AWS Lambda
(specifically container loading)

Marc Brooker

VP/Distinguished Engineer

mbrooker@amazon.com

“AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources.”

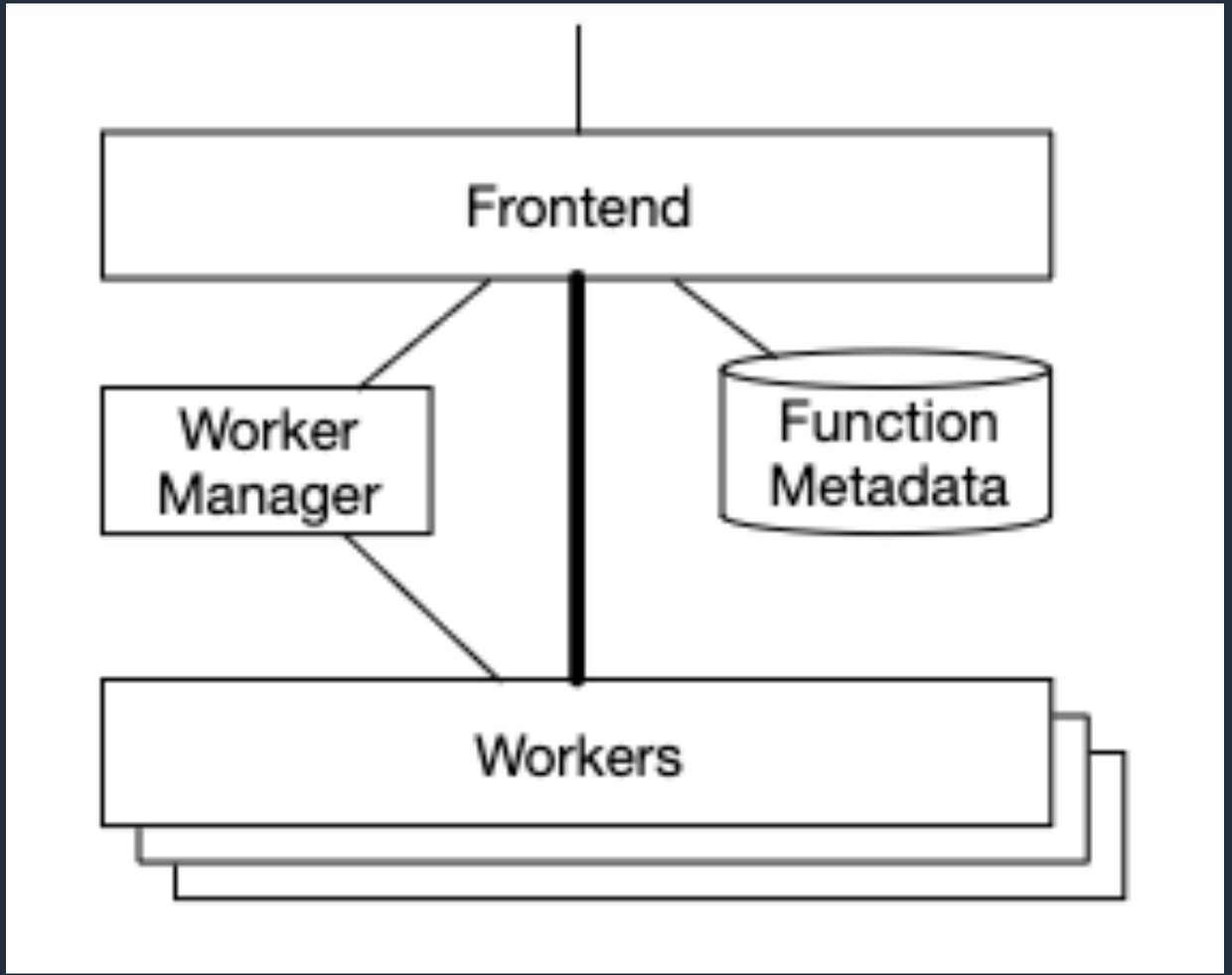
```python
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'body': json.dumps('Hello 6.5840')
    }
```

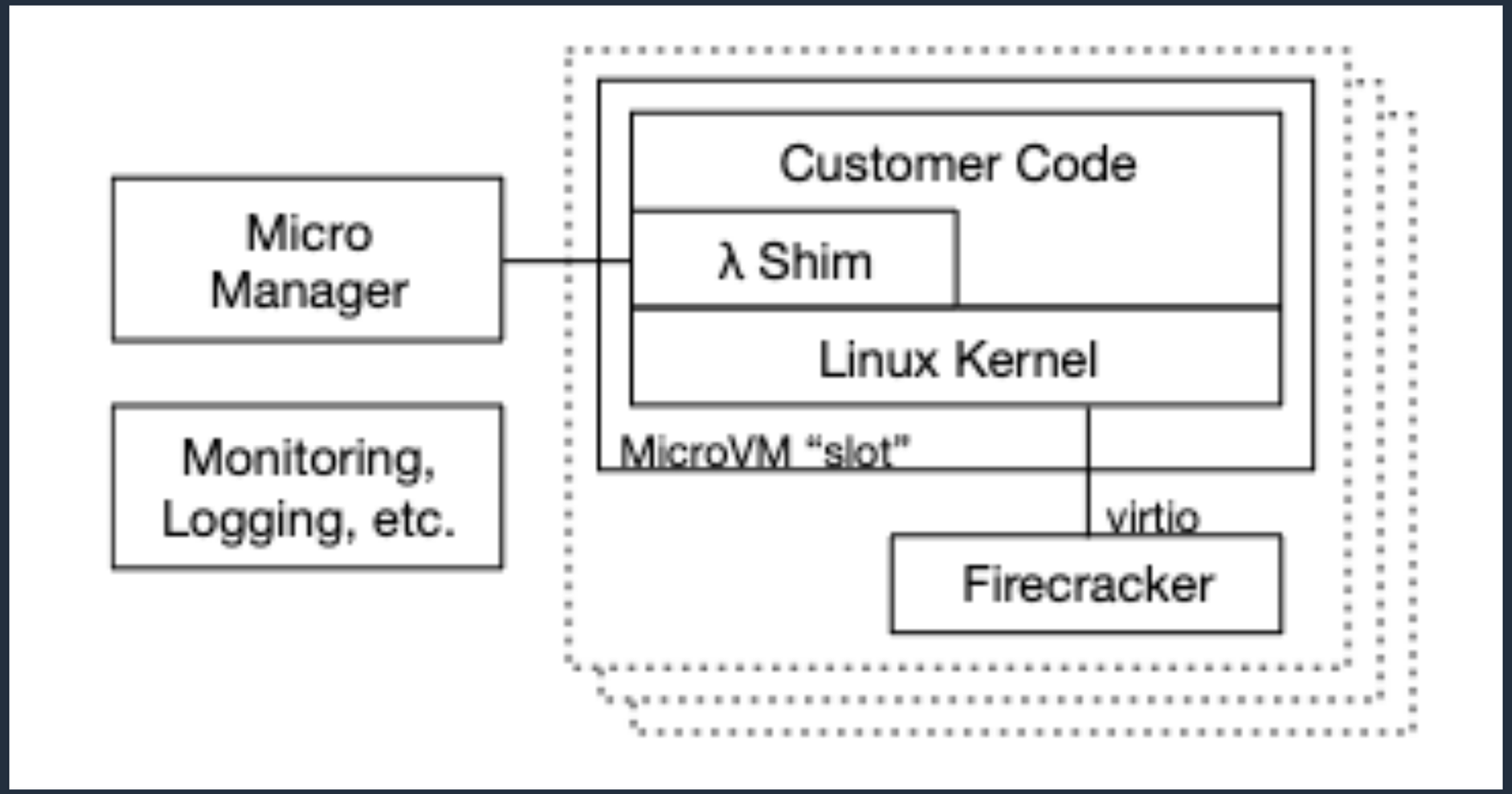
Why?

- Building scalable, fault-tolerant systems is hard.
- Driving high hardware utilization is hard.
    - Multi-tenancy makes it easier.
- The cloud needs *glue.*

- *Code close to data is more efficient?*

aws





```python
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'body': json.dumps('Hello 6.5840')
    }
```

“… simple …”

“… easy …”

“… fast …”

“… complex …”

“… complicated …”

“… difficult …”

Before	After
250MB max	10GB max
Code or .zip	Container image
Custom tools	Standard tools

aws

Before	After
250MB max	10GB max
Code or .zip	Container image
Custom tools	Standard tools

without slower cold starts!

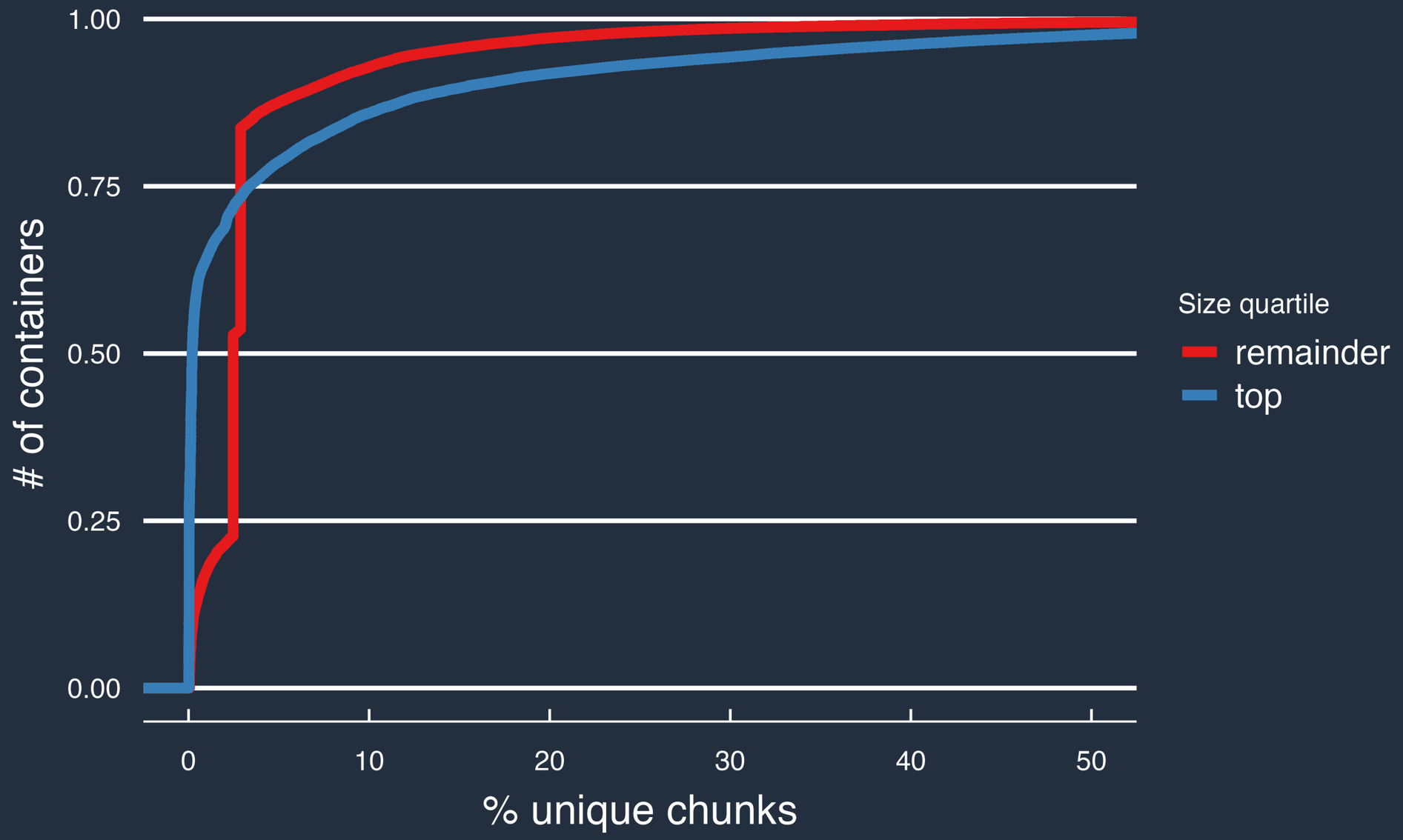
aws

150 Pb/s

Harter et al: Only 6.4% of container data is needed at startup!

Harter et al, *Slacker: Fast Distribution with Lazy Docker Containers*, FAST'16

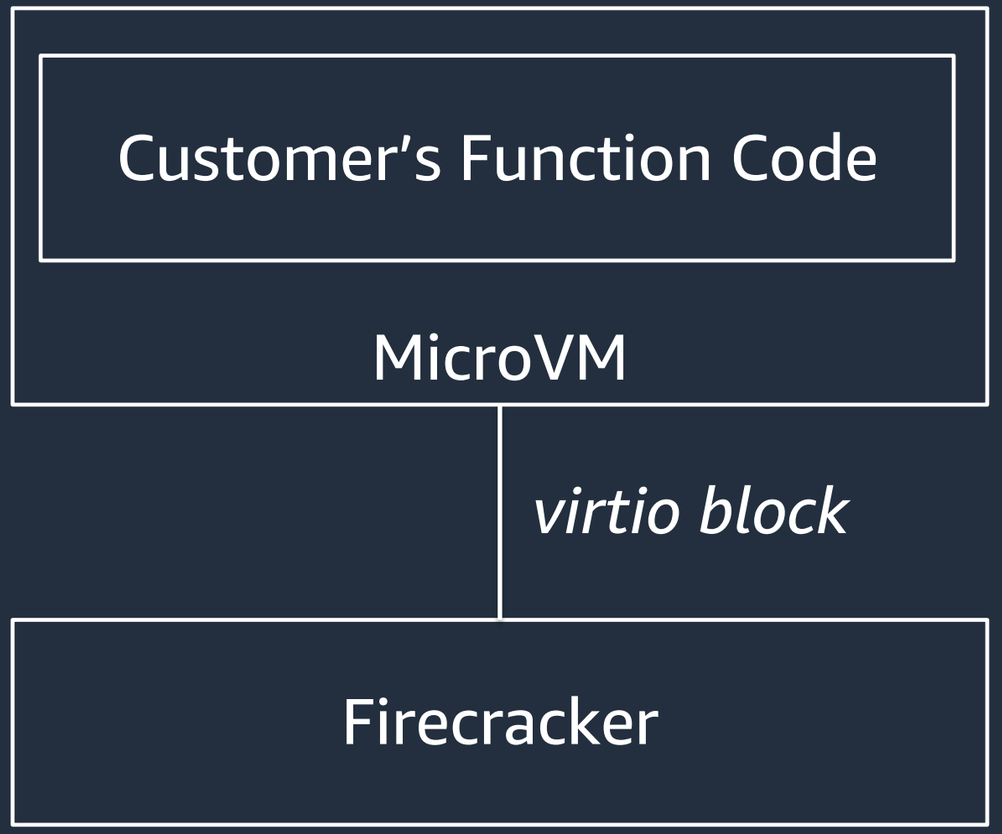
aws



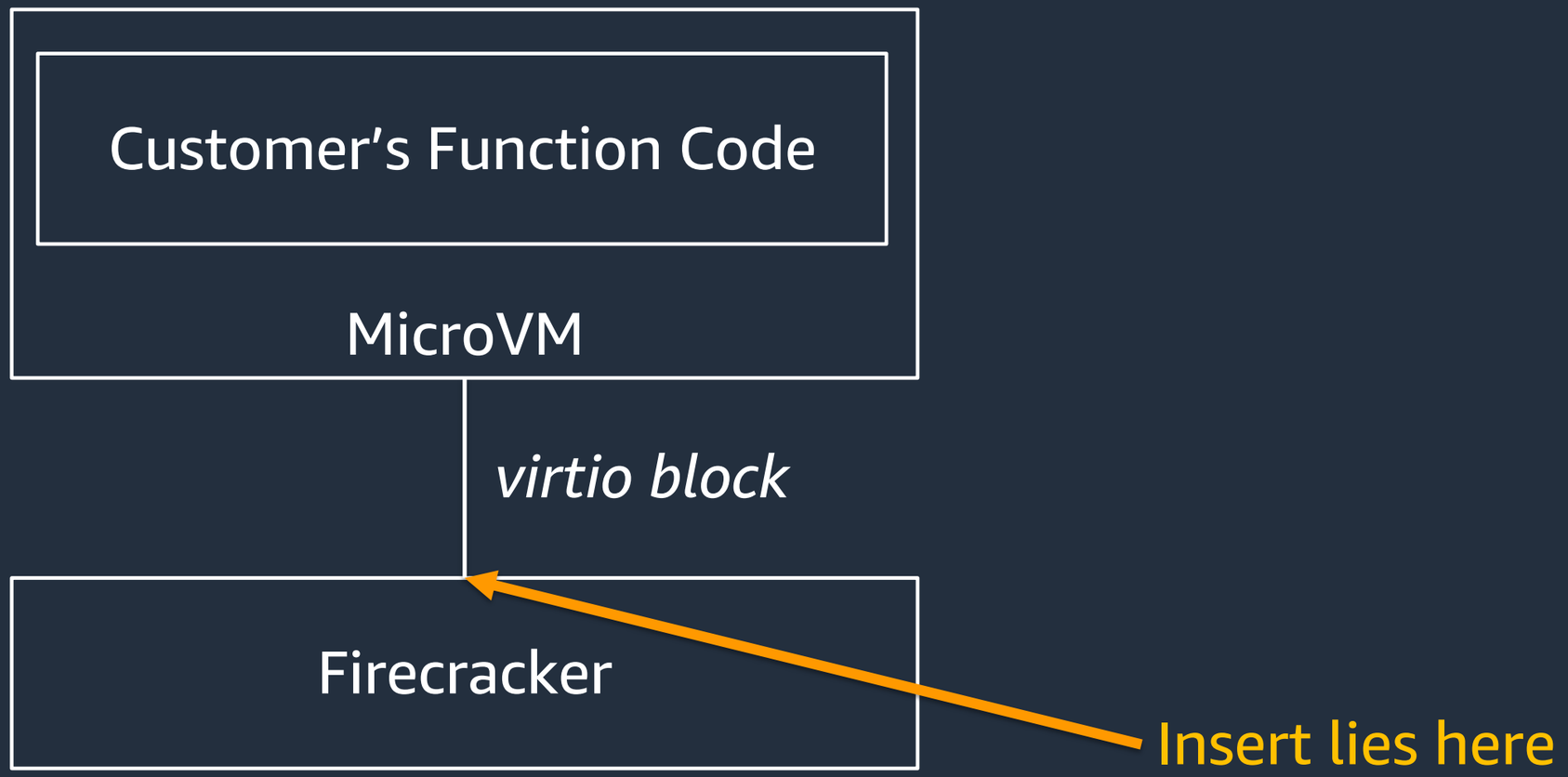
aws

Find a way, invisible to applications, to load data on demand and deduplicate common data.

aws



Agache et al, “Firecracker: Lightweight Virtualization for Serverless Applications”, AWS, 2020



But…
Container images aren’t filesystems.

Deterministic Flattening

Ext4 filesystem image

512kB chunks

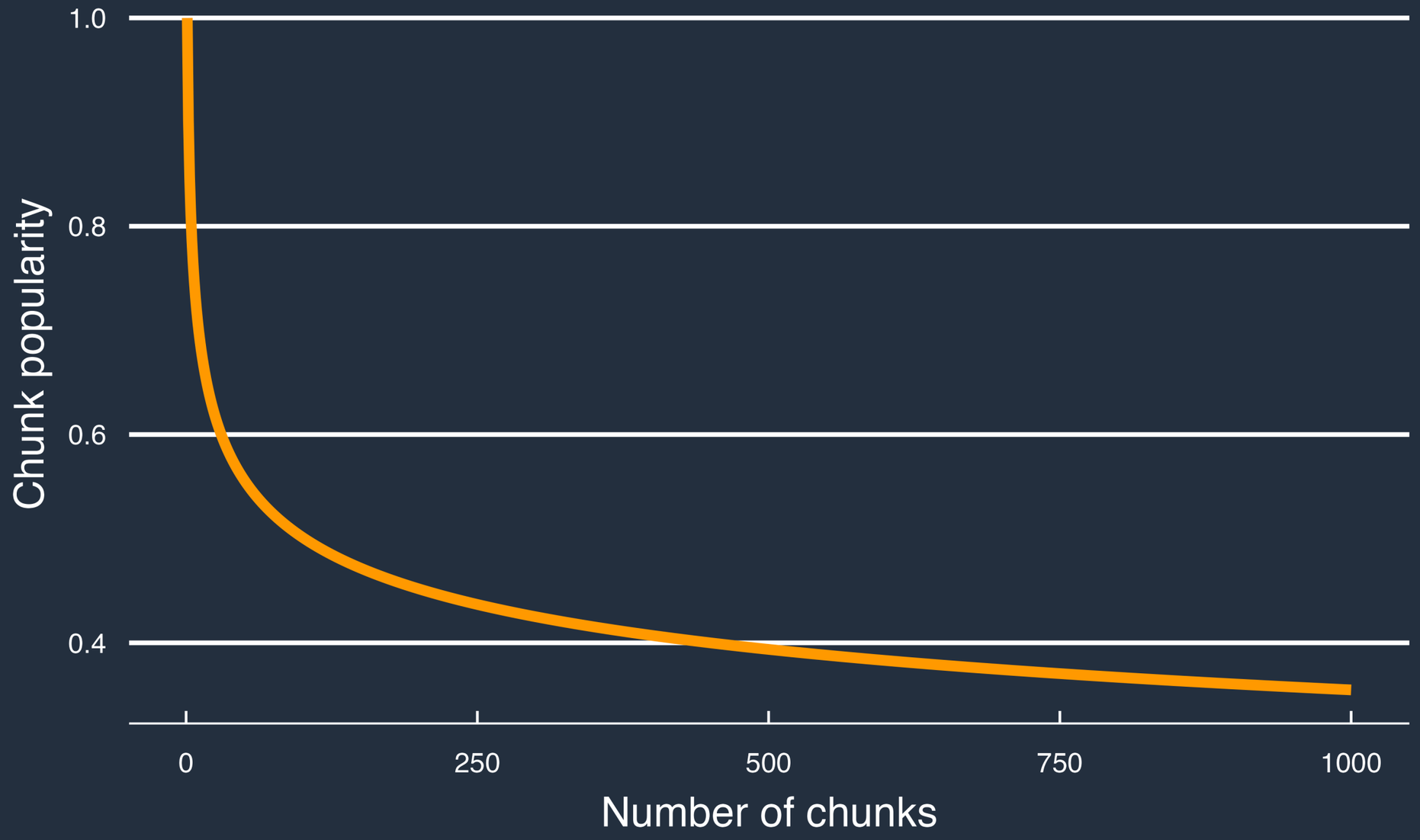
aws

But…
Real storage is really fast (<100µs), so loading on demand must be really fast too.

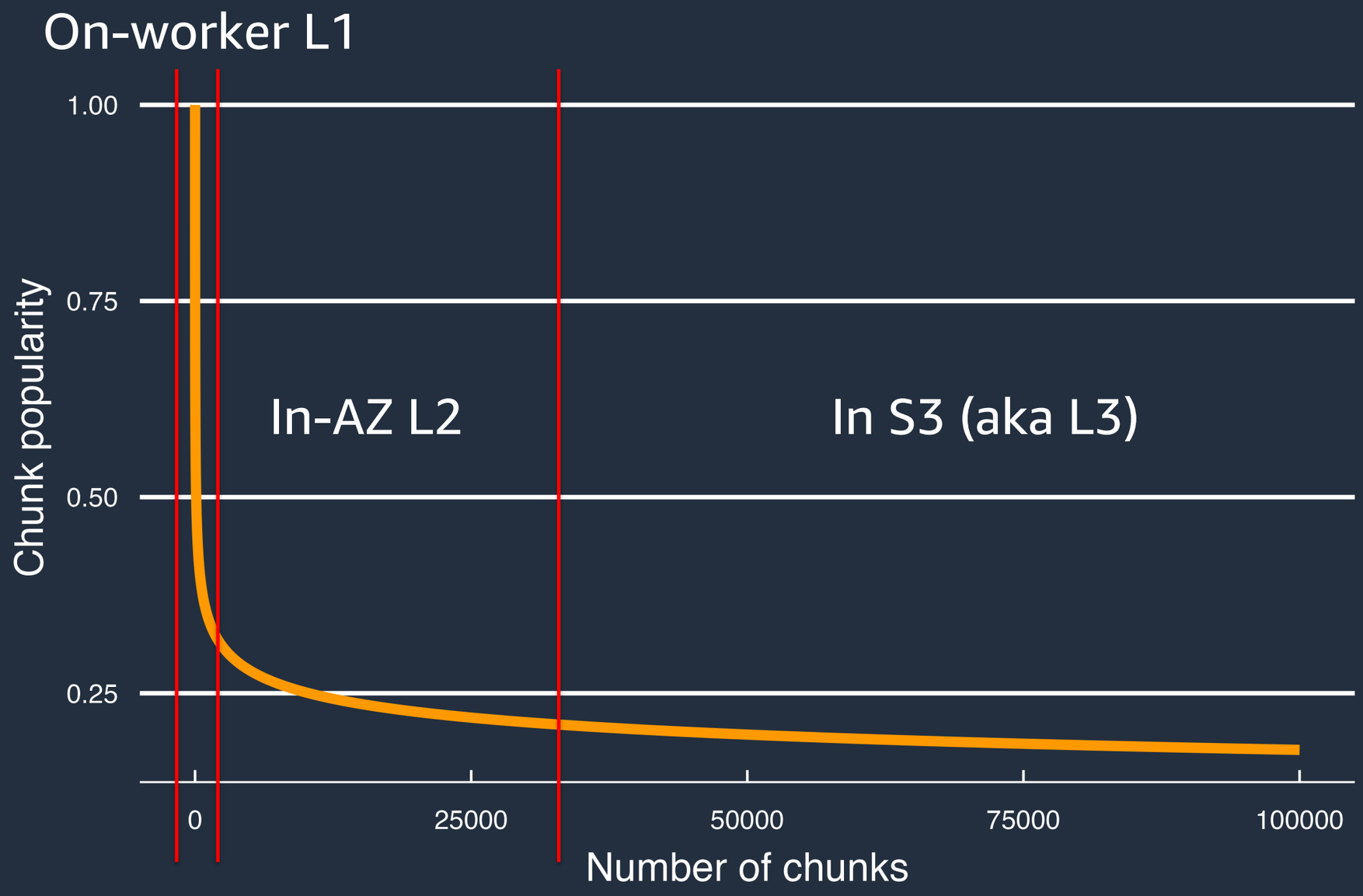
Lambda worker

Amazon S3 Express One Zone

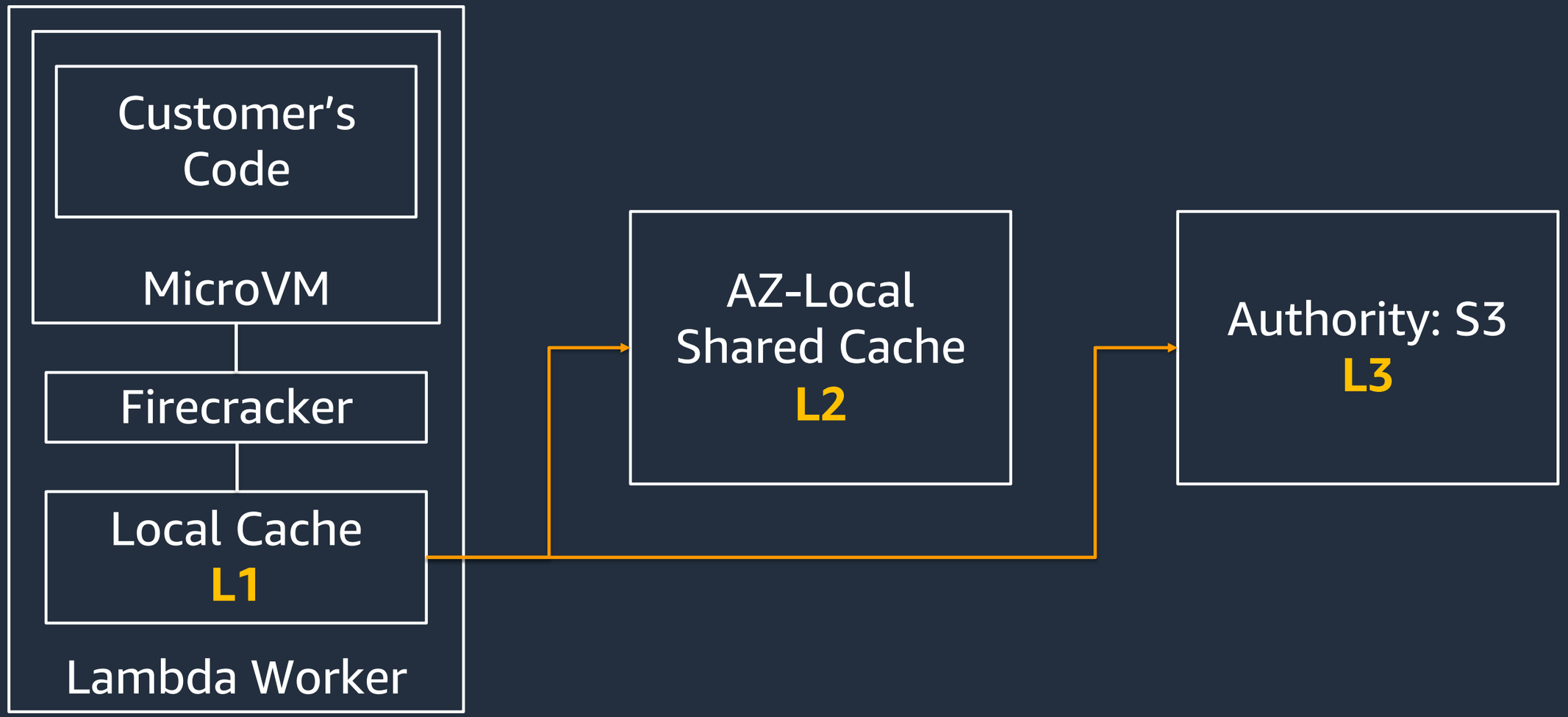
Delivers consistent single-digit millisecond request latency on hundreds of thousands of transactions per second for faster data processing

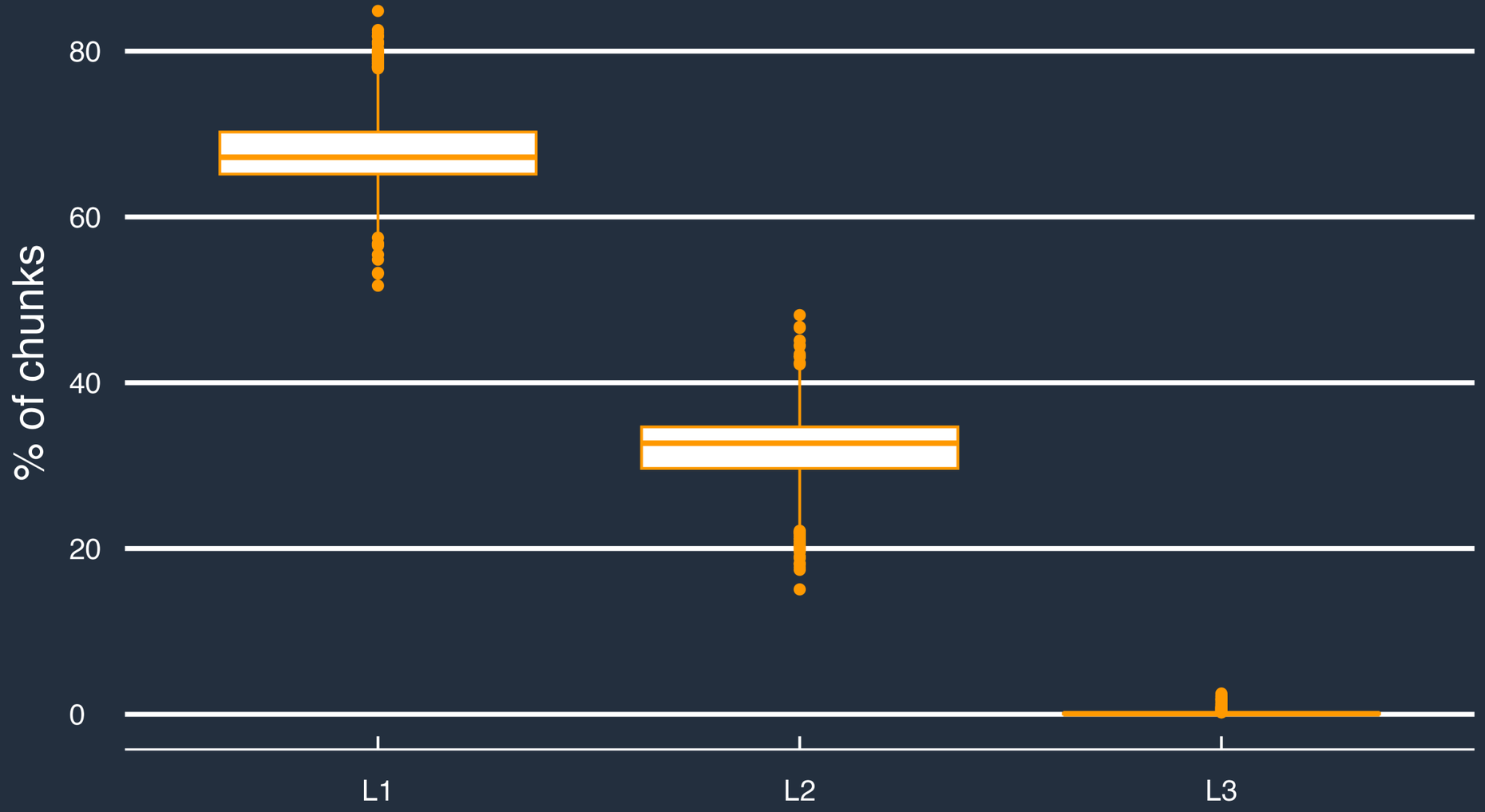


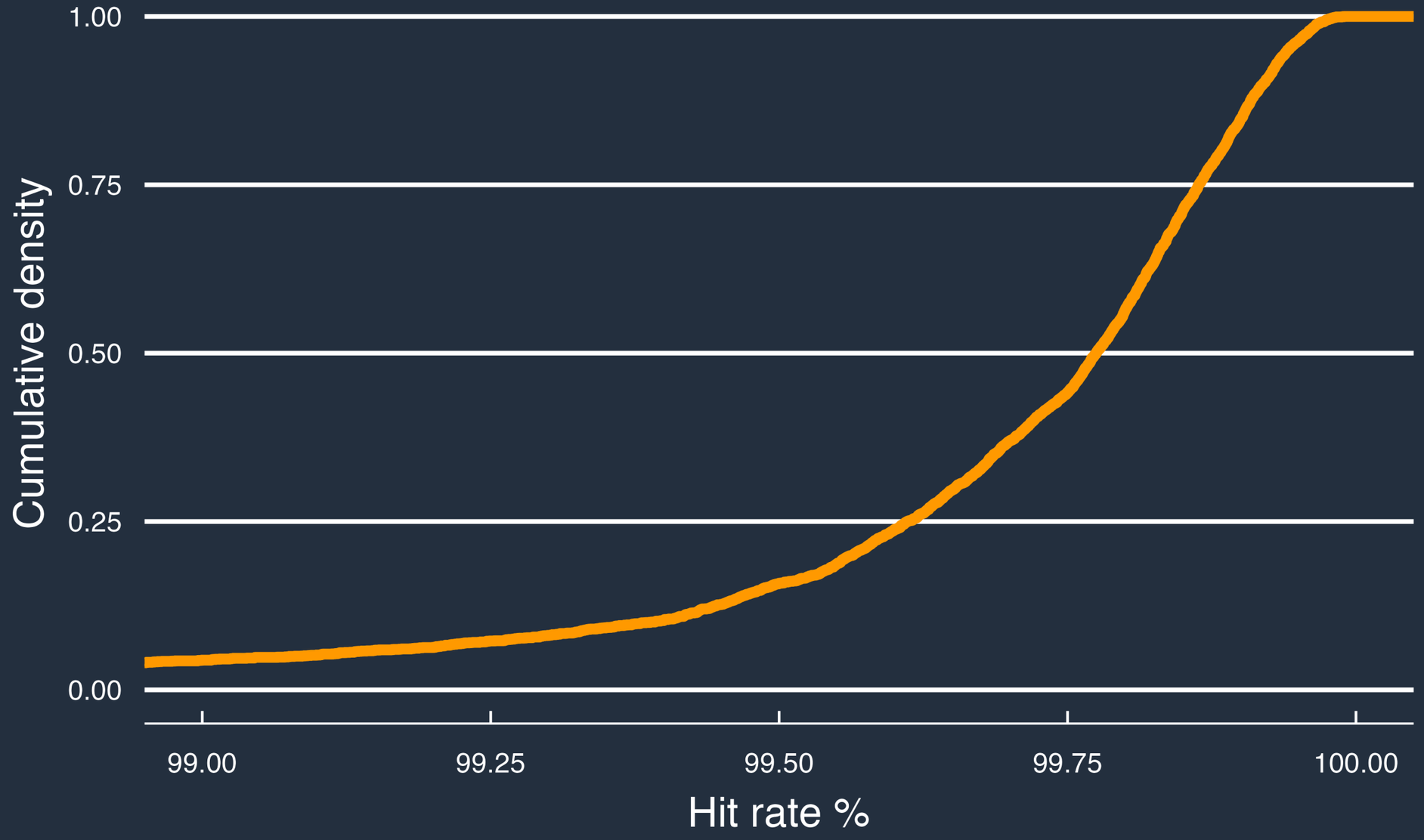
This is fake data, aimed to give the general idea, because I can't share the real numbers.

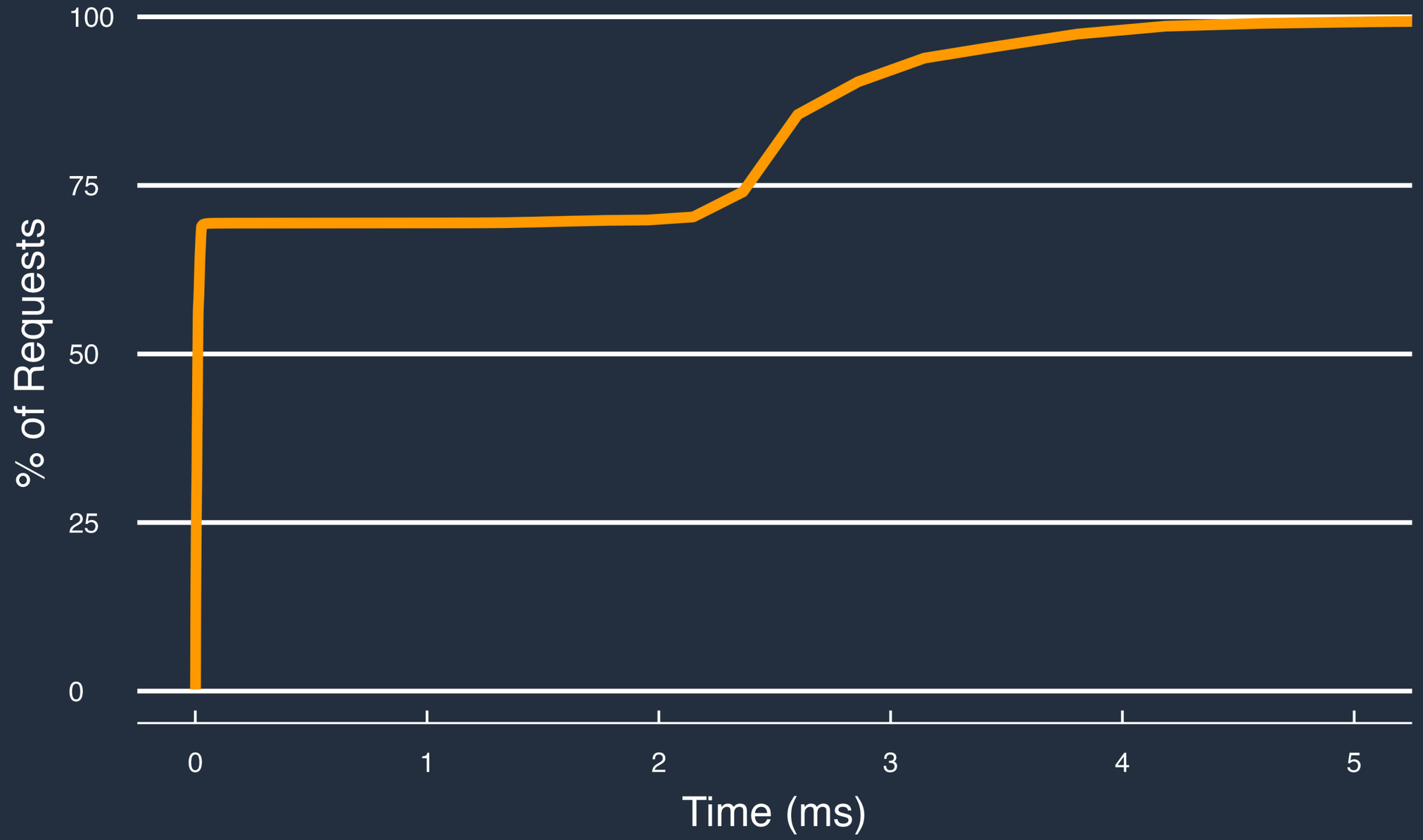


This is fake data, aimed to give the general idea, because I can't share the real numbers.

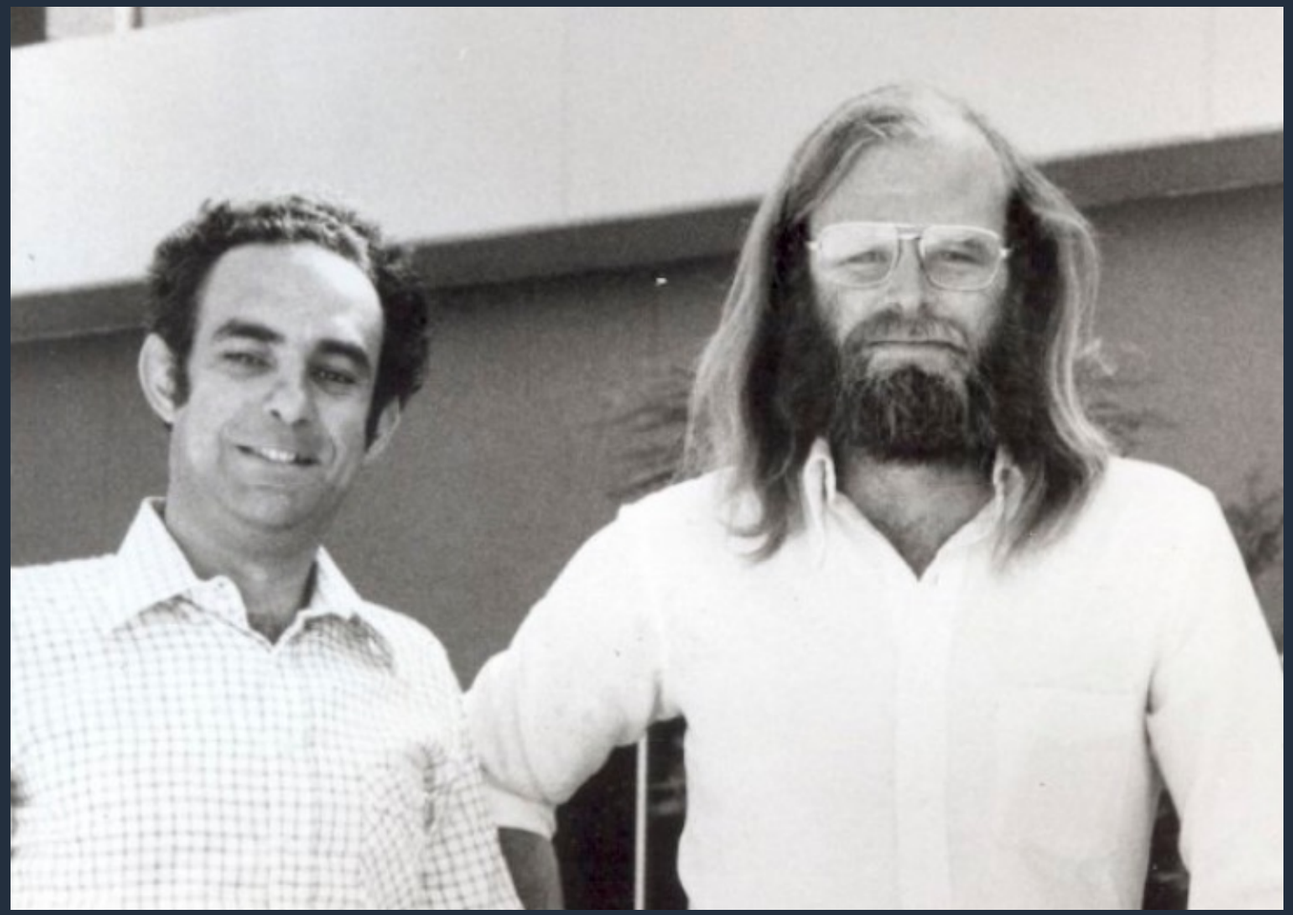








How Big?

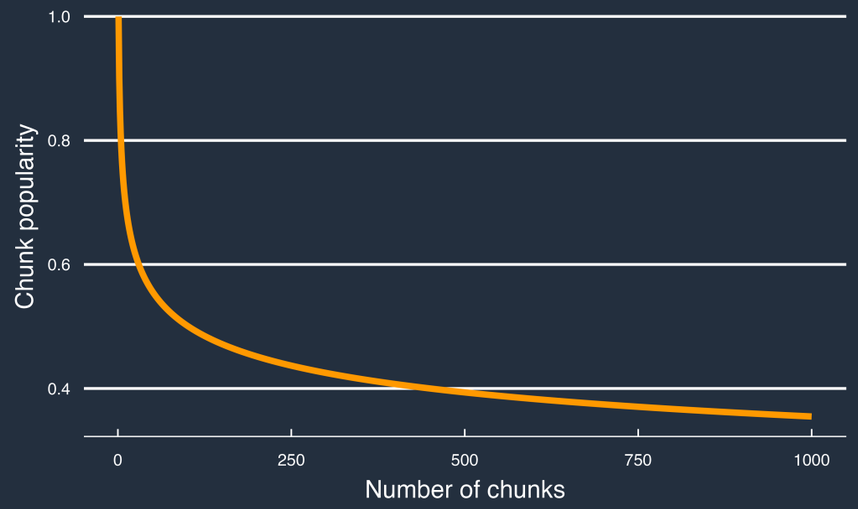


cost of caching = cost of reloading

Jim Gray and Franco Putzolu, IBM Research, 1977 (picture thanks to Pat Helland)

aws

Max(size needed for cost,
size needed for latency)



This is fake data, aimed to give the general idea, because I had no way to measure this directly.

What about failures?

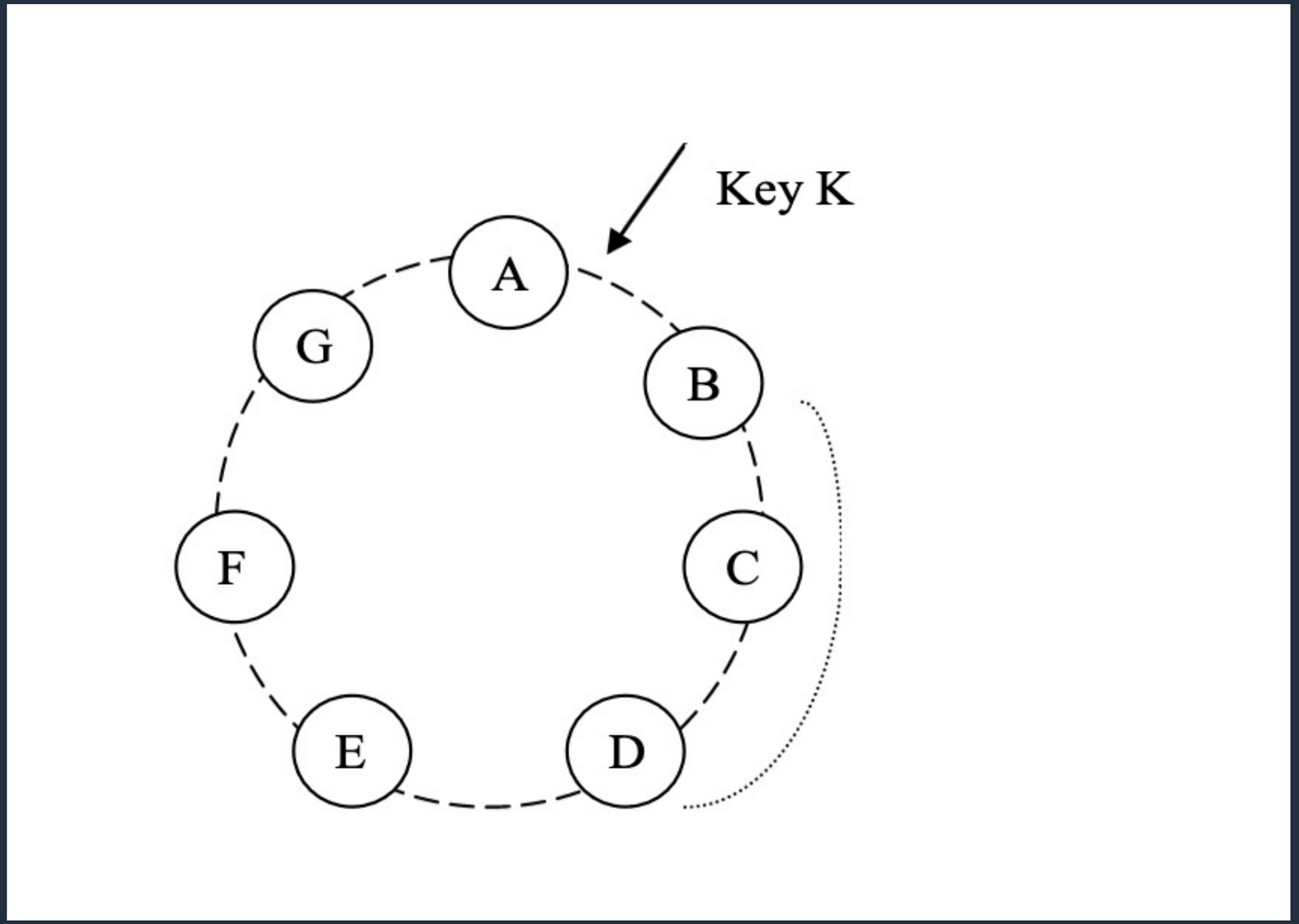
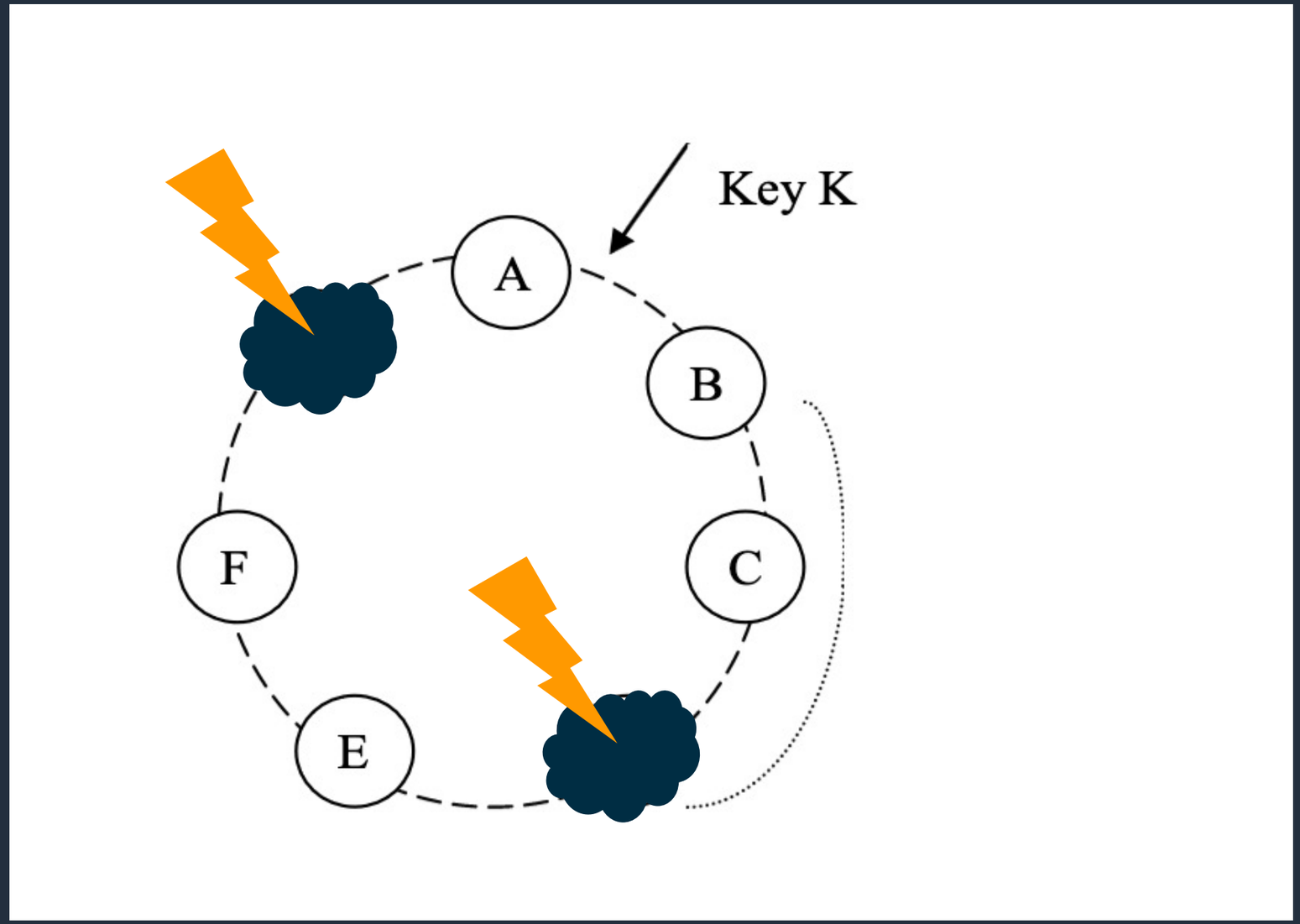
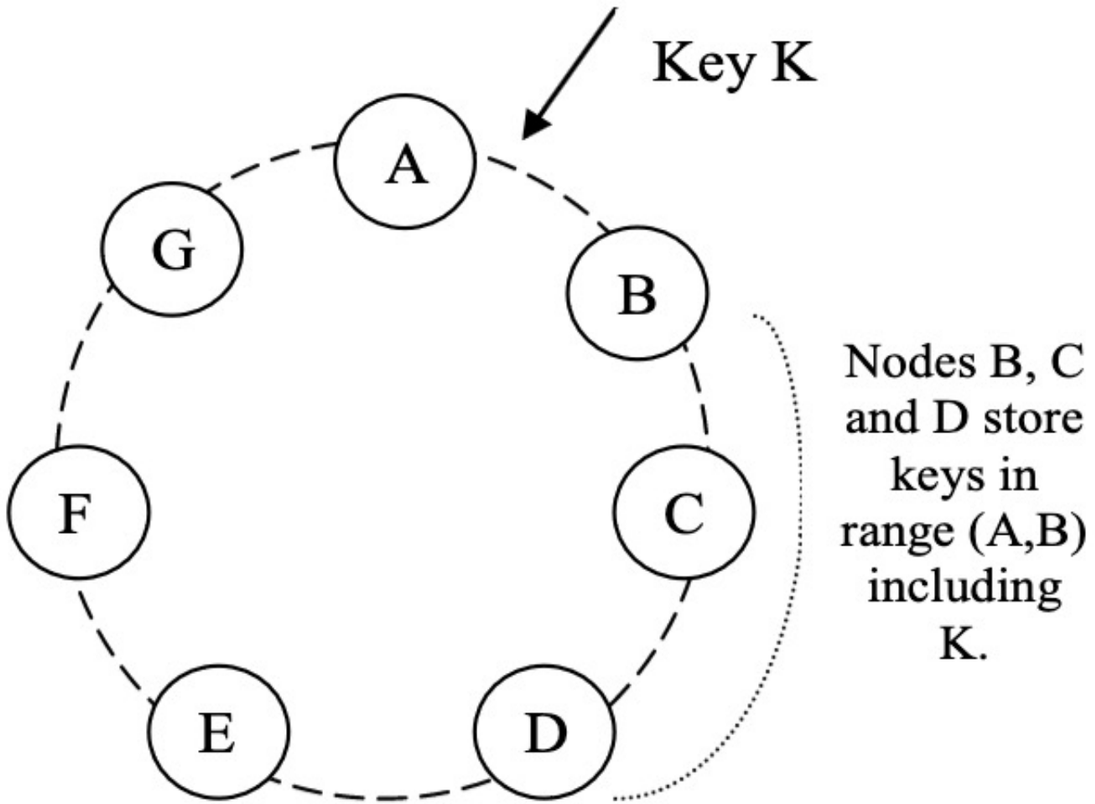


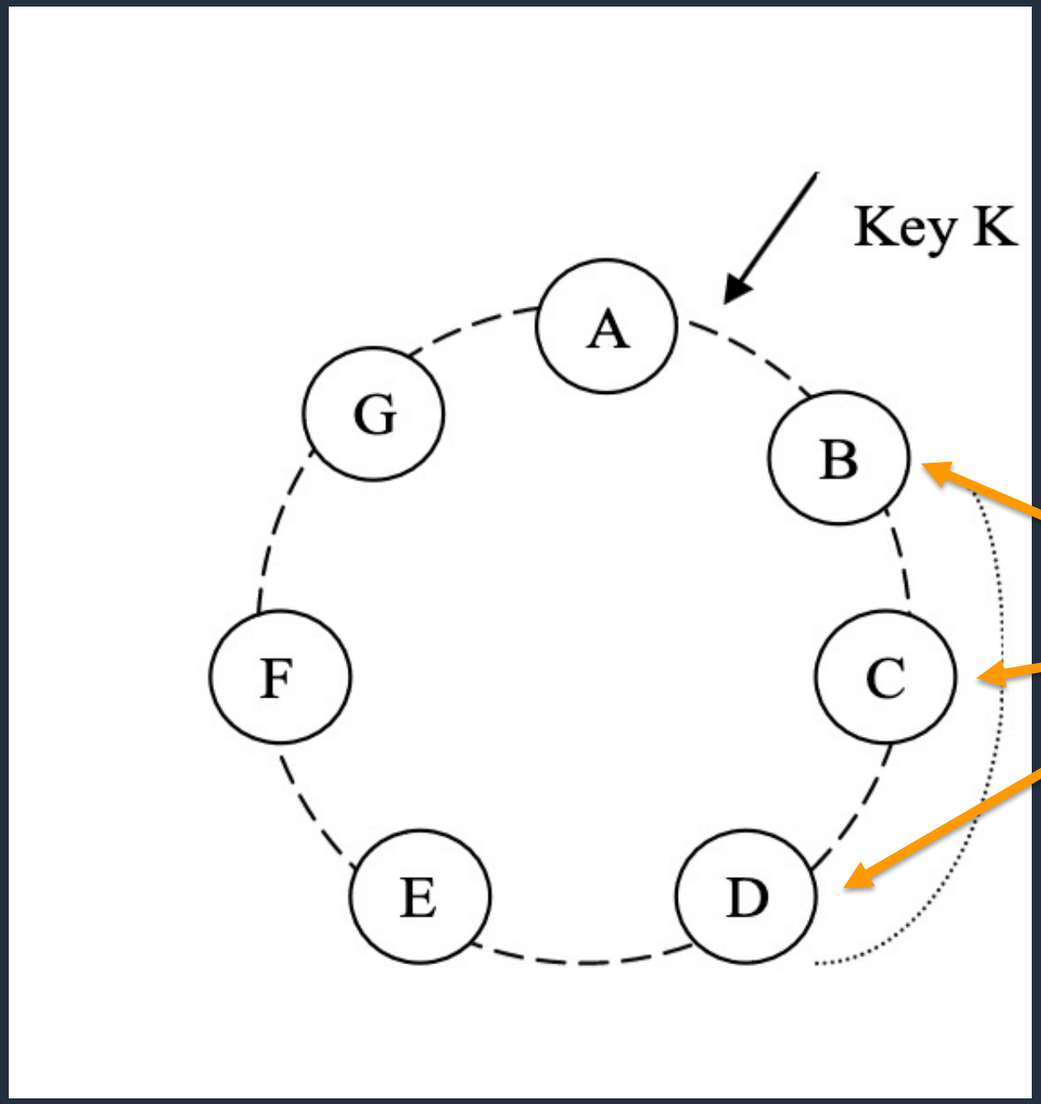
Figure shamelessly borrowed from Decandia et al, *Dynamo: Amazon's Highly Available Key-value Store*, SOSP'07



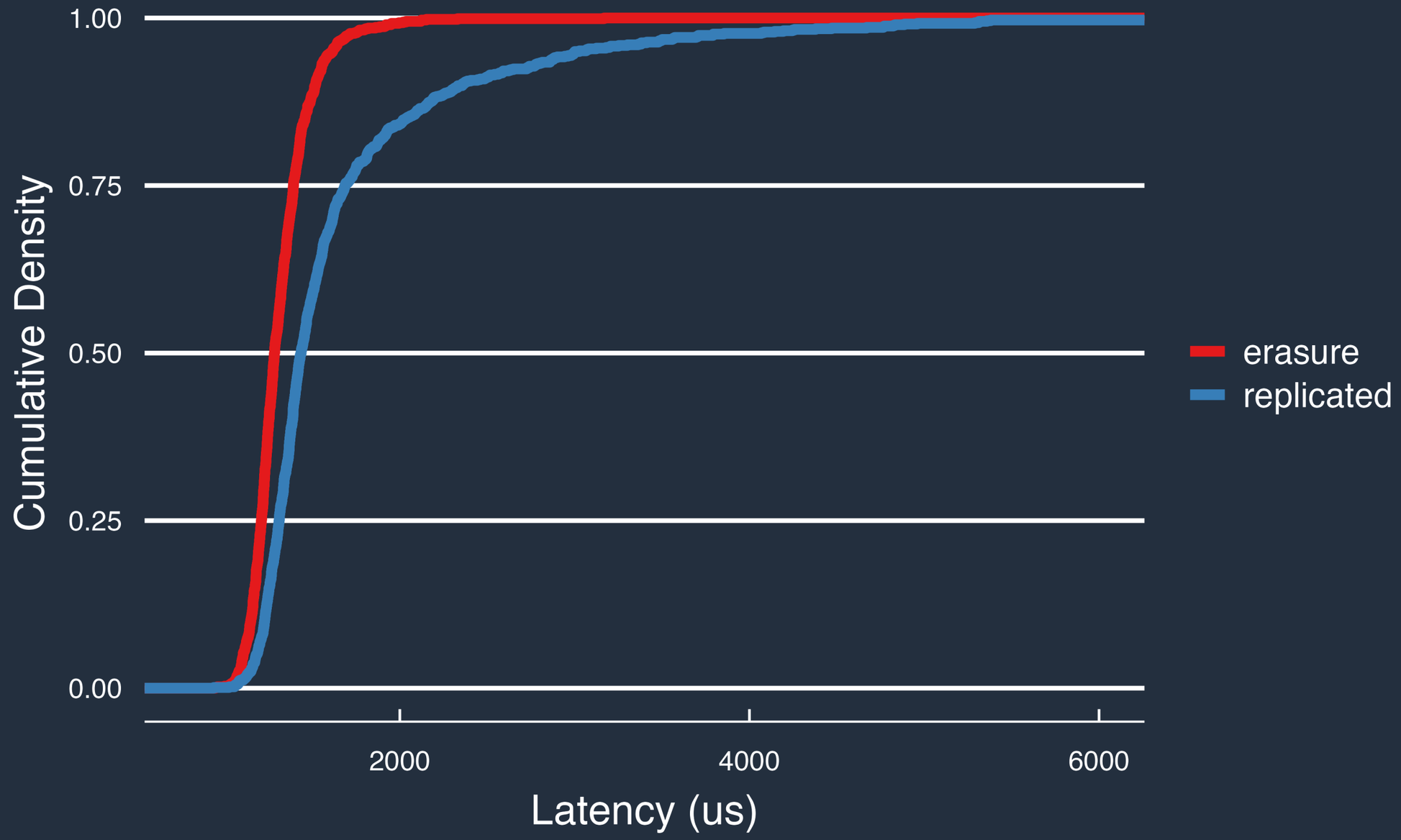


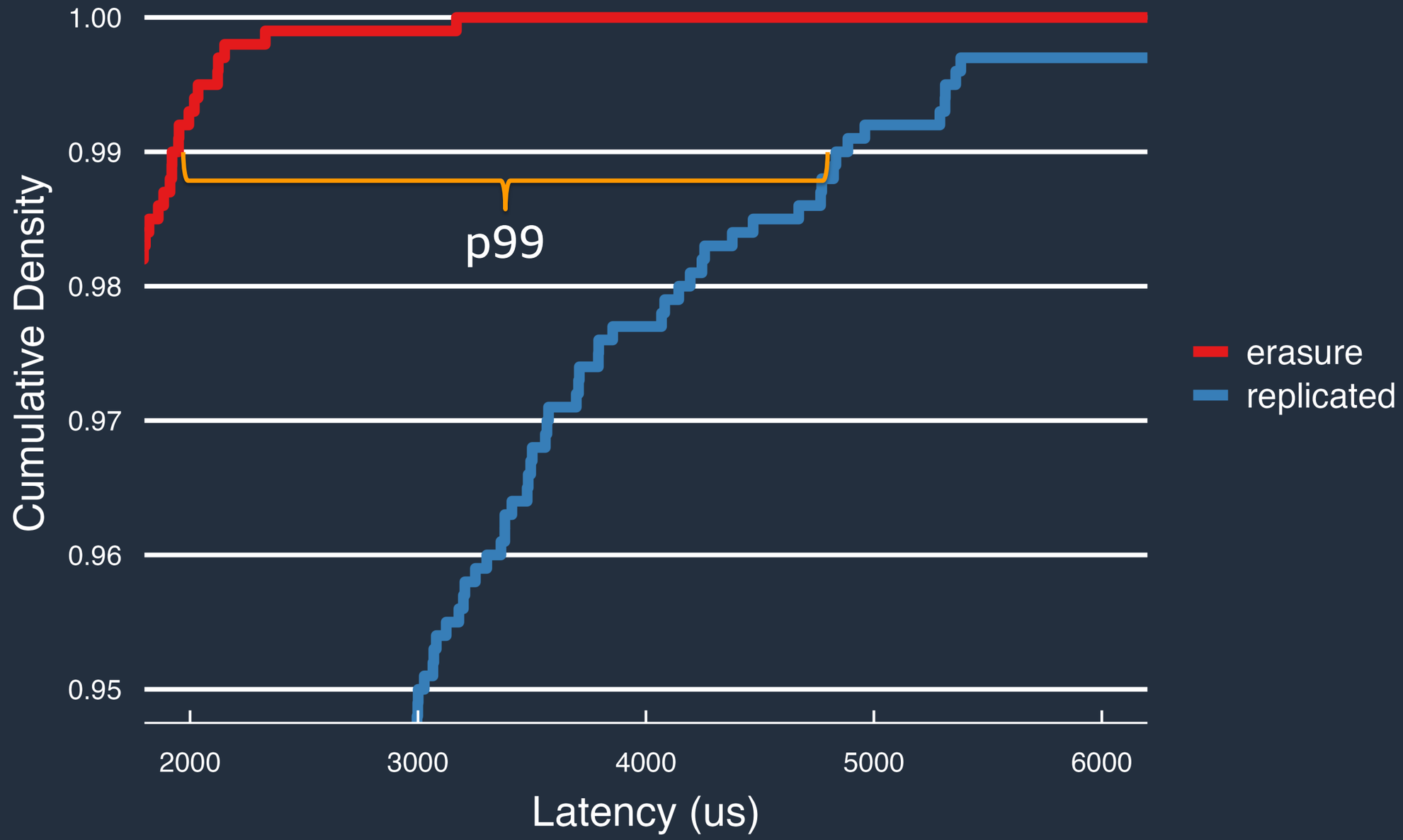
- One Copy
  - Not durable enough.

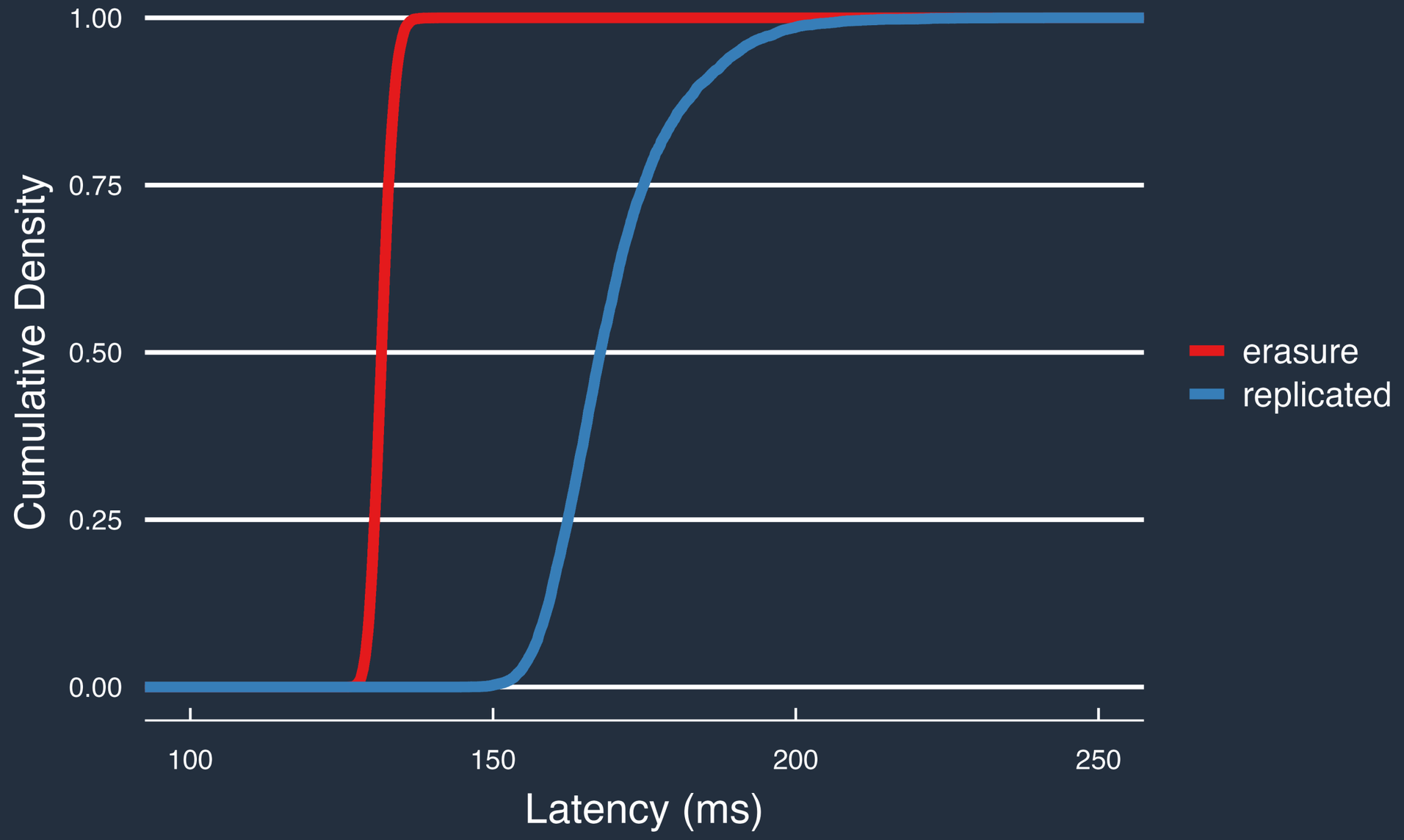
- Two Copies
  - 2x the cost! (or, half the effective cache size)



Erasure Code
“Any 4 of 6”

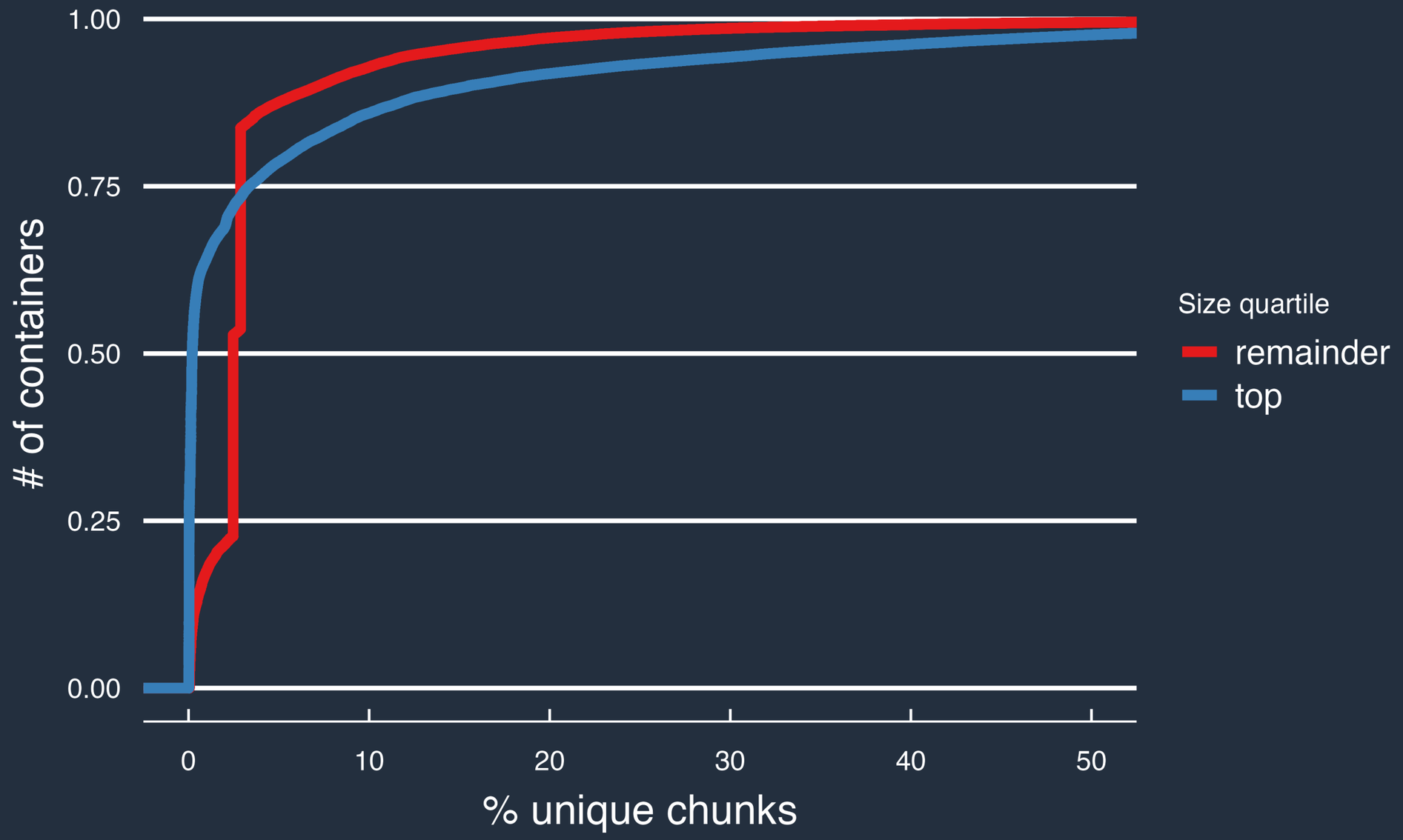






Erasure Coding
vs
Complexity

aws



aws

- Deterministic Flattener
    - Can only put chunks, not read them.
- Worker
    - Can only read chunks for the functions it is running.
- Everybody else
    - Can do basically nothing.

aws

Chunk name =

SHA2(Chunk data)

Chunk name =

SHA2(AES(

))

Chunk ciphertext =

$$AES_{gcm}(\boxed{\text{Chunk data}},\ key = f(\boxed{\text{Chunk data}}),\ iv = [0,\ldots])$$

Douceur et al, “Reclaiming Space from Duplicate Files in a Serverless Distributed File System”, Microsoft, 2002

aws

Chunk name =

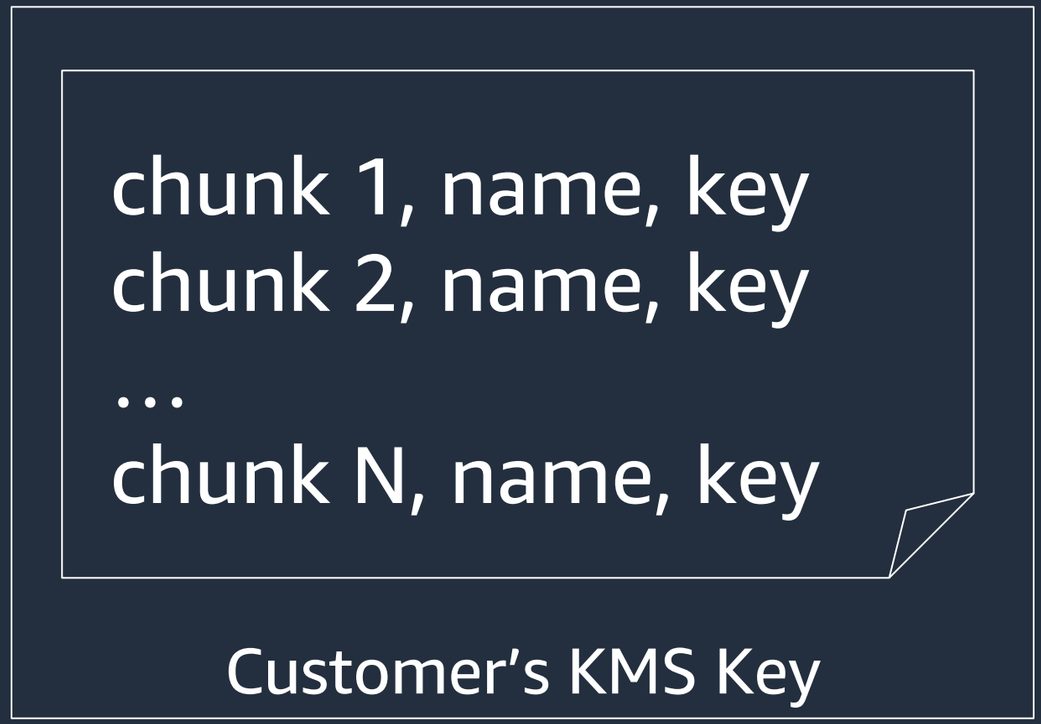
SHA2(chunk ciphertext)

Chunk name =

SHA2(chunk ciphertext)

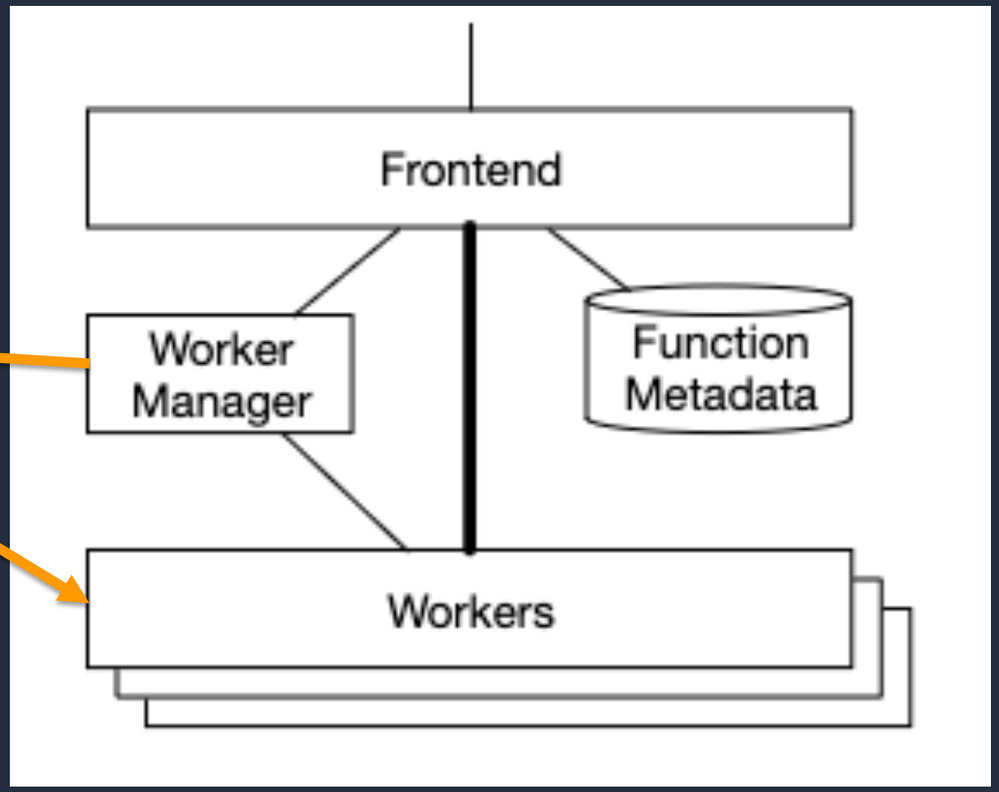
Q: Why not use the GCM *tag*?

aws



chunk 1, name, key
chunk 2, name, key
…
chunk N, name, key

Customer’s KMS Key



- Deterministic Flattener
    - Can only put chunks, not read them.
- Worker
    - Can only read chunks for the functions it is running.
- Everybody else
    - Can do basically nothing.

aws

Questions