



Using Lightweight Formal Methods to Validate a Key-Value Storage Node in Amazon S3

James Bornholt

AWS & UT Austin

Rajeev Joshi

AWS

Vytautas Astrauskas

ETH Zurich

Brendan Cully

AWS

Bernhard Kragl

AWS

Seth Markle

AWS

Kyle Sauri

AWS

Drew Schleit

AWS

Grant Slatton

AWS

Serdar Tasiran

AWS

Jacob Van Geffen

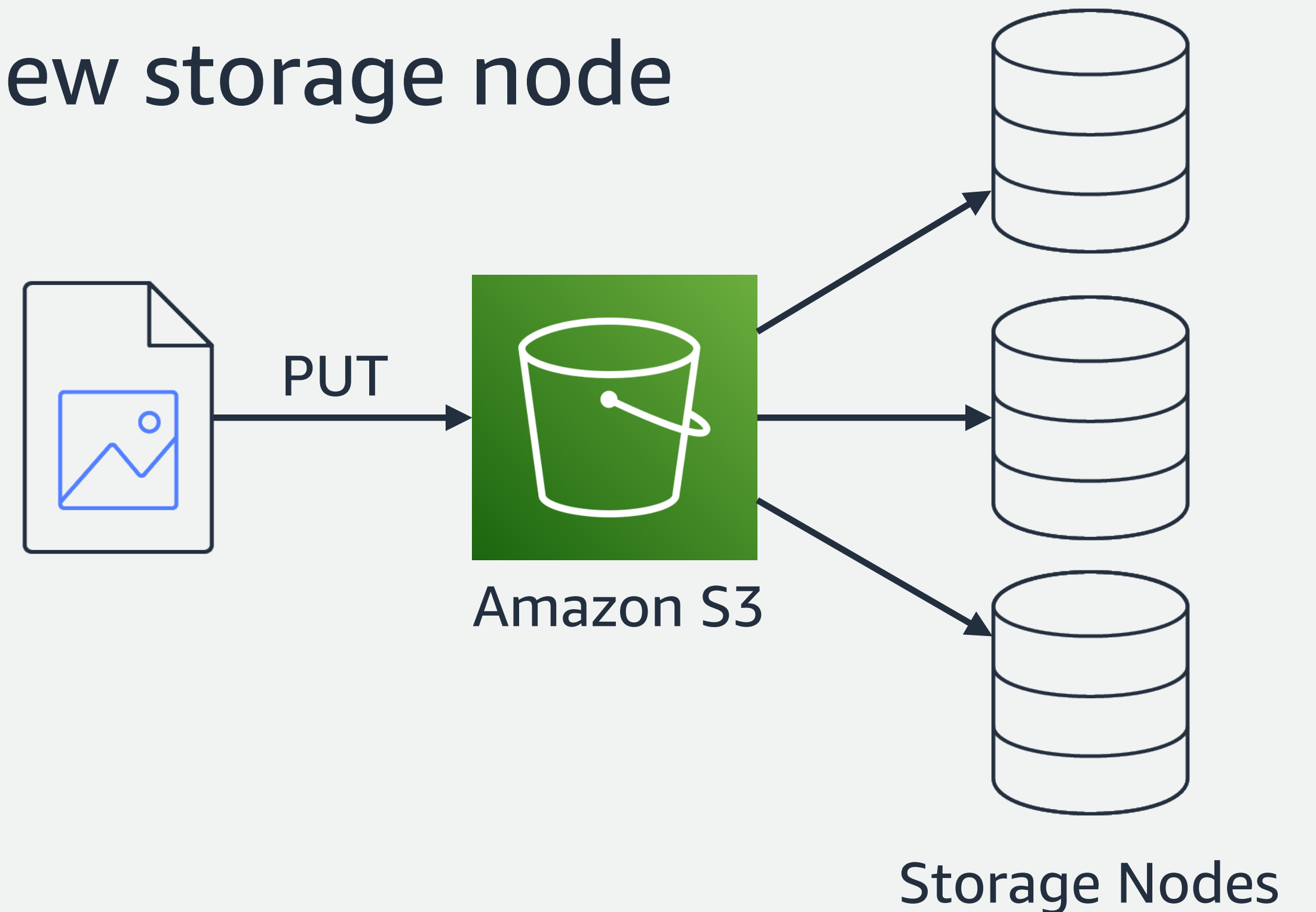
University of Washington

Andrew Warfield

AWS

S3's new ShardStore storage node

- Amazon S3 is an object storage service (PUT, GET) holding over 100 trillion objects
- We replicate object data on *storage nodes*
- Currently deploying **ShardStore**, a new storage node written in Rust



Formal methods for ShardStore

- Production storage systems are complex and frequently changing
 - Crash consistency, concurrency, IO, etc.
 - Over 40,000 lines of Rust, deployed weekly
- *Formal methods* can help increase confidence, but challenging to incorporate in a rapid development process

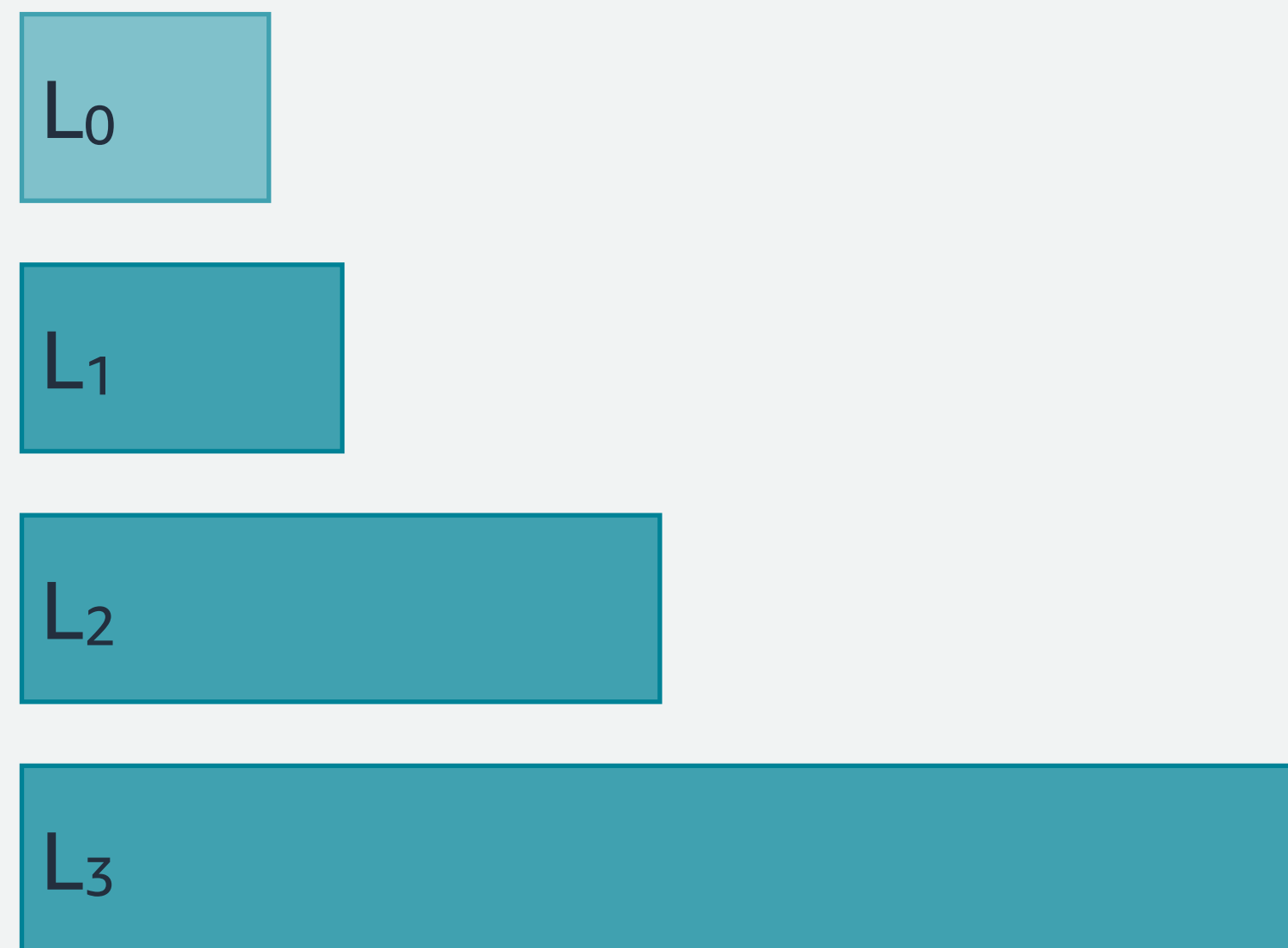
***Lightweight* formal methods**

1. Executable *reference models* as specifications
2. Automated tools to check implementations against models
3. Coverage tools to track effectiveness over time

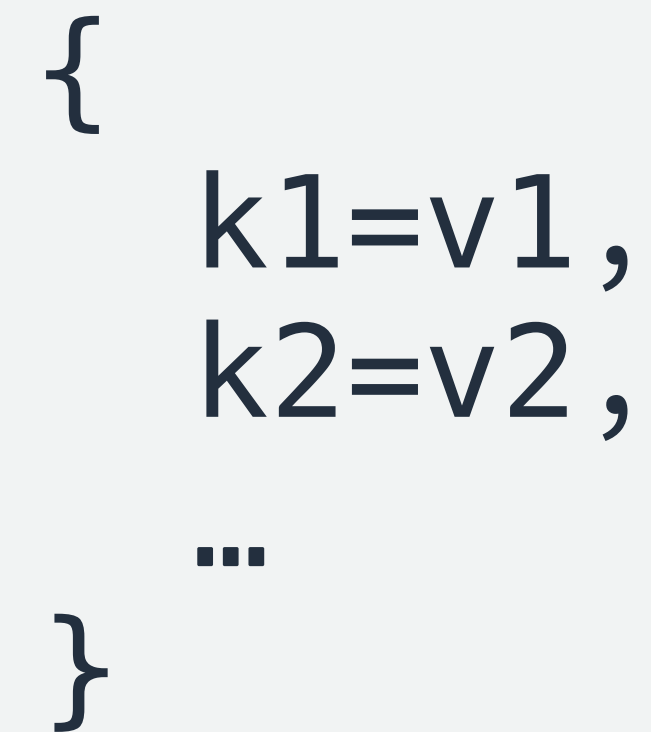
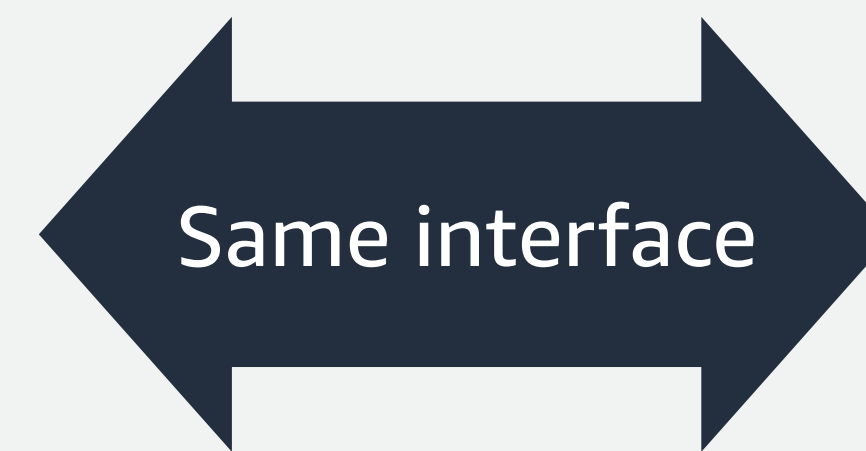
In return for being lightweight and automated, we accept weaker correctness guarantees than full formal verification

Writing *reference model specs*

- Small, executable specifications, written in Rust, alongside the code



LSM tree



Hash map

Correctness properties

- Decompose correctness into three parts and check each separately:
 - Sequential correctness: refinement of the reference model
 - Crashes: refinement against a weaker reference model
 - Concurrency: linearizability against the reference model

Property-based testing for refinement

Random sequence:

Put(a, 5)

GC

Delete(a)

Property-based testing for refinement

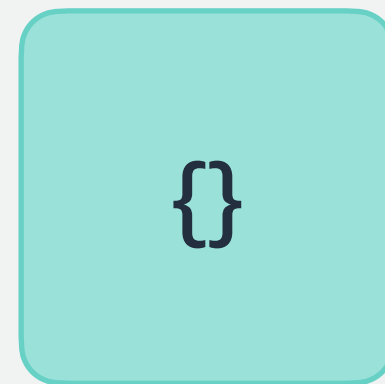
Random sequence:

Put(a, 5)

GC

Delete(a)

Reference model:



Implementation:



Property-based testing for refinement

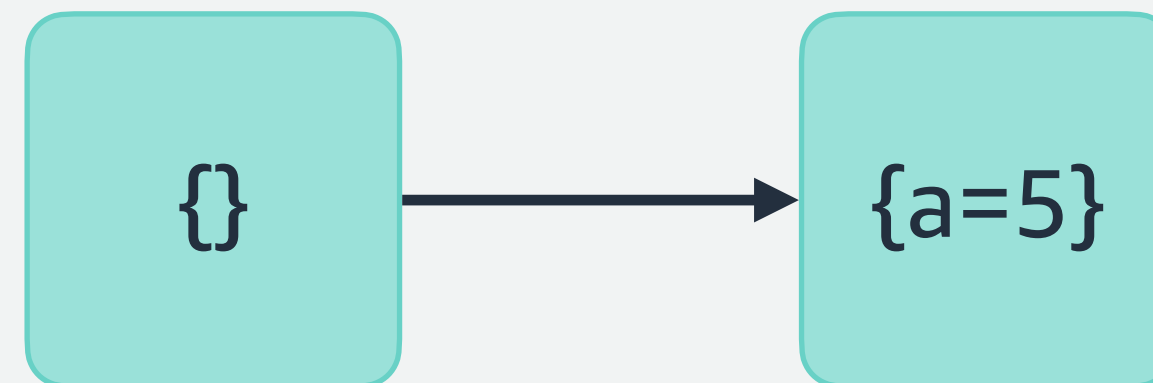
Random sequence:

Put(a, 5)

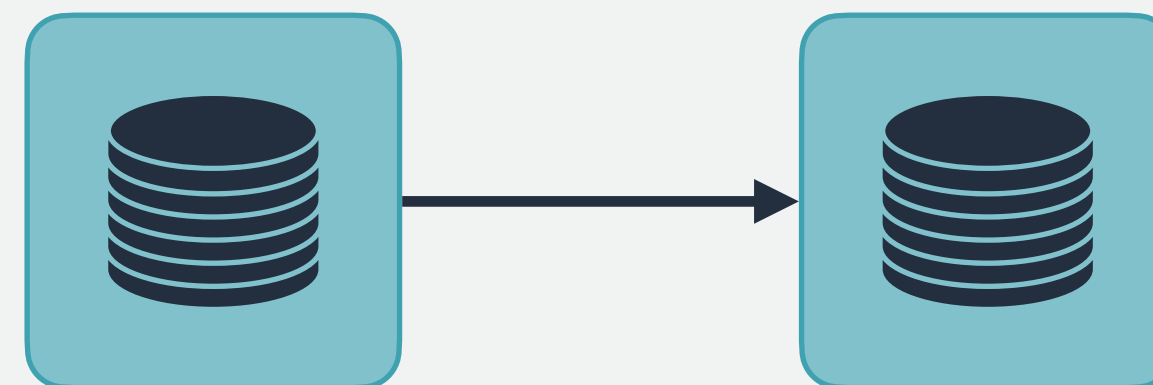
GC

Delete(a)

Reference model:



Implementation:



Property-based testing for refinement

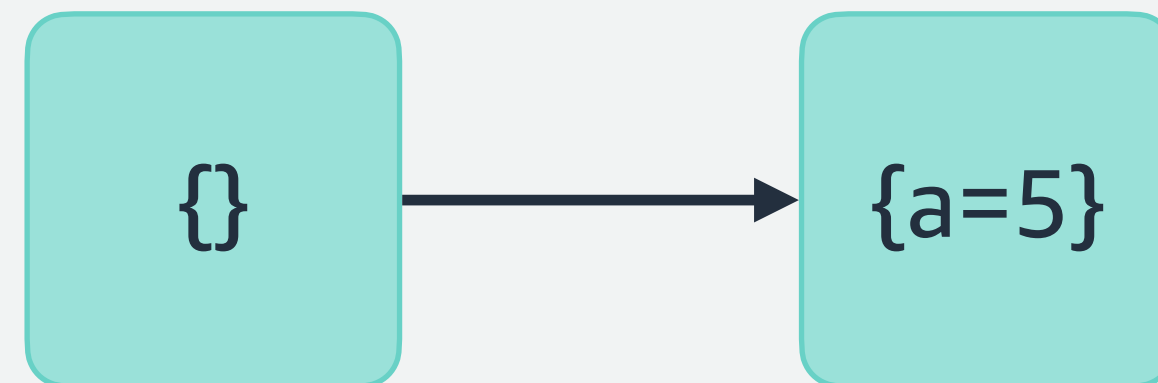
Random sequence:

Put(a, 5)

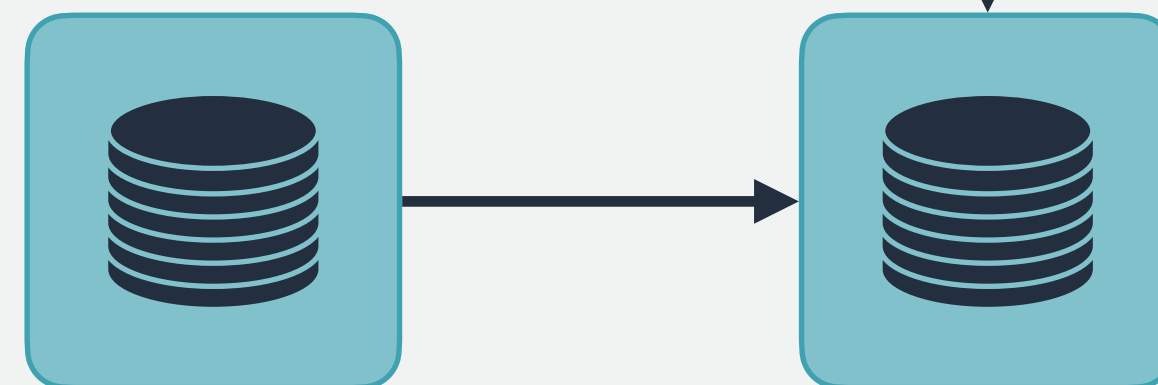
GC

Delete(a)

Reference model:



Check for same
key-value
mapping



Implementation:

Property-based testing for refinement

Random sequence:

Put(a, 5)

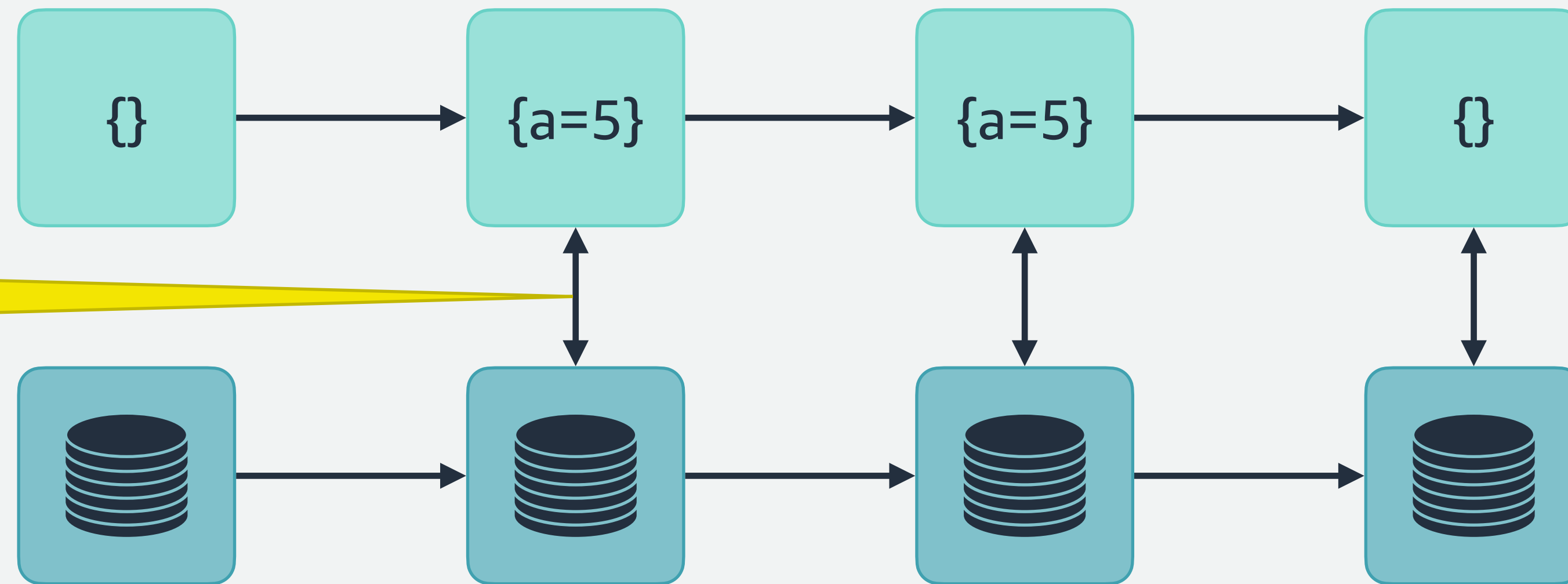
GC

Delete(a)

Reference model:

Check for same
key-value
mapping

Implementation:



Property-based testing for refinement

"Pay-as-you-go": test small scale locally, larger scale before deployment

Random sequence:

Put(a, 5)

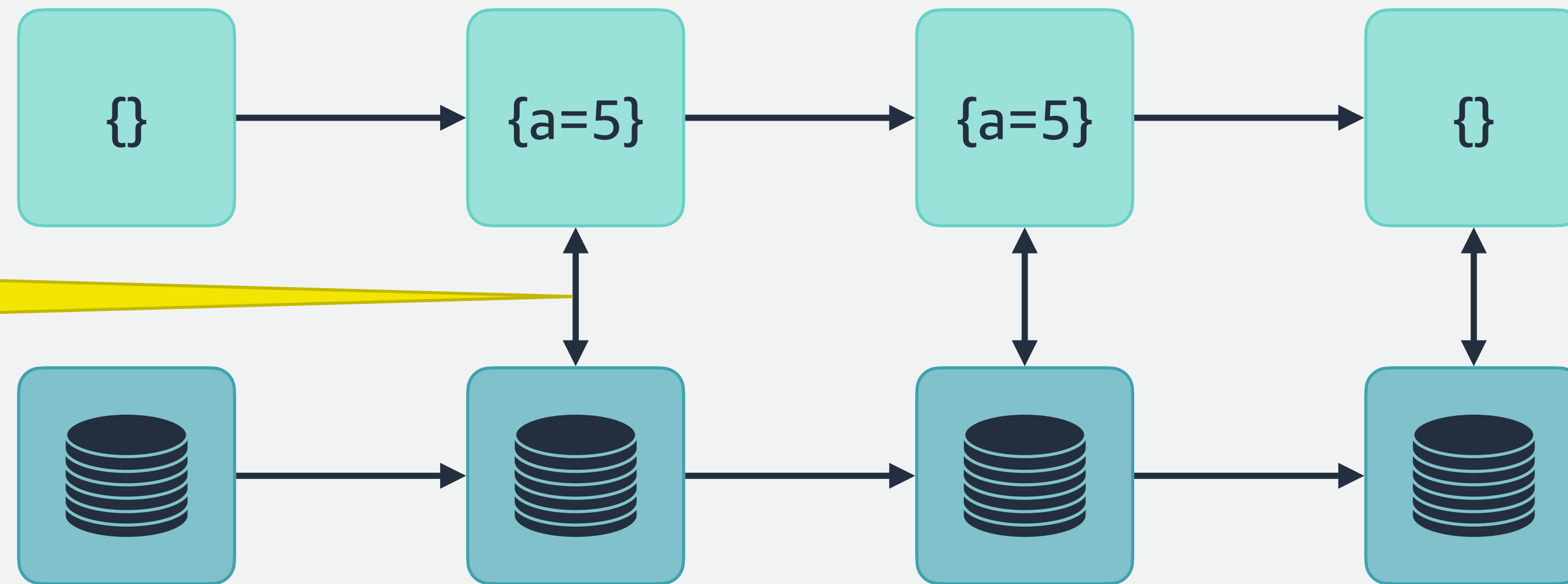
GC

Delete(a)

Reference model:

Check for same key-value mapping

Implementation:



Experience with FM in production

- Automated lightweight tools prevent issues from even reaching code review
- Maintainable: 20% of model code by non-FM experts; 1/3rd of engineers have written their own new models/checks
- “Pay-as-you-go” and continuous validation makes FM viable in a rapid production engineering process



Using Lightweight Formal Methods to Validate a Key-Value Storage Node in Amazon S3

Thank you!

We're hiring (full-time and interns)!

s3-arg-jobs@amazon.com or bornholt@amazon.com