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

Authors: James Bornholt, Rajeev Joshi , Vytautas Astrauskas , Brendan Cully , Bernhard Kragl , Seth Markle , Kyle Sauri , Drew Schleit , Grant Slatton , Serdar Tasiran , Jacob Van Geffen , Andrew Warfield

Keywords: cloud storage, lightweight formal methods

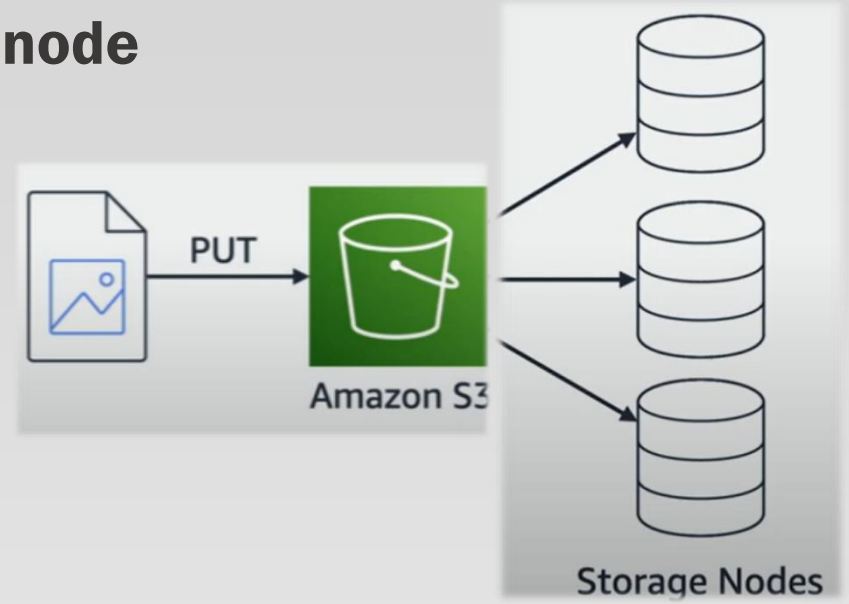
From: SOSP2021

Date: 2021.12.17

Reporter: 夏谦

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

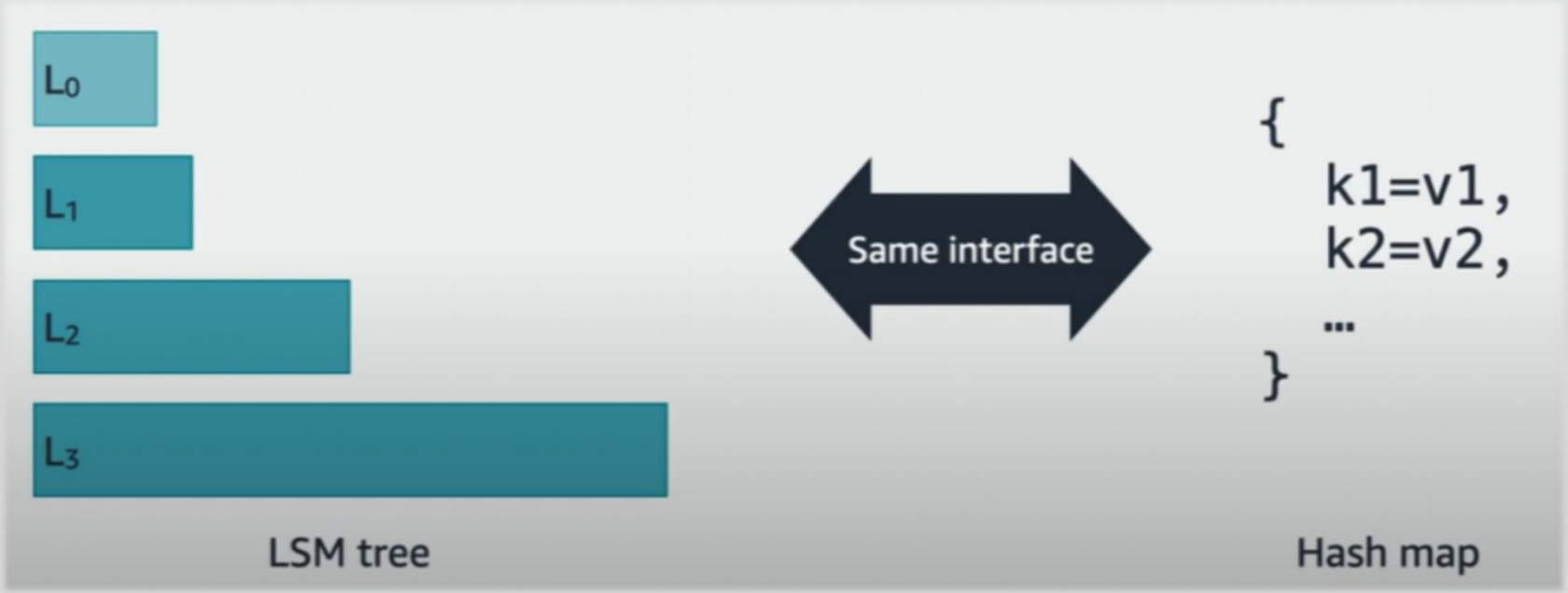
Lightweight formal methods

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**



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

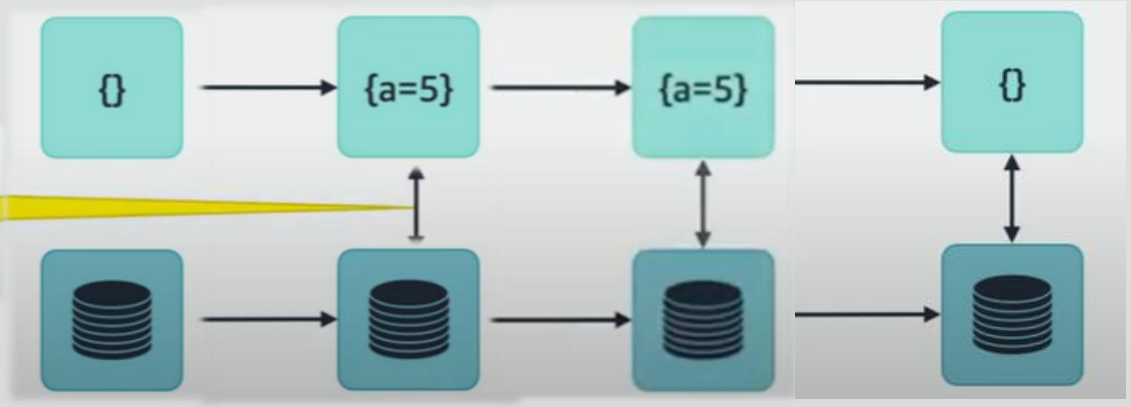
Property-based testing for refinement

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

Random sequence: Put(a,5) GC Delete(a)

Reference model:

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