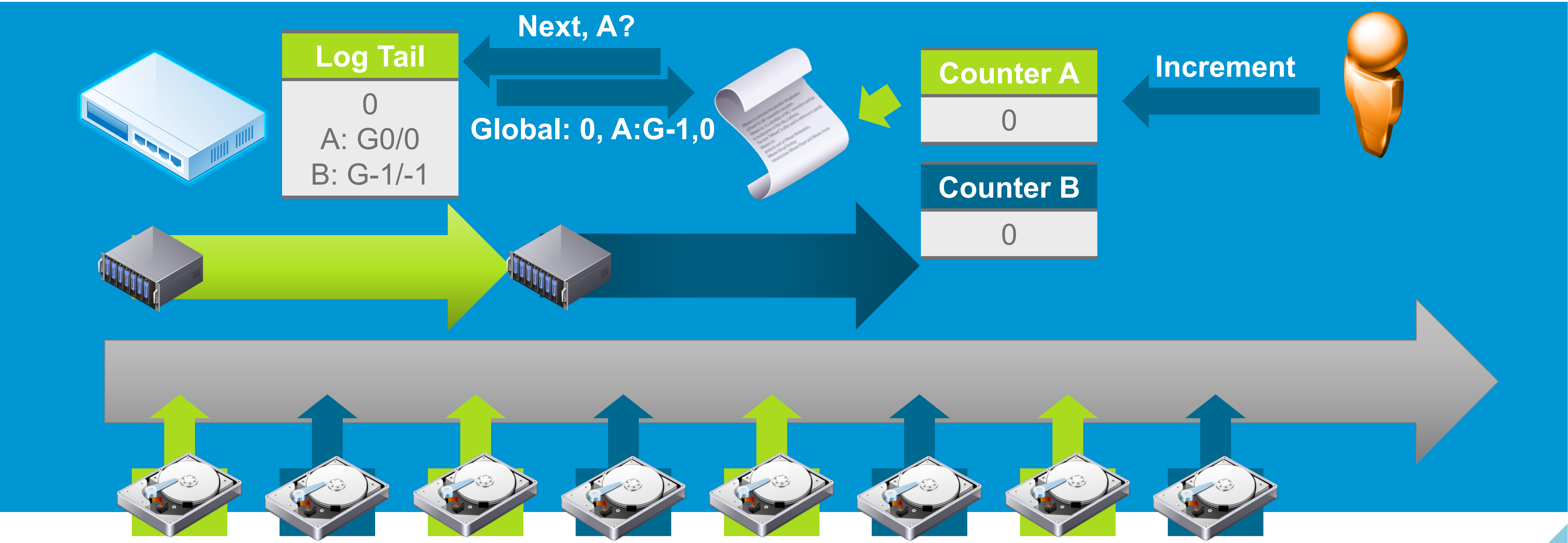
In Corfu/Tango, the log is striped across replica sets, as described by the layout, and each replica set is replicated via chain replication.

Materialized Streams



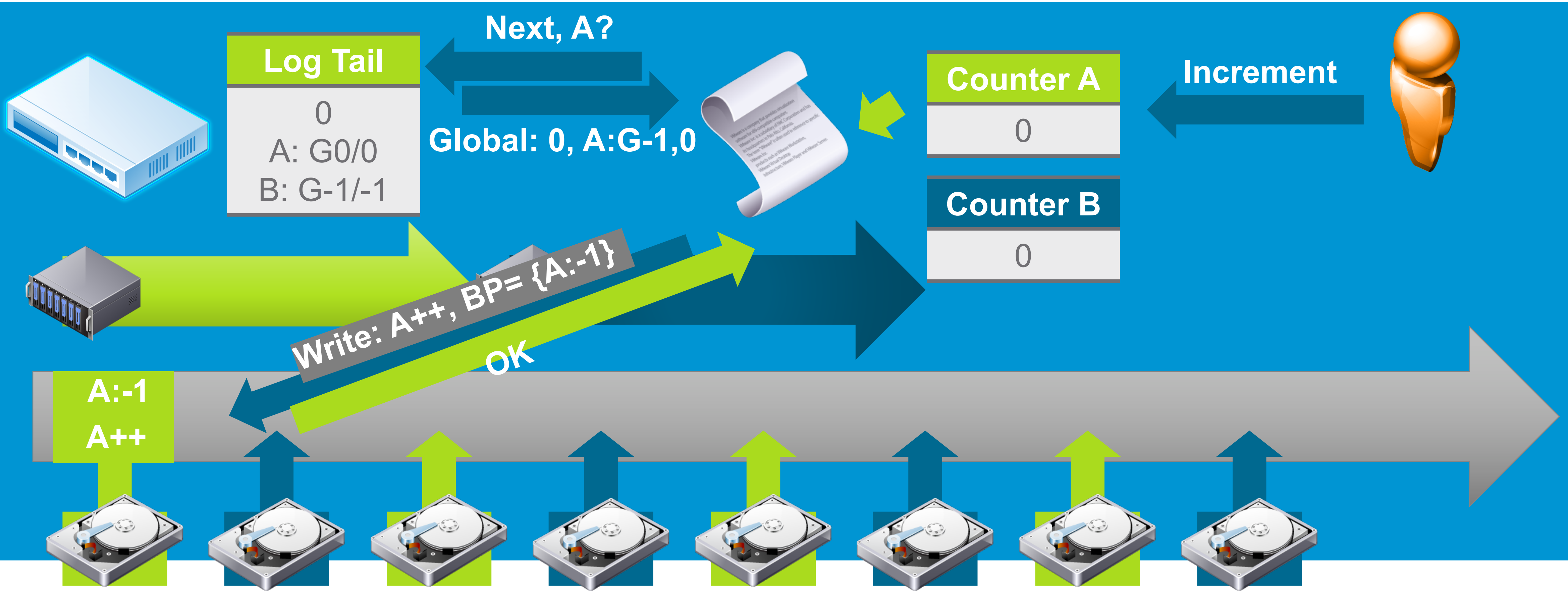
In vCorfu, the layout also maps each stream to a stream replica, which serve materialized views of each stream.

Example: Incrementing a Counter



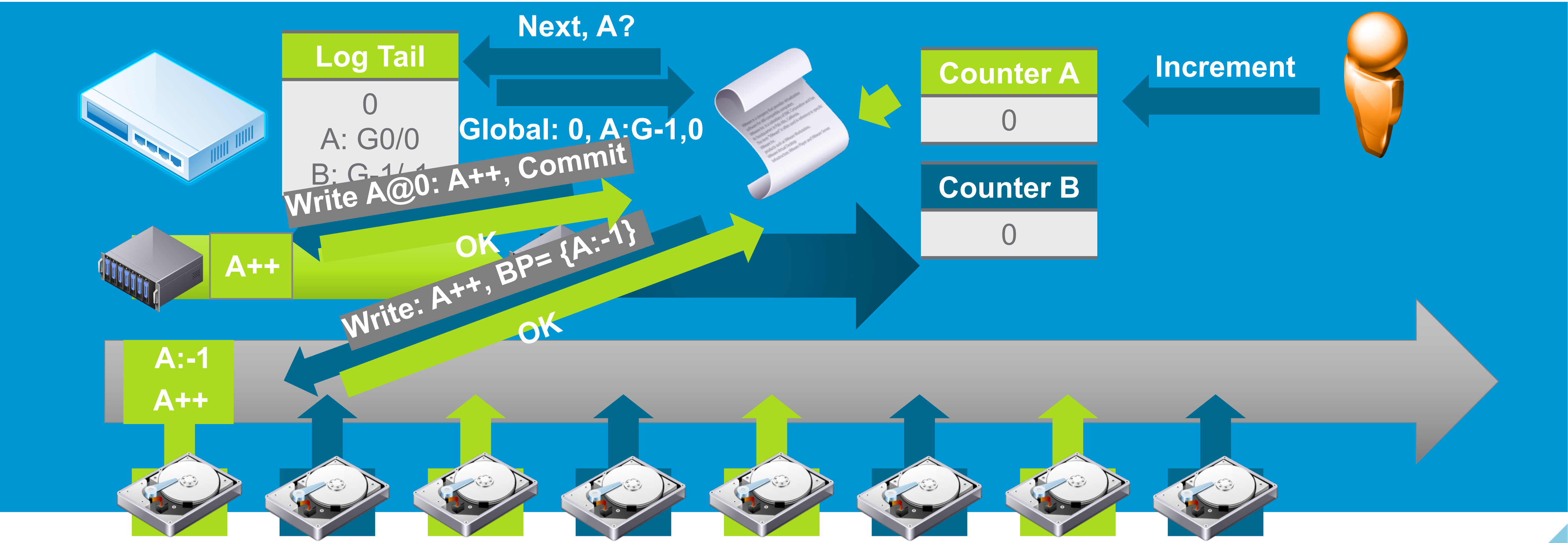
To append to stream A, we now obtain a global address, backpointer and stream address from the sequencer

Example: Incrementing a Counter



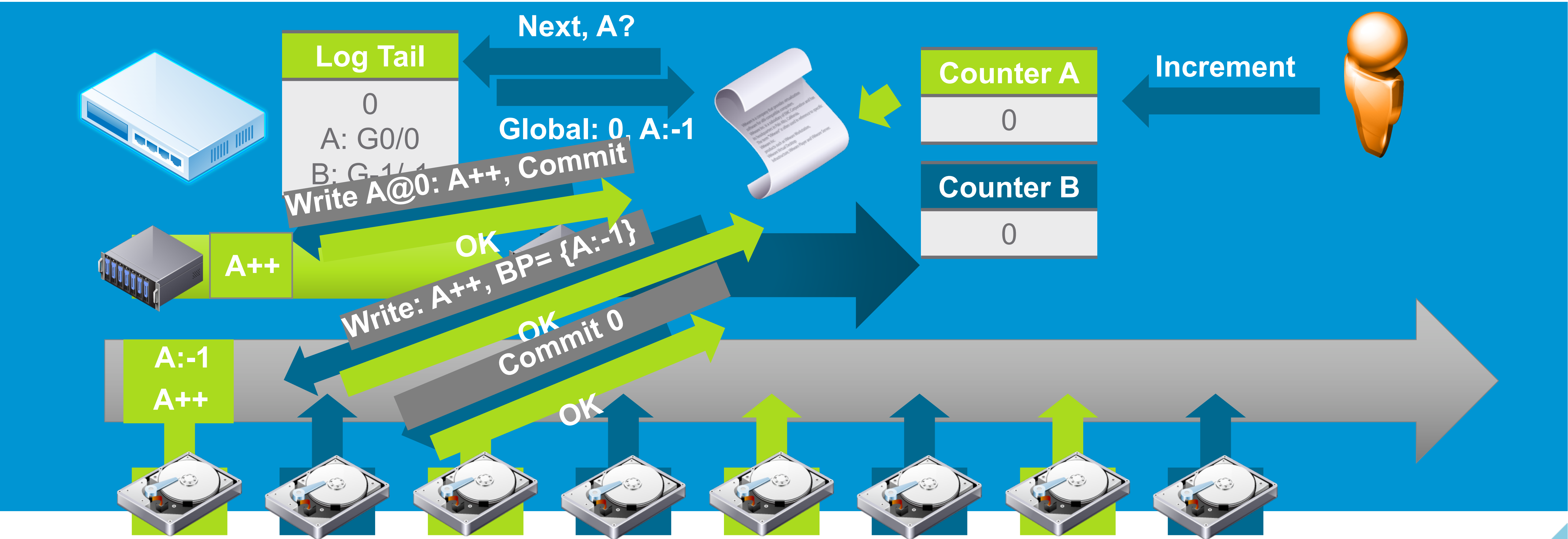
Using the global (log) address, we write to the log replica

Example: Incrementing a Counter



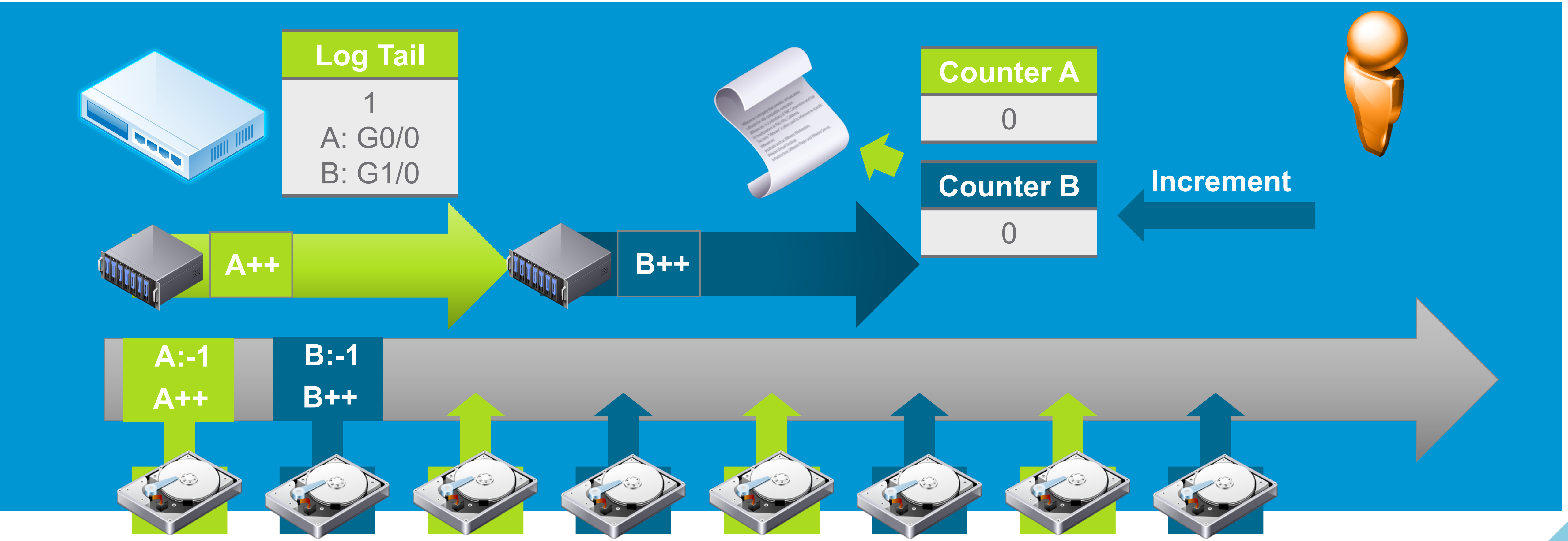
Then using the stream address, we write to the stream replica.
Since this is the last write we will perform, we also indicate that it is okay to commit this write.

Example: Incrementing a Counter



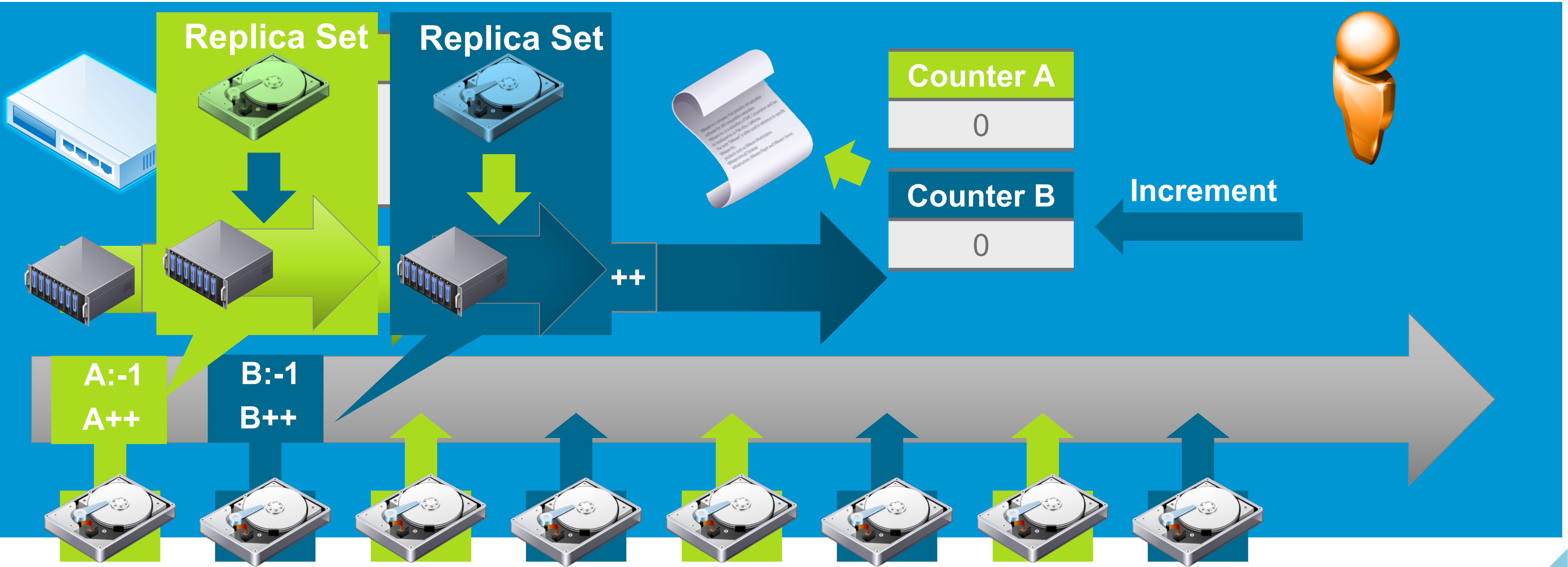
We then broadcast commit to any replicas we have written to. Replicas only serve committed data.

Example: Incrementing a Counter



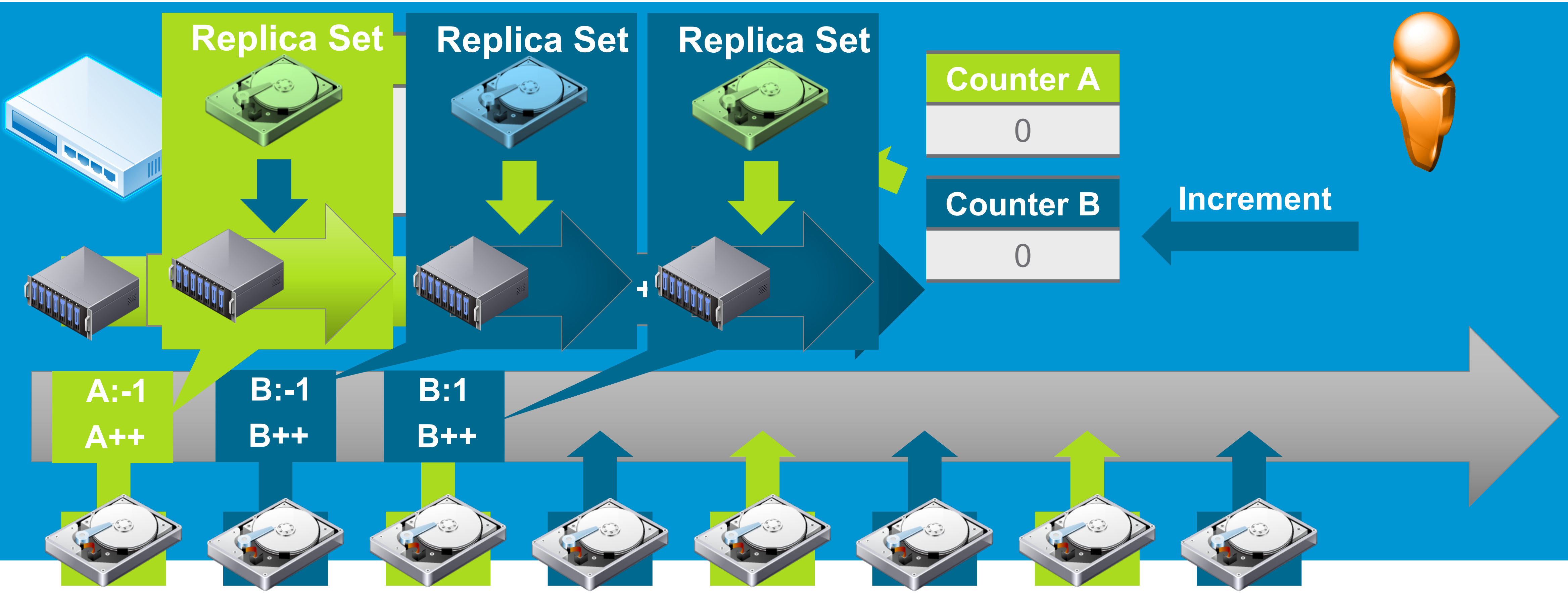
As a result, each stream replica holds only the updates for each stream, which we refer to as a materialized stream when a stream replica is available.

Dynamic Replica Sets



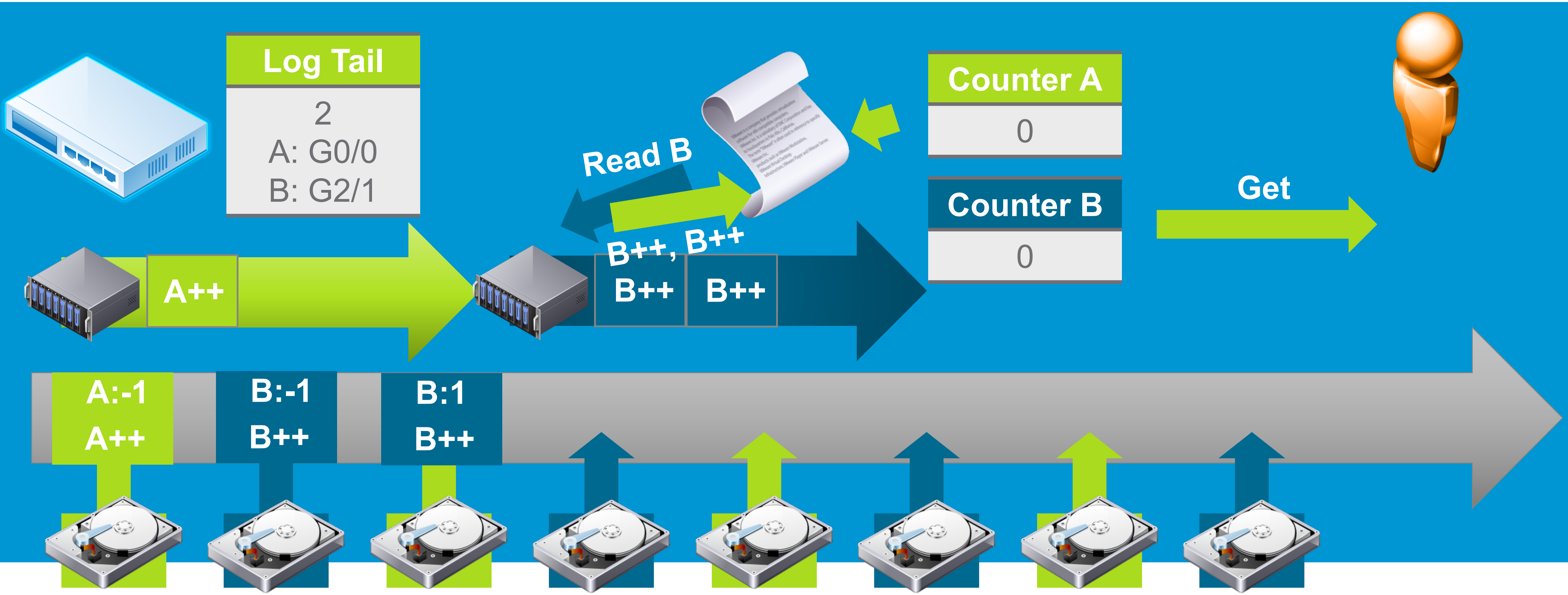
In vCorfu, replica sets are no longer static. Instead, we dynamically generate replica sets based on two indexes, the log address and the stream id plus stream address.

Dynamic Replica Sets



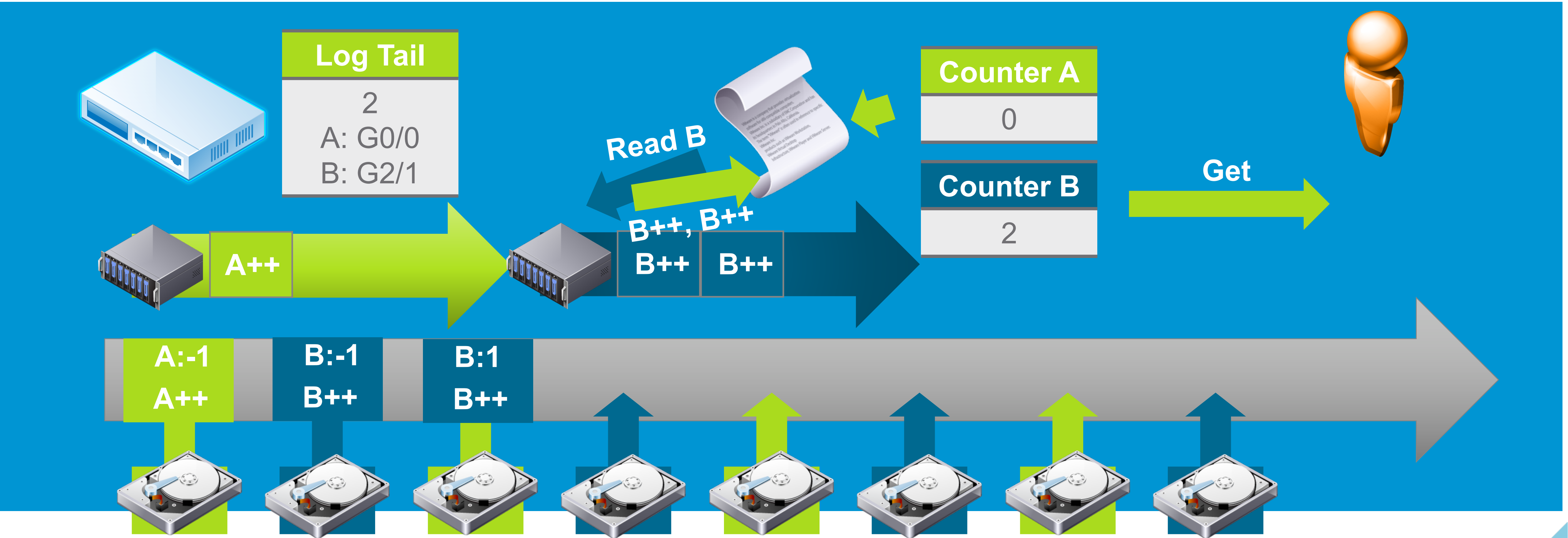
So that in this example, three different replica sets are constructed, instead of the static chain replication protocol in Corfu/Tango.

Example: Reading a Counter



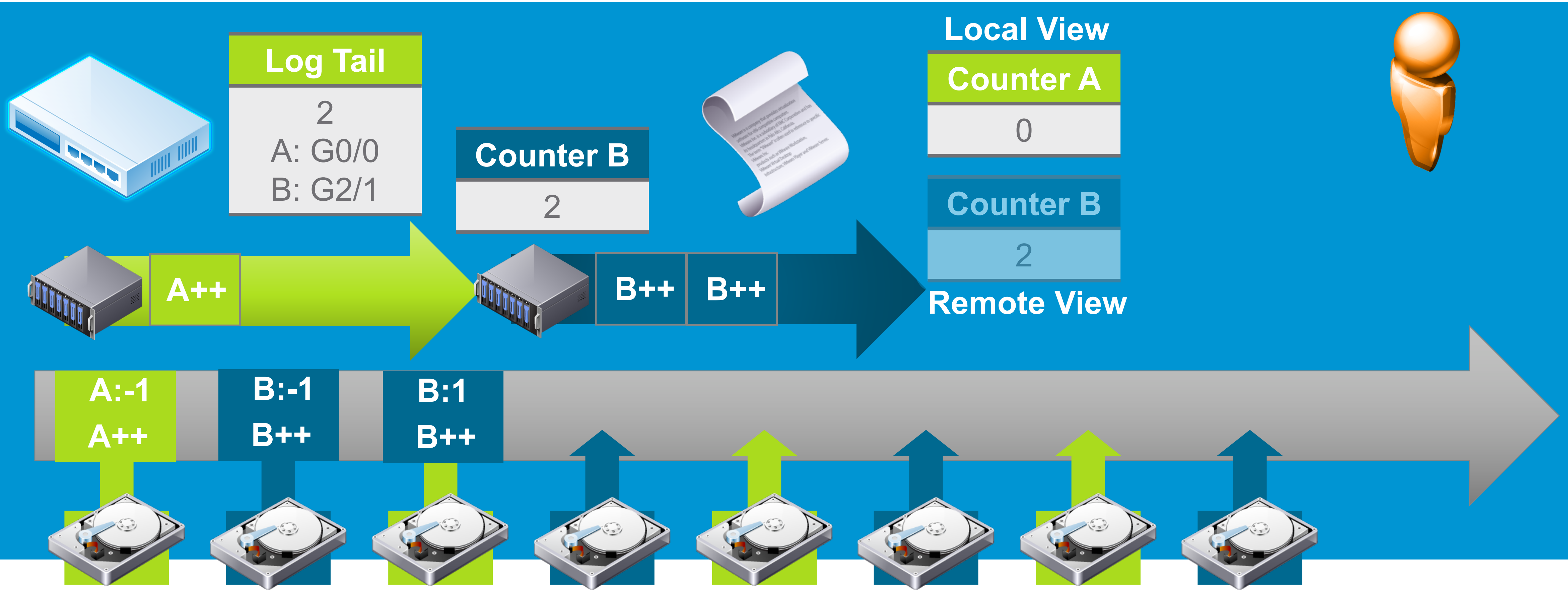
With materialized streams, reading is greatly simplified. Now instead of reading backpointers in sequence, we can read an entire stream with one request.

Example: Reading a Counter



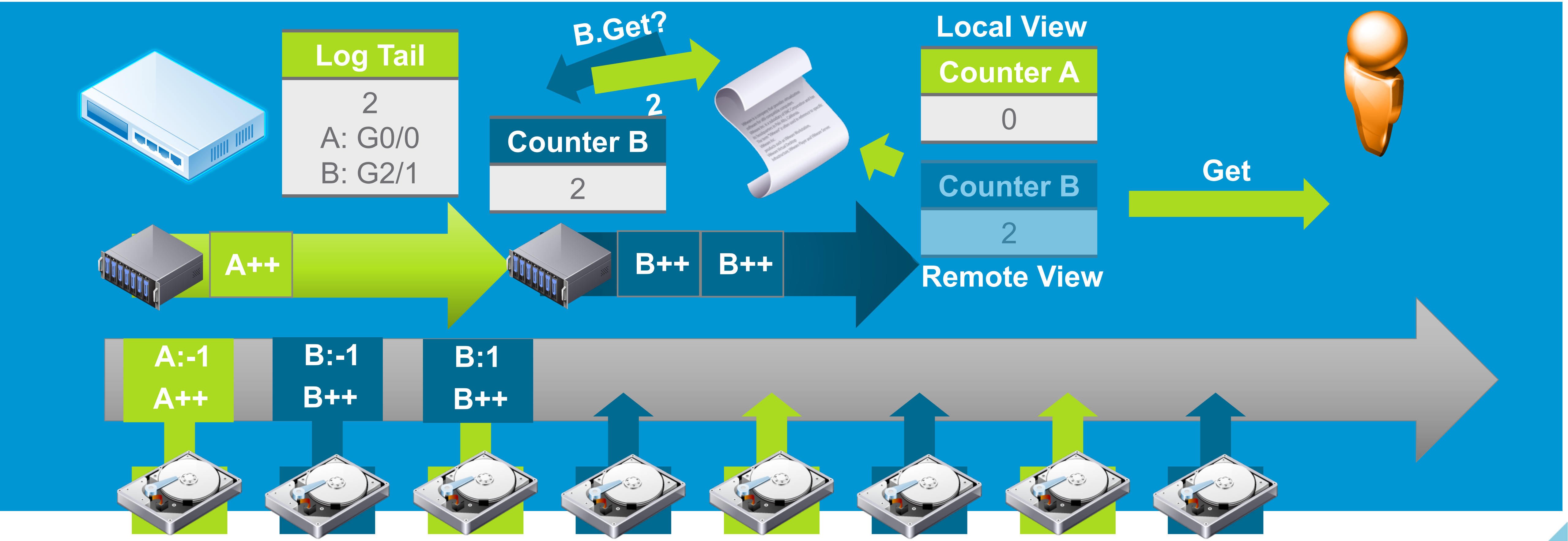
Now we can easily update counter B without contacting multiple replicas.

Remote Views and Local Views



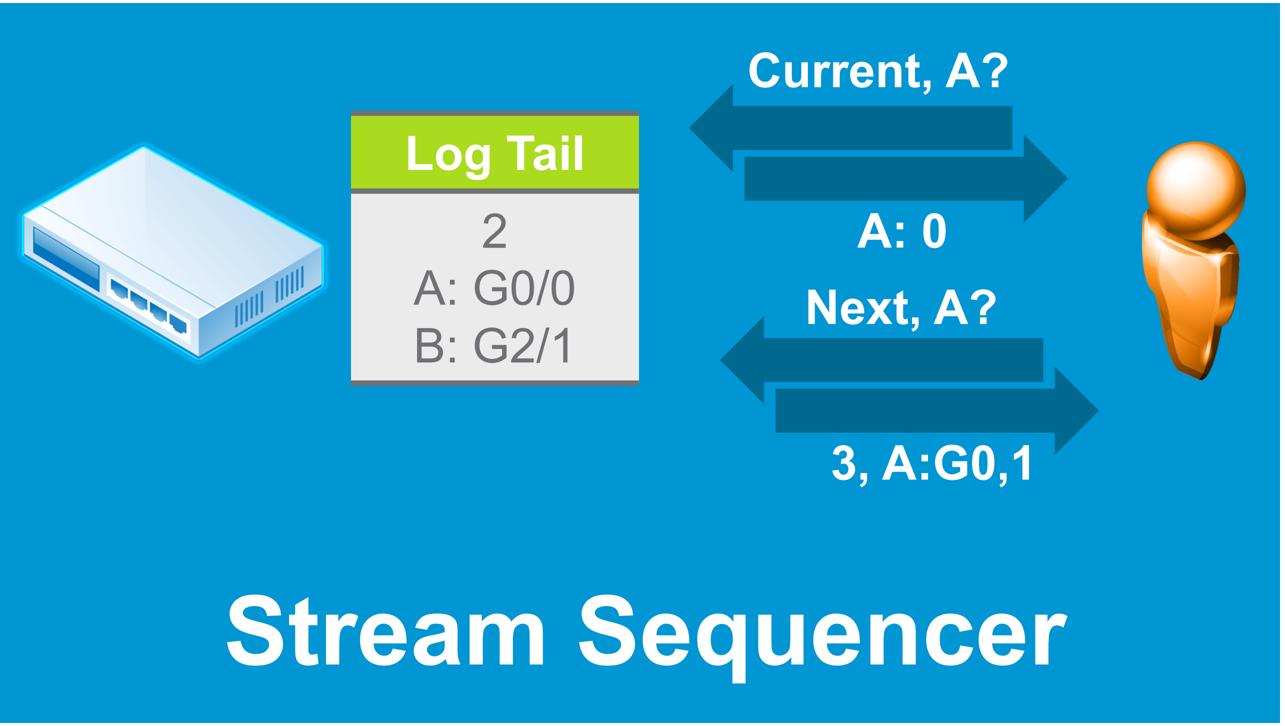
Having a single replica hold all the updates for a stream allows us to delegate playback to that replica.

Example: Reading a Counter with a Remote View



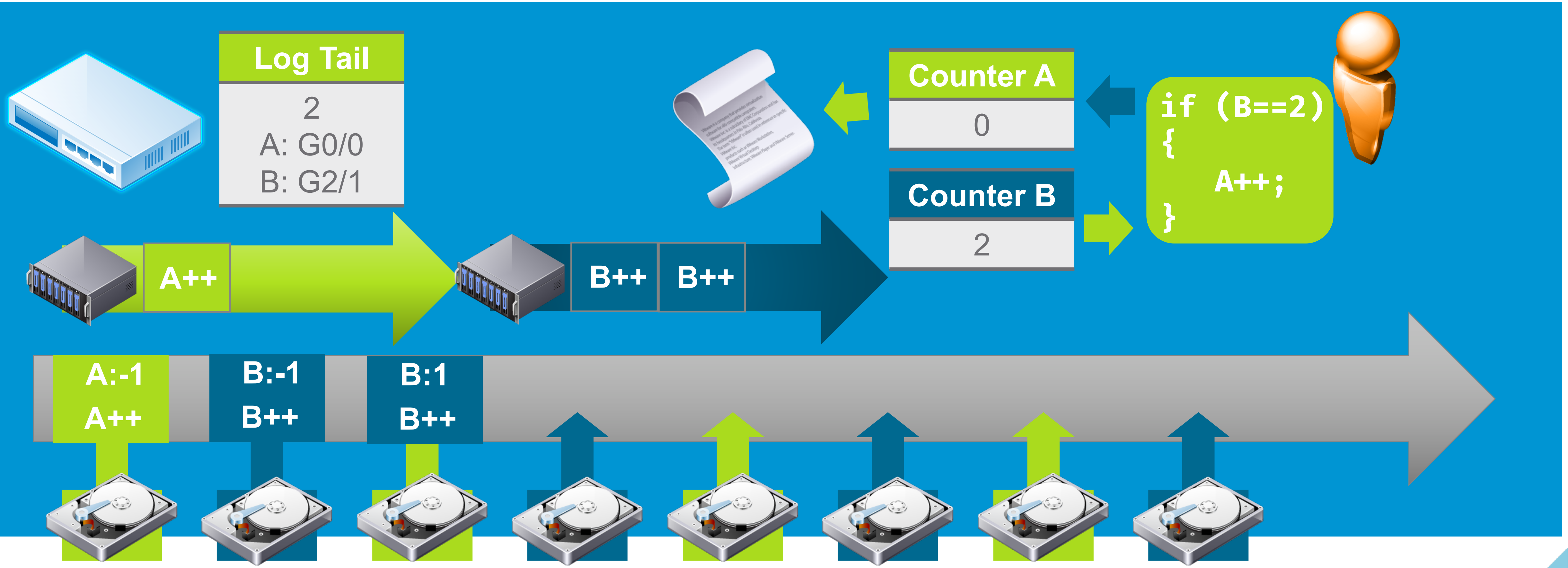
With a remote view, the client doesn't need to have the updates or the state machine in memory.

Modifying an Existing Component: Lightweight Transaction Resolution



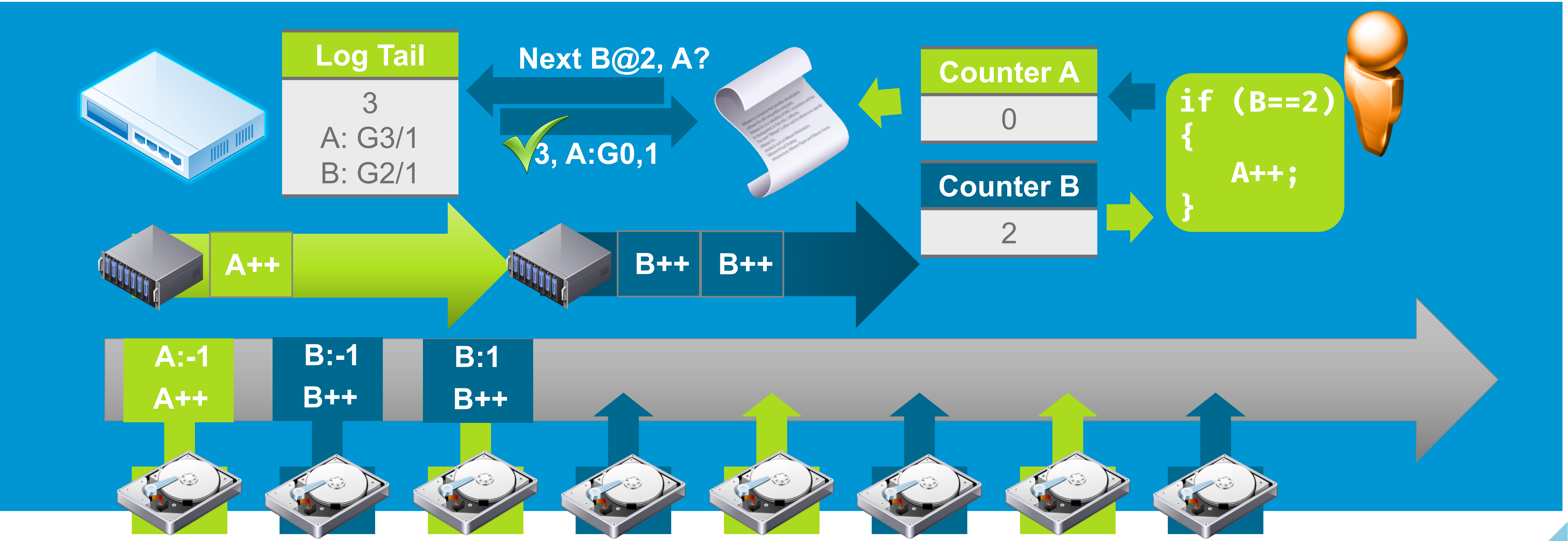
By adding conditional address issuance, the sequencer can perform transaction resolution.

Example: Incrementing Multiple Counters Conditionally



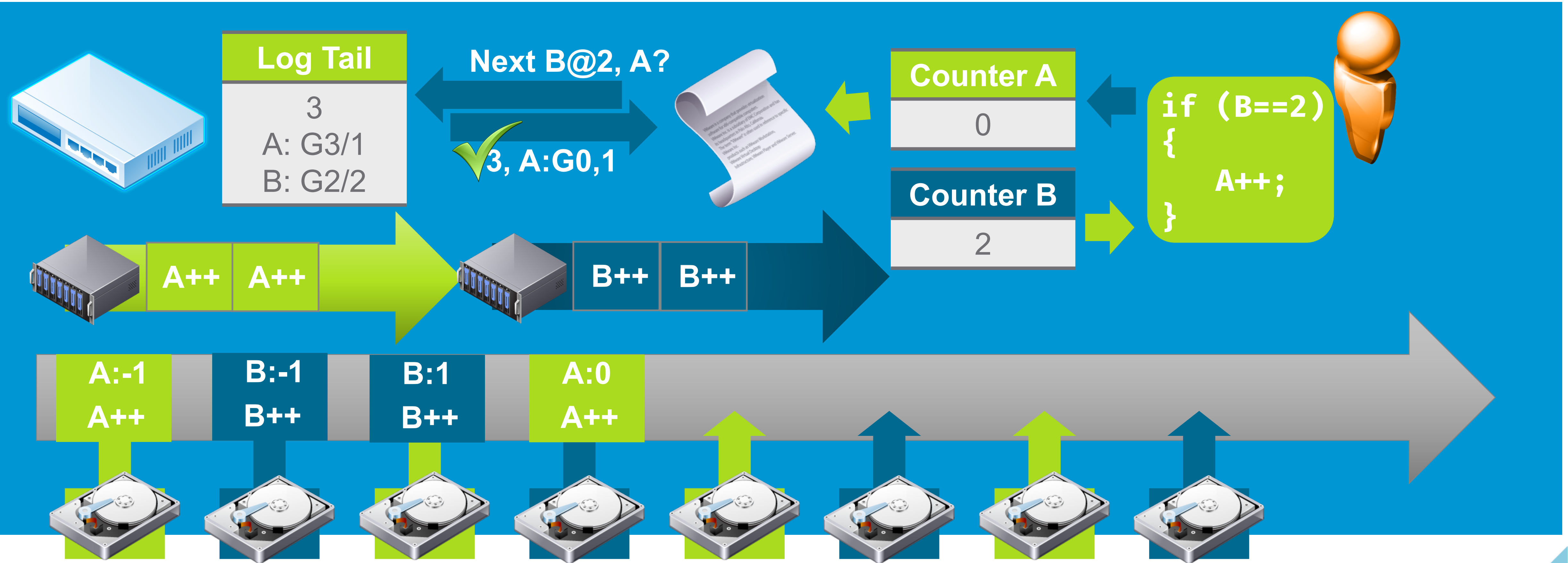
The transaction in this example reads counter B and increments counter A if counter B is equal to two.

Example: Incrementing Multiple Counters Conditionally



The client performs this transaction optimistically, and requests an address only if counter B has not changed since the client accessed it. In this case, it has not, so the address is granted.

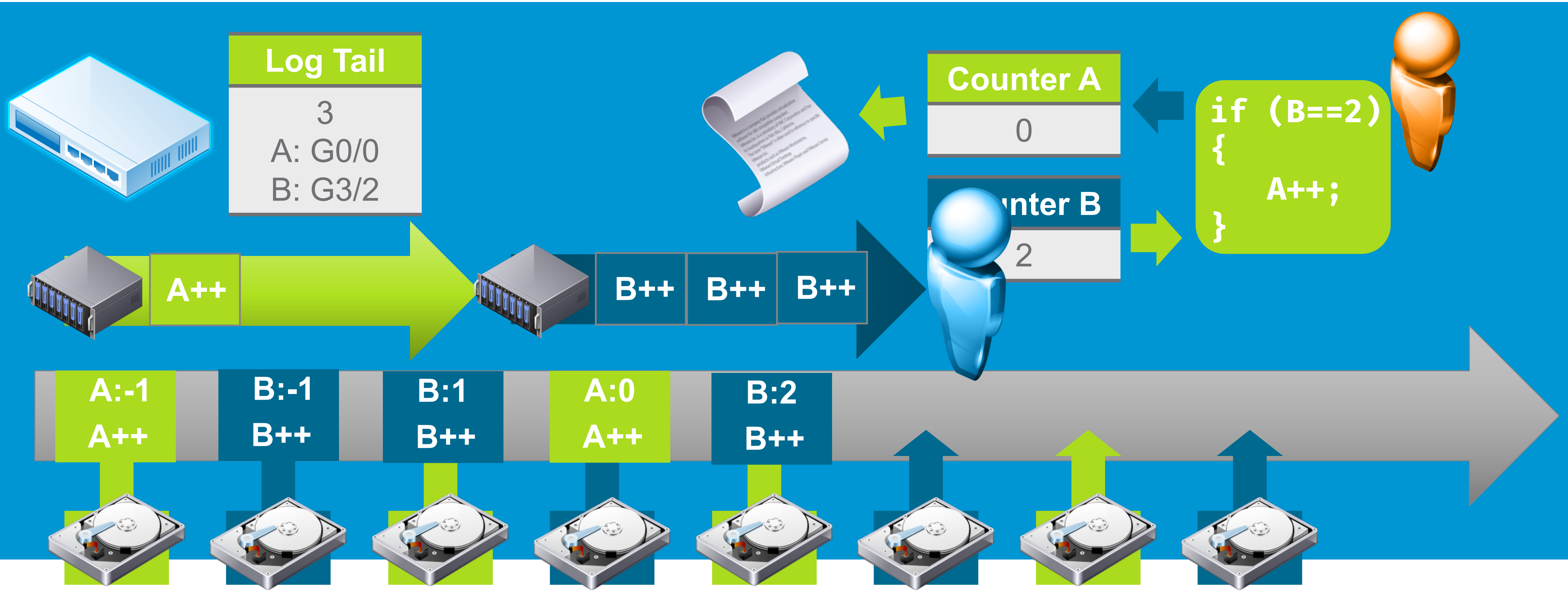
Example: Incrementing Multiple Counters Conditionally



Now, clients can read this update directly, and a client trying to determine Counter A's state does not need to read counter B at all.

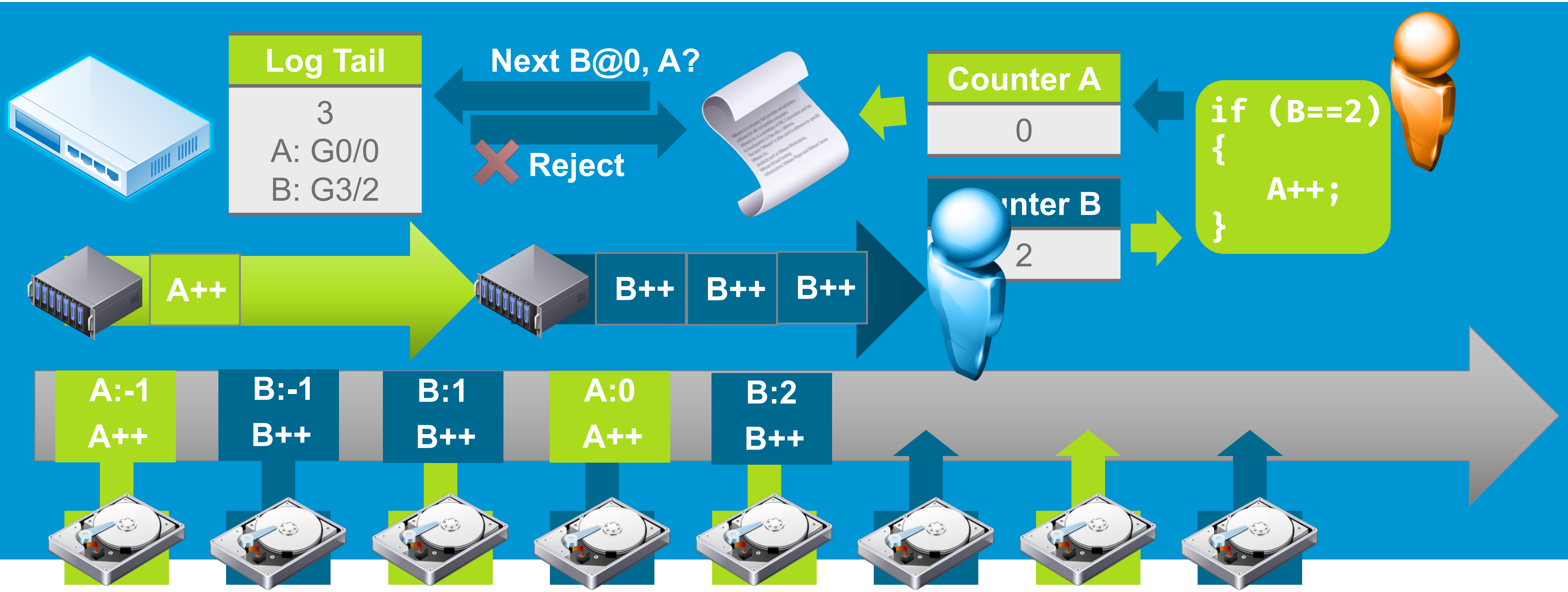
vmware®

Example: Incrementing Multiple Counters Conditionally



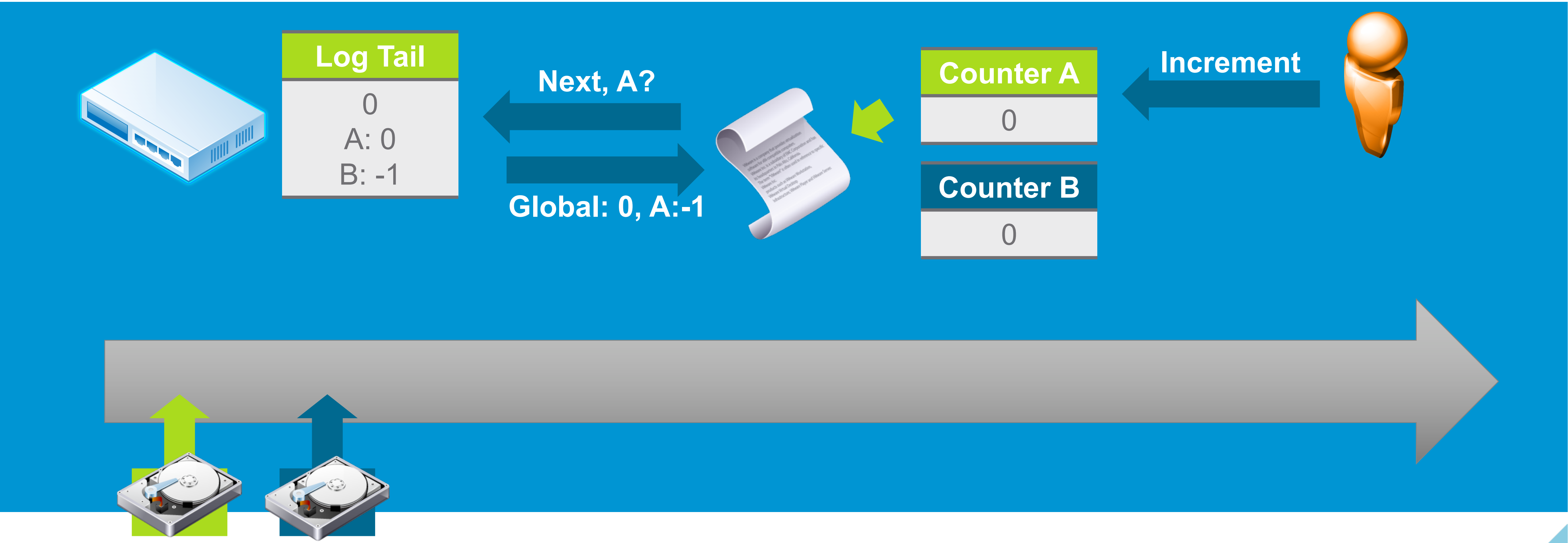
In the case that another client modifies a read object, causing the optimistic view of counter B to become invalid…

Example: Incrementing Multiple Counters Conditionally



The sequencer will reject the client's request for an address – all by doing a simple comparison (B@0 < 3).

Example: Incrementing a Counter



And the sequencer responds with the current global address and previous stream addresses, incrementing the counters for the log and the stream.

vmware