Running production-grade logs safely with *Tessera*

Philippe Boneff & Roger Ng - Transparency.dev Summit 2025

Google

Philippe Boneff
phboneff@

Engineer @ Google Open Source Security, TrustFabric
Certificate Transparency Tech Lead

Roger Ng
rogerng@

Engineer @ Google Open Source Security, TrustFabric

What is Tessera?

It’s a library

To build tile-based transparency logs

Open-source

v1.0
Golang

Philosophy

Simplicity

Resilient & available

Flexible, but opinionated

APIs

Write
 `appender.Add($DATA)`

Read
 c2sp.org/tlog-tiles specs

Backends

POSIX
GCP
AWS
S3+MySQL

Performance

From 1 to 18k QPS

*depends on backend and $

Deployments

Sigstore

TesseraCT
 Prod @IPng
 Staging @Google

Reliability

Multiple, 1, (0?!) server

9 common log challenges,

and how Tessera addresses them

1. Computers go down

Goals
- Write endpoints need to remain available
- Two server instances should be able to run simultaneously

Tessera's approach
- Use atomic operations
- Two servers can't update the tree at the same time

AWS **GCP** **S3+MySQL**

Use a database
Aurora / Spanner / MySQL

Lessons learned
not all S3 compliant storage systems are equal!

POSIX

Atomic POSIX operations
Synchronous sequencing + integration

This is a reliability feature, not a performance one

2. Reads and writes are equally important

Goals

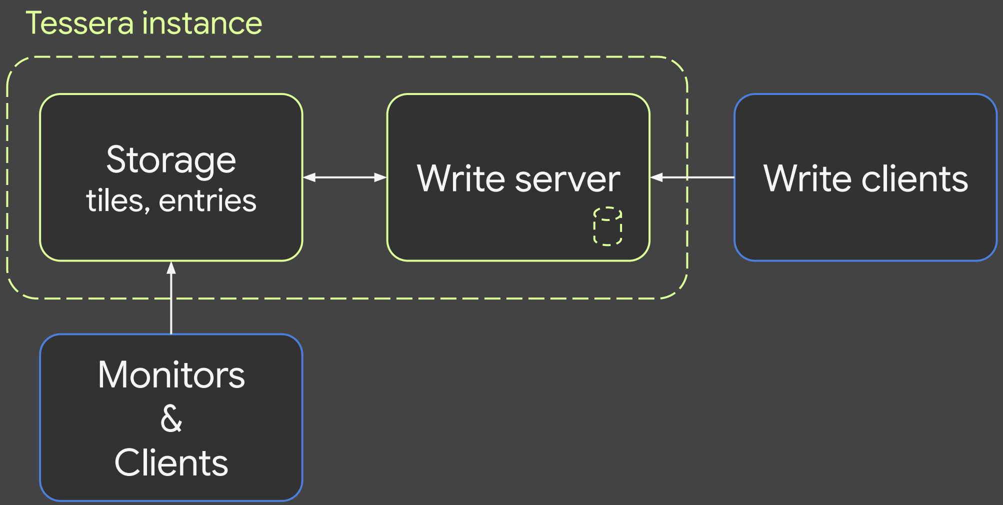
Read and writes should not block each other

Reads should be as cheap as possible

Tessera's approach

Reads don't go through the write server

AWS/GCP writes are staged in a different database



3. Append only, accepting public submissions?!

Goal
Filter the data getting into the load

Tessera's approach
Tessera does not filter data, your server does

Think carefully of your claimant model
↳ only accept what you really need to

3. Append only, accepting public submissions?!

Goal
Filter the data getting into the load

Tessera's approach
Tessera does not filter data, your server does

Think carefully of your claimant model
↳ only accept what you really need to

Examples from TesseraCT:

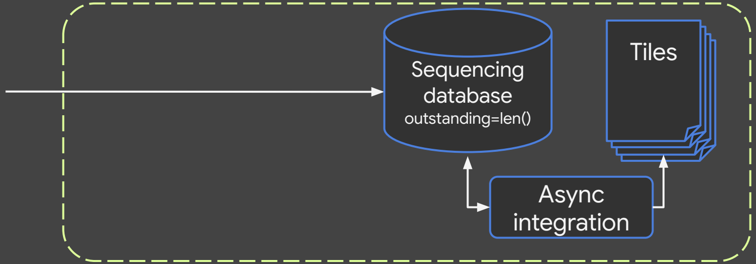
→ Reject submissions that are too old
→ Only accept certificates chaining
to specific roots

4. err_too_many_requests

Goal
Avoid accepting more requests than Tessera can process

Tessera's approach
Push back if too many requests are yet to be integrated

AWS GCP S3+MySQL



4. err_too_many_requests

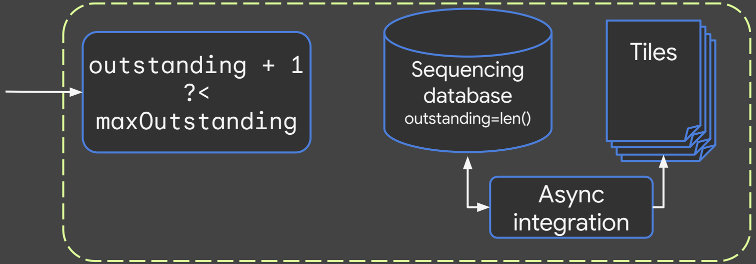
Goal
Avoid accepting more requests than Tessera can process

Tessera's approach
Push back if too many requests are yet to be integrated

AWS GCP S3+MySQL

```
tessera.NewAppendOptions().
 WithPushback(*MaxOutstanding)

appender.Add()
  ↳
```



4. err_too_many_requests

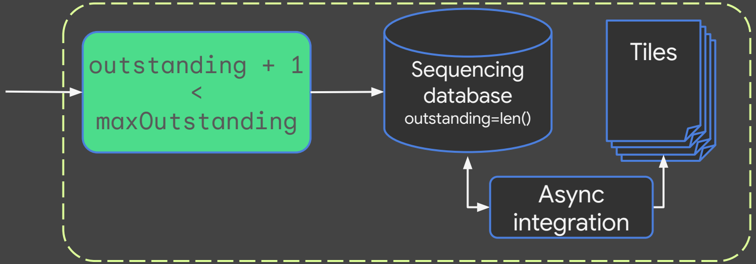
Goal
Avoid accepting more requests than Tessera can process

Tessera's approach
Push back if too many requests are yet to be integrated

AWS GCP S3+MySQL

```
tessera.NewAppendOptions().
 WithPushback(*MaxOutstanding)

appender.Add()
 ↳ tessera.future()
    ↳ index
```



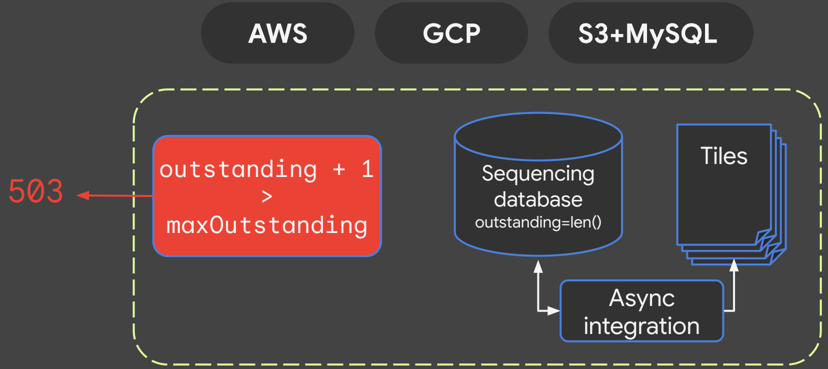
4. err_too_many_requests

Goal
Avoid accepting more requests than Tessera can process

Tessera's approach
Push back if too many requests are yet to be integrated

```
tessera.NewAppendOptions().
 WithPushback(MaxOutstanding)

appender.Add()
 ↳ tessera.ErrPushback
    ↳ 503, HTTP Retry-After
```



5. Identical data can be submitted repeatedly

Goal
Prevent duplicates submissions from spamming the log

Tessera's approach
Best effort, asynchronous deduplication

Easier said than done
For efficiency, logs batch write & read operations
Deduplication is a 1:1 operation, more expensive
↳ Antispam is best effort
Antispam is asynchronous

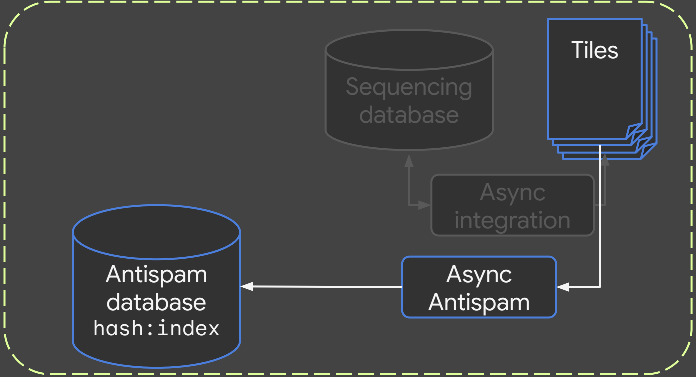
5. Identical data can be submitted repeatedly

Goal
Prevent duplicates submissions from spamming the log

Tessera's approach
Best effort, asynchronous deduplication

Easier said than done
For efficiency, logs batch write & read operations
Deduplication is a 1:1 operation, more expensive
↳ Antispam is best effort
Antispam is asynchronous

```
tessera.NewAppendOptions().
 WithAntispam(antispamCacheSize, antispam)
```



5. Identical data can be submitted repeatedly

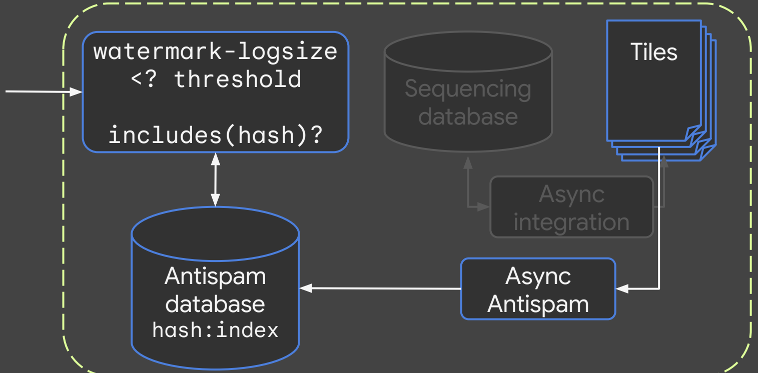
Goal
Prevent duplicates submissions from spamming the log

Tessera's approach
Best effort, asynchronous deduplication

Easier said than done
For efficiency, logs batch write & read operations
Deduplication is a 1:1 operation, more expensive
↳ Antispam is best effort
Antispam is asynchronous

```
tessera.NewAppendOptions().
 WithAntispam(antispamCacheSize, antispam)

 appender.Add()
  ↳
```



5. Identical data can be submitted repeatedly

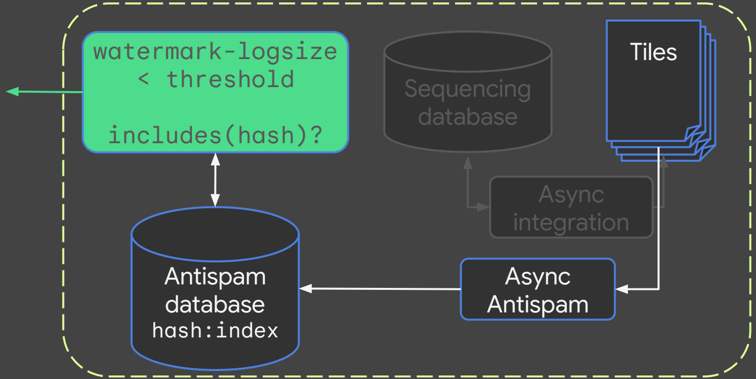
Goal
Prevent duplicates submissions from spamming the log

Tessera's approach
Best effort, asynchronous deduplication

Easier said than done
For efficiency, logs batch write & read operations
Deduplication is a 1:1 operation, more expensive
↳ Antispam is best effort
Antispam is asynchronous

```
tessera.NewAppendOptions().
 WithAntispam(antispamCacheSize, antispam)

 appender.Add()
  ↳ index from Antispam DB
```



5. Identical data can be submitted repeatedly

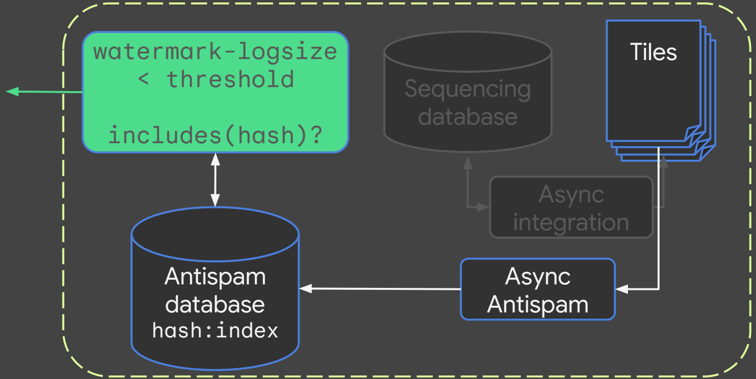
Goal
Prevent duplicates submissions from spamming the log

Tessera's approach
Best effort, asynchronous deduplication

Easier said than done
For efficiency, logs batch write & read operations
Deduplication is a 1:1 operation, more expensive
↳ Antispam is best effort
Antispam is asynchronous

```
tessera.NewAppendOptions().
 WithAntispam(antispamCacheSize, antispam)

 appender.Add()
  ↳ index from Antispam DB or not
```



5. Identical data can be submitted repeatedly

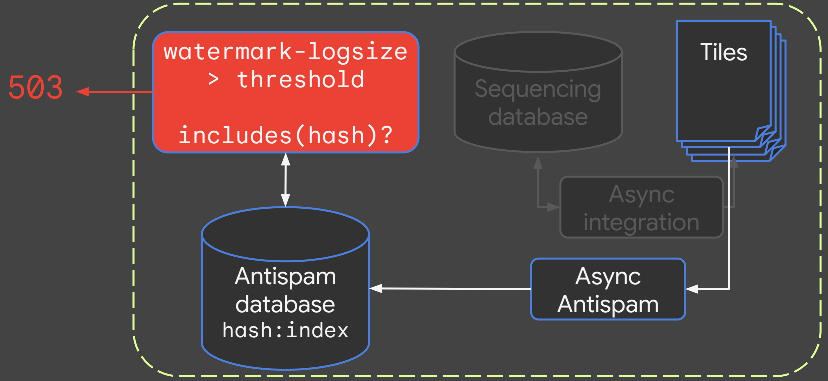
Goal
Prevent duplicates submissions from spamming the log

Tessera's approach
Best effort, asynchronous deduplication

Easier said than done
For efficiency, logs batch write & read operations
Deduplication is a 1:1 operation, more expensive
↳ Antispam is best-effort
Antispam is asynchronous

```
tessera.NewAppendOptions().
 WithAntispam(antispamCacheSize, antispam)

 appender.Add()
  ↳ tessera.ErrPushback
    ↳ 503, HTTP Retry-After
```



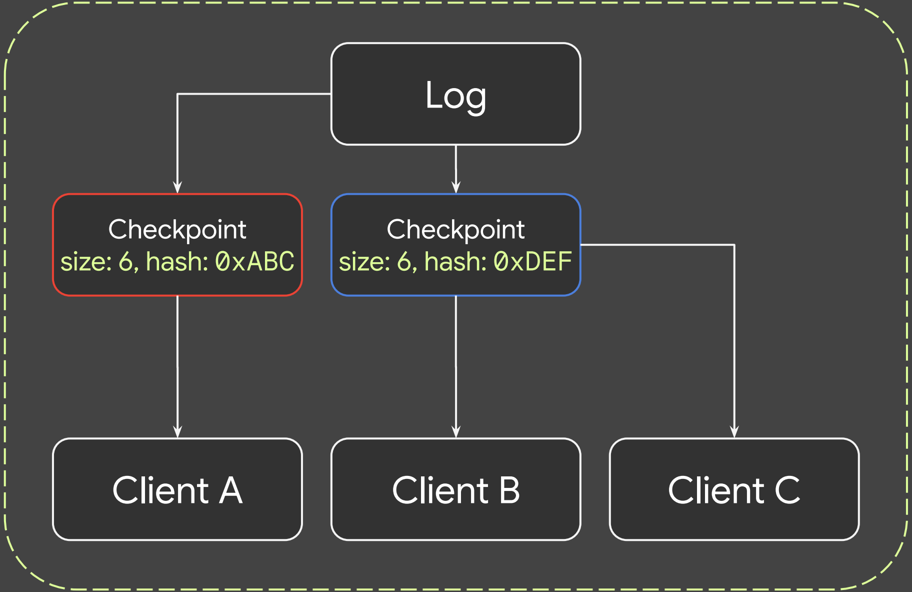
6. Clients need to agree on checkpoints

Goal
Prevent the log from presenting split views

Tessera's approach
Get witnesses to countersign checkpoints

What is a split view attack?

A malicious log can provide different views with verifiable signatures to different clients at the same time.



6. Clients need to agree on checkpoints

Goal
Prevent the log from presenting split views

Tessera's approach
Get witnesses to countersign checkpoints

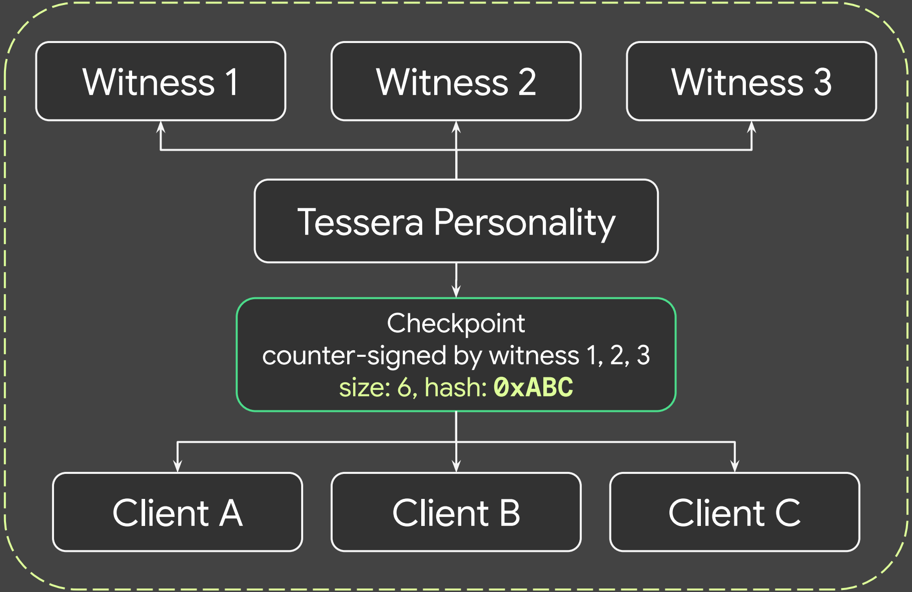
Checkpoints can be ***countersigned by witness(es)*** compatible with the C2SP Witness Protocol before publishing them.

```
tessera.NewAppendOptions().
    WithWitnesses(witnesses, Opts)
```

Option: Fail-open if witnesses are unavailable*

```
tessera.WitnessOptions{FailOpen: true}
```

*Eventually, once the witness network is more robust, this will fail closed by default



7. Commitment is hard

Goal
Ensure the log commits to data before returning it

Tessera's approach
Only return committed index, optionally wait for publication

Signed Certificate Timestamps (SCTs)

aka. promises of inclusion

are *BAD*

Don’t design new systems using them

7. Commitment is hard

Goal
Ensure the log commits to data before returning it

Tessera's approach
Only return committed index, optionally wait for publication

7. Commitment is hard

Goal
Ensure the log commits to data before returning it

Tessera's approach
Only return committed index, optionally wait for publication

POSIX *synchronous*
The entry is integrated when the index is returned

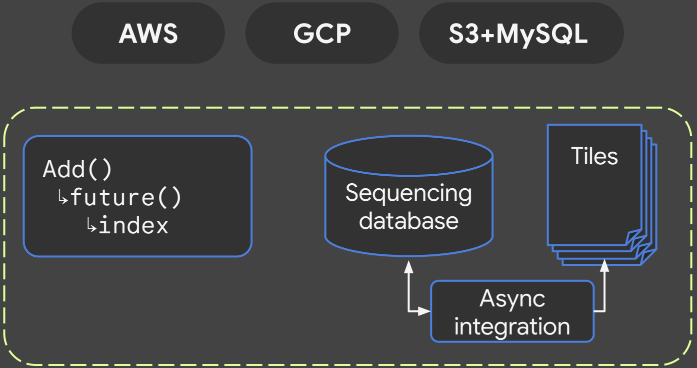
7. Commitment is hard

Goal
Ensure the log commits to data before returning it

Tessera's approach
Only return committed index, optionally wait for publication

POSIX *synchronous*
The entry is integrated when the index is returned

AWS / GCP / S3+MySQL *asynchronous*
The entry will eventually be integrated at that index
Within a few seconds



```
Add()
 ↳future()
   ↳index
```

7. Commitment is hard

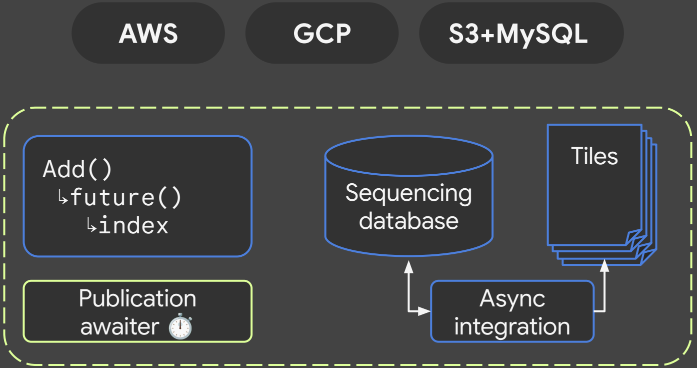
Goal
Ensure the log commits to data before returning it

Tessera's approach
Only return committed index, optionally wait for publication

POSIX *synchronous*
The entry is integrated when the index is returned

AWS / GCP / S3+MySQL *asynchronous*
The entry will eventually be integrated at that index
Within a few seconds

Wait for integration AND a checkpoint
↳ `tessera.PublicationAwaiter.Await()`



8. Humans make mistakes

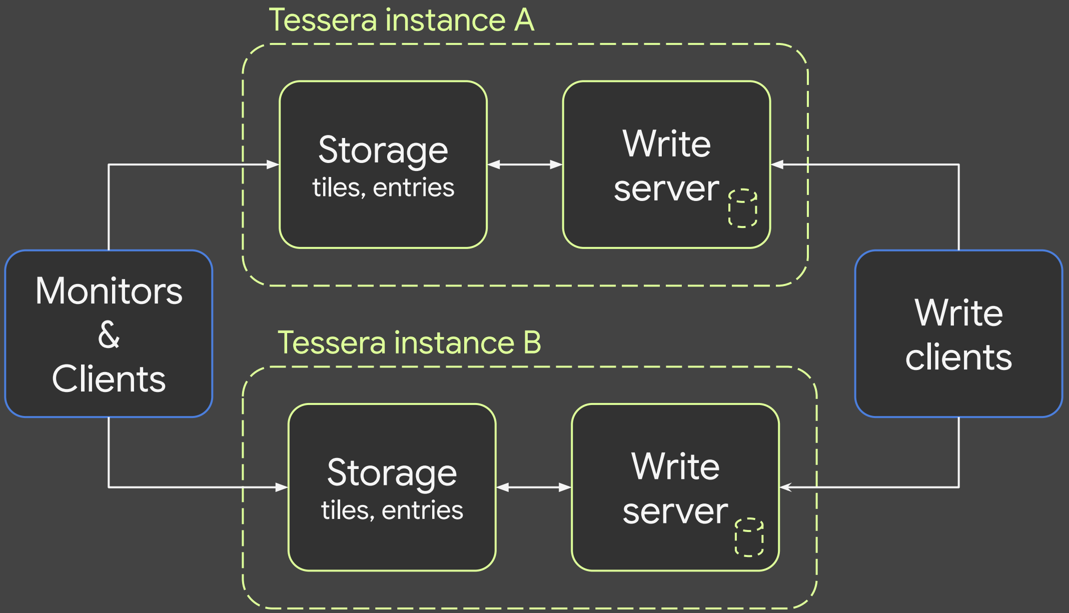
Goal
No system outage or human operations should result in data loss

Tessera's approach
No multi-tenancy & persisted data when `future()` is resolved

Effortless resume with no data loss

Tessera will continue to process the "in-flight" entries (i.e. the entries with sequence numbers assigned, but which are not yet integrated into the log) after restart.

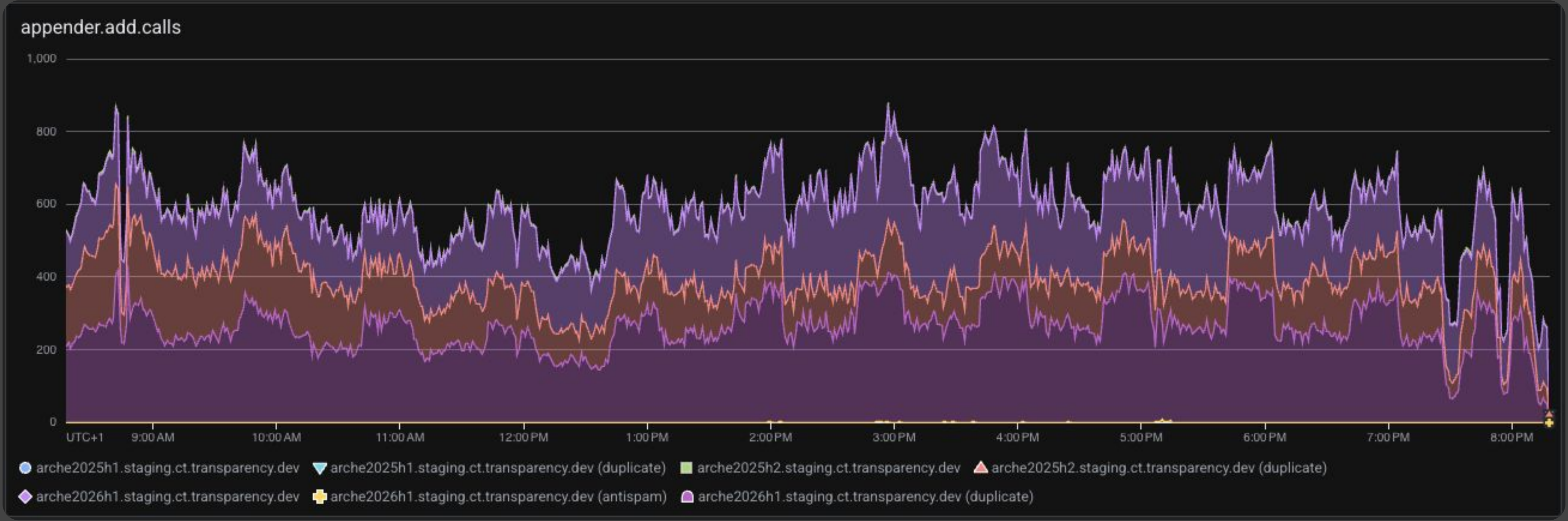
One log = dedicated {servers, log storage, state}



9. Visibility is key

Goal
Knowing what's going on in the log

Tessera's approach
Metrics from OpenTelemetry



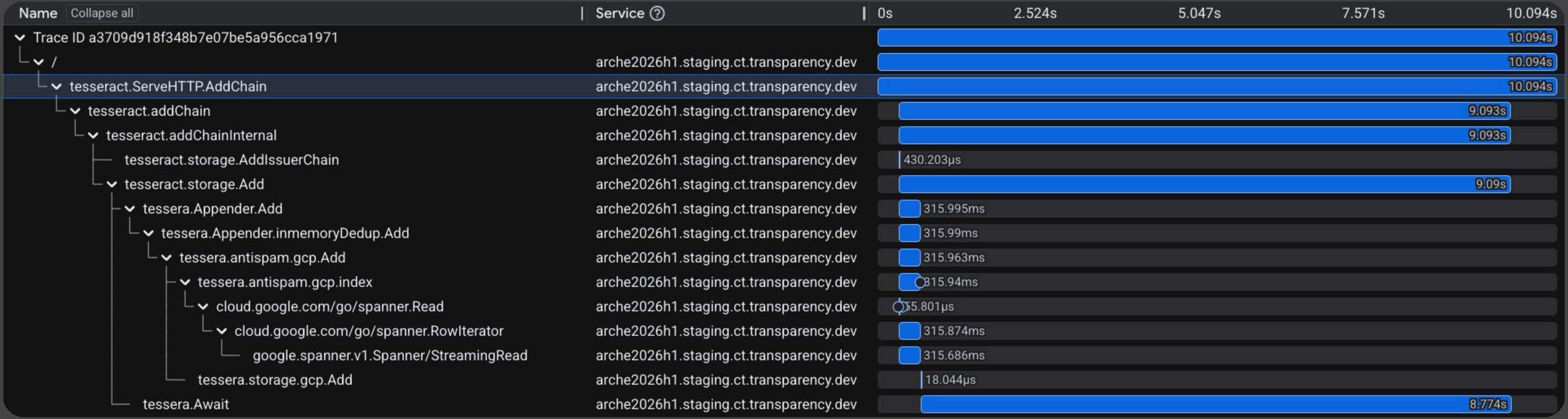
9. Visibility is key

Goal

Knowing what's going on in the log

Tessera's approach

Metrics from OpenTelemetry



Name	Service	0s 2.524s 5.047s 7.571s 10.094s
Trace ID a3709d918f348b7e07be5a956cca1971		10.094s
/	arche2026h1.staging.ct.transparency.dev	10.094s
tesseract.ServeHTTP.AddChain	arche2026h1.staging.ct.transparency.dev	10.094s
tesseract.addChain	arche2026h1.staging.ct.transparency.dev	9.093s
tesseract.addChainInternal	arche2026h1.staging.ct.transparency.dev	9.093s
tesseract.storage.AddIssuerChain	arche2026h1.staging.ct.transparency.dev	430.203µs
tesseract.storage.Add	arche2026h1.staging.ct.transparency.dev	9.09s
tessera.Appender.Add	arche2026h1.staging.ct.transparency.dev	315.995ms
tessera.Appender.inmemoryDedup.Add	arche2026h1.staging.ct.transparency.dev	315.99ms
tessera.antispam.gcp.Add	arche2026h1.staging.ct.transparency.dev	315.963ms
tessera.antispam.gcp.index	arche2026h1.staging.ct.transparency.dev	315.94ms
cloud.google.com/go/spanner.Read	arche2026h1.staging.ct.transparency.dev	55.801µs
cloud.google.com/go/spanner.RowIterator	arche2026h1.staging.ct.transparency.dev	315.874ms
google.spanner.v1.Spanner/StreamingRead	arche2026h1.staging.ct.transparency.dev	315.686ms
tessera.storage.gcp.Add	arche2026h1.staging.ct.transparency.dev	18.044µs
tessera.Await	arche2026h1.staging.ct.transparency.dev	8.774s

Questions?

github.com/transparency-dev

github.com/transparency-dev/tessera

github.com/transparency-dev/tesseract

transparency.dev/slack

Google