Envoy internals deep dive

Matt Klein / @mattklein123, Software Engineer @Lyft

Agenda

- Envoy design goals
- Architecture overview
- Threading model
- Hot restart
- Stats
- Q&A

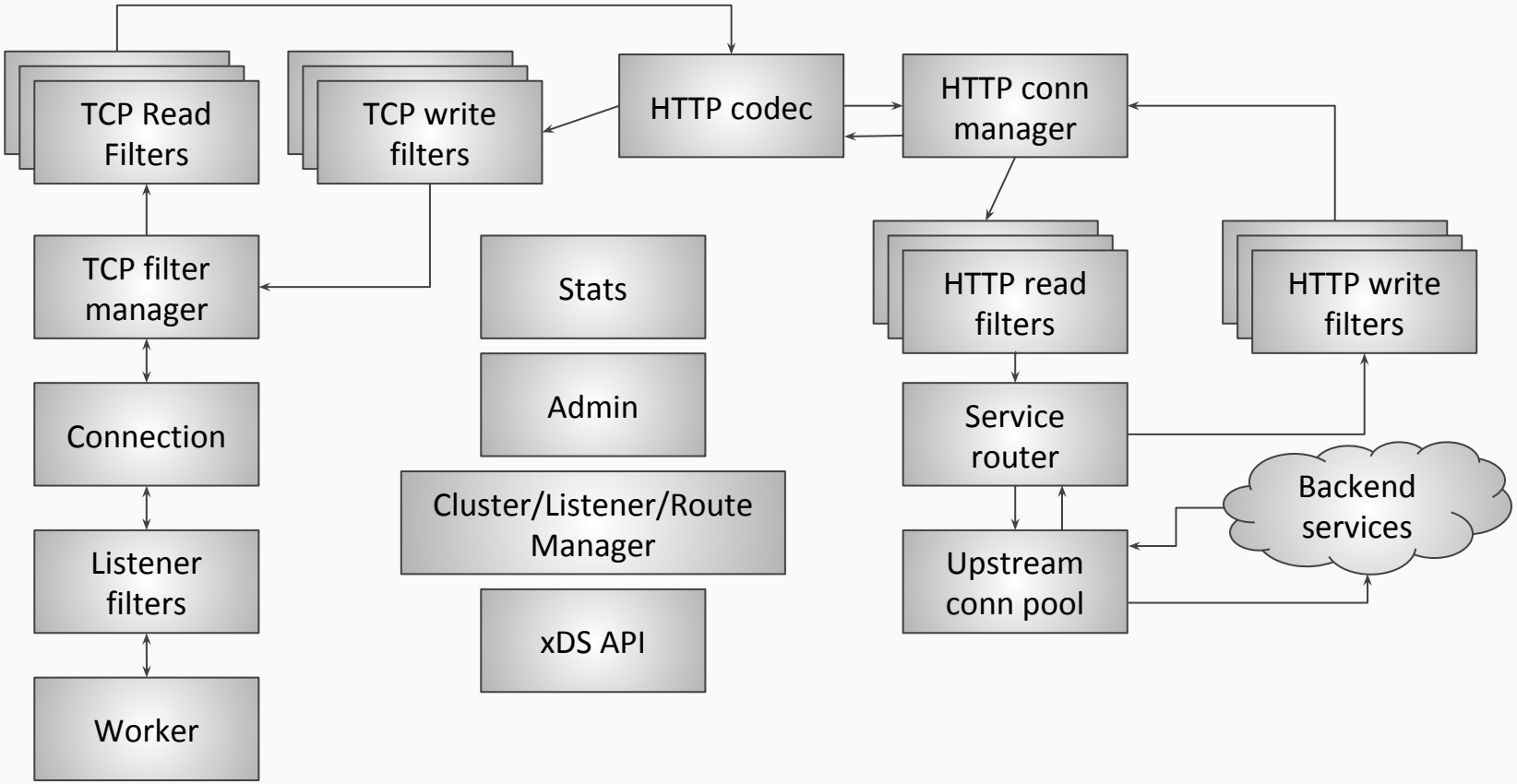
What is Envoy?

The network should be transparent to applications. When network and application problems do occur it should be easy to determine the source of the problem.

Envoy design goals

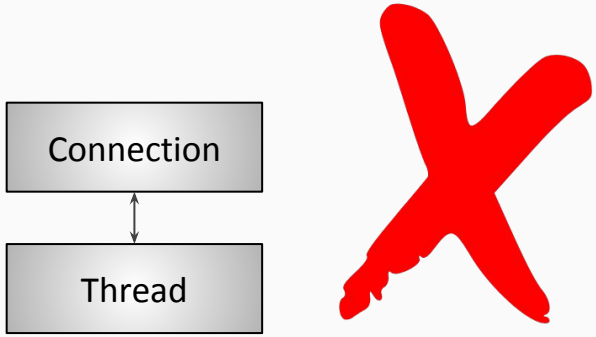
- Out of process architecture
- Low latency, high performance, dev productivity
- L3/L4 filter architecture
- HTTP L7 filter architecture
- HTTP/2 first
- Service/config discovery
- Active/passive health checking
- Advanced load balancing
- Best in class observability
- Edge proxy
- Hot restart

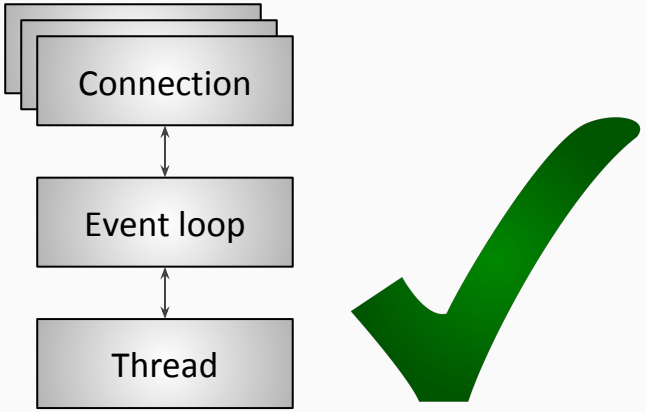
Envoy architecture diagram



Envoy threading model (c10k)

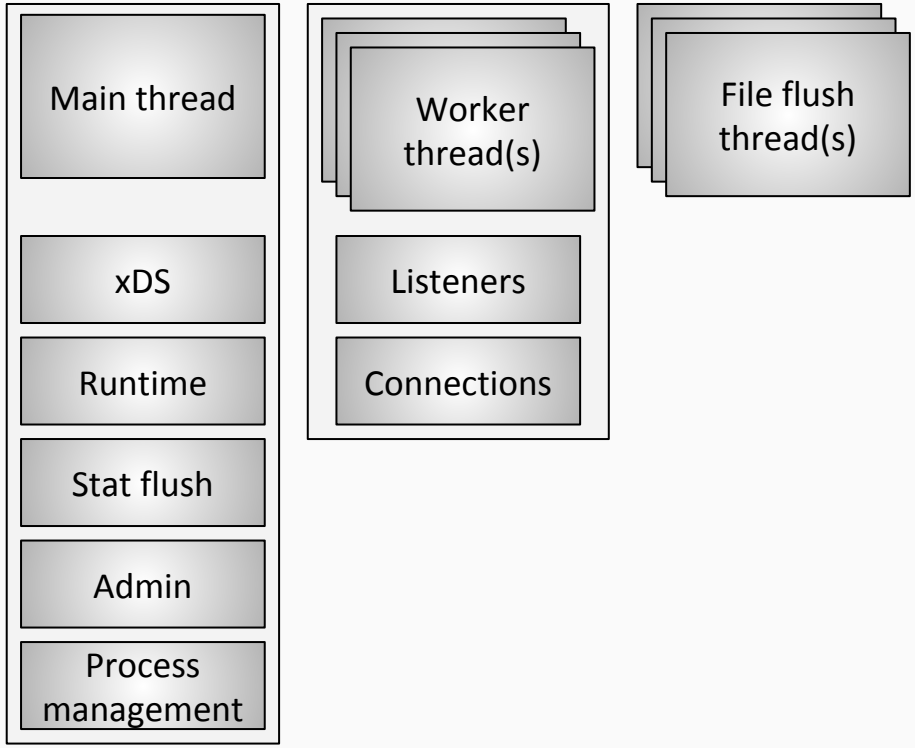
- Connection per thread does not scale
- Scaling requires many connections per thread: “c10k”
- Requires *asynchronous* programming paradigms: harder





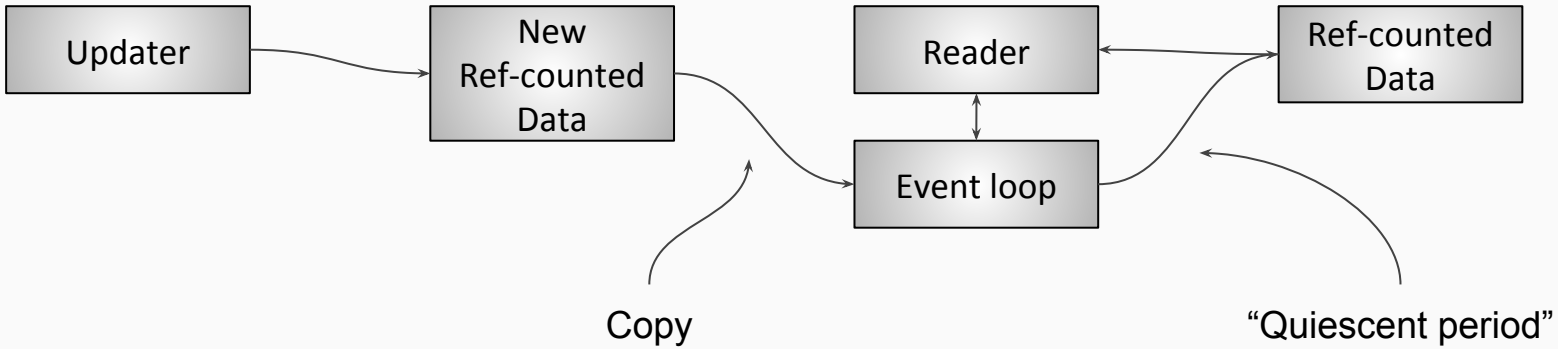
Envoy threading model (overview)

- Main thread handles non-data plane misc tasks
- Worker thread(s) **embarrassingly parallel** and handle listeners, connections, and proxying
- File flush threads avoid blocking
- Designed to be **100% non-blocking**
- Designed to scale to **massive parallelism** (# of HW threads)



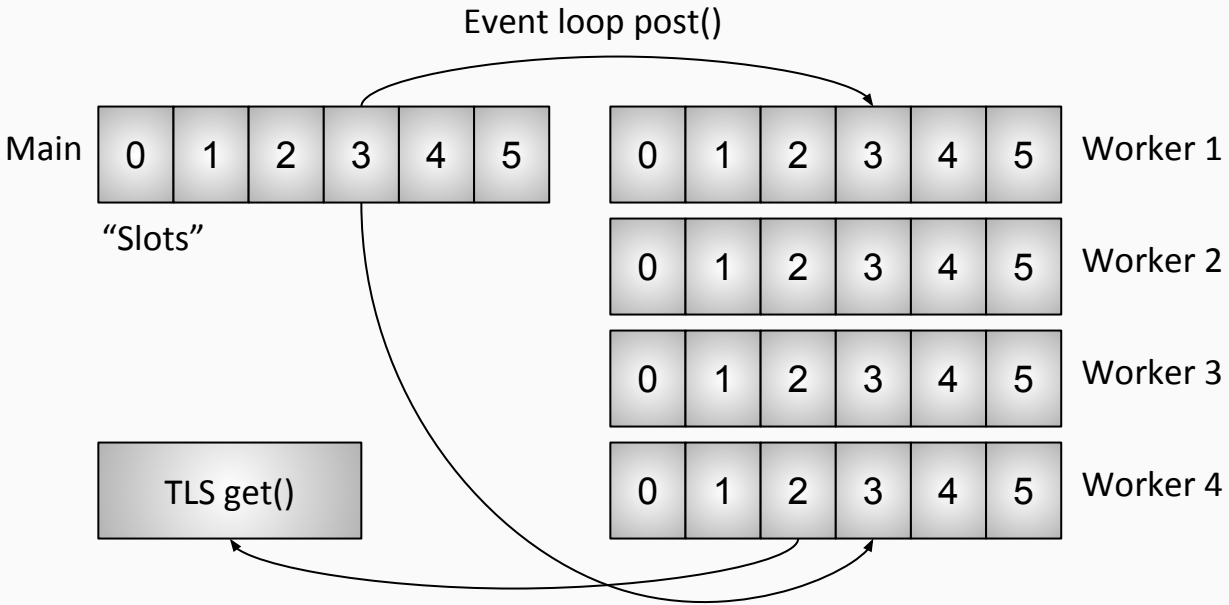
Envoy threading model (RCU)

- **RCU** = Read-Copy-Update
- Synchronization primitive heavily used in the Linux kernel
- Scales extremely well for R/W locking that is read heavy



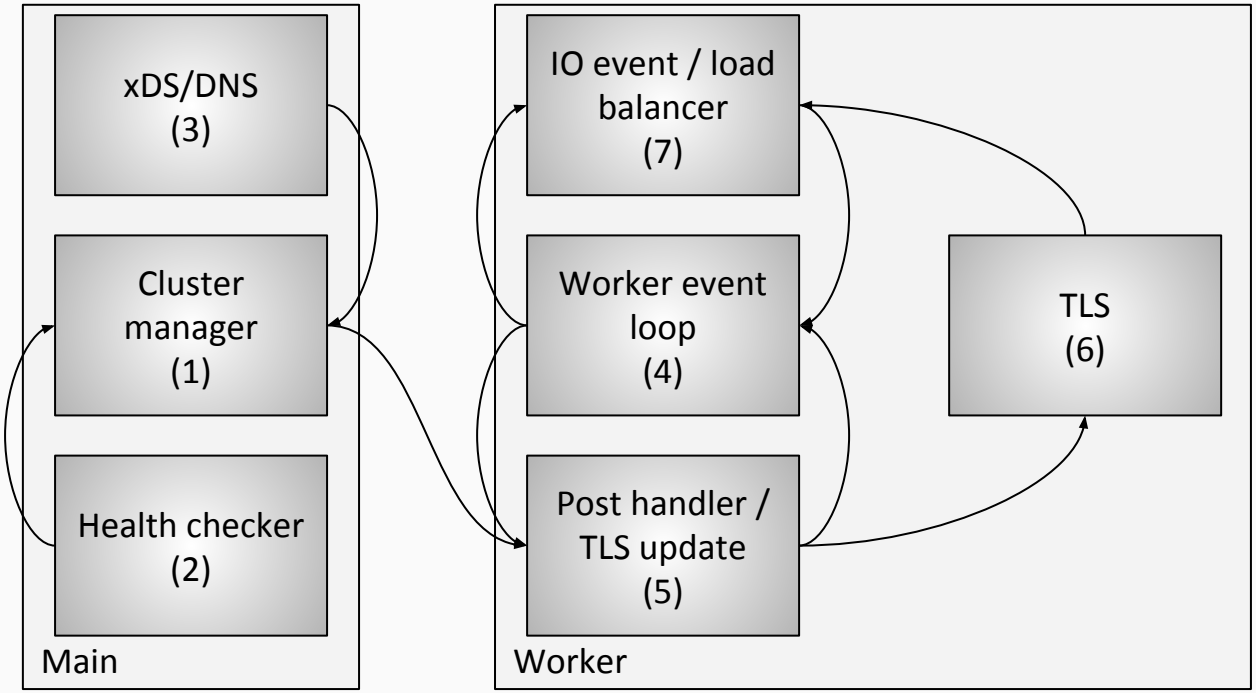
Envoy threading model (TLS and RCU)

- **TLS** = Thread Local Storage
- TLS slots can be allocated dynamically by objects
- RCU is used to post shared read-only data from the main thread to workers



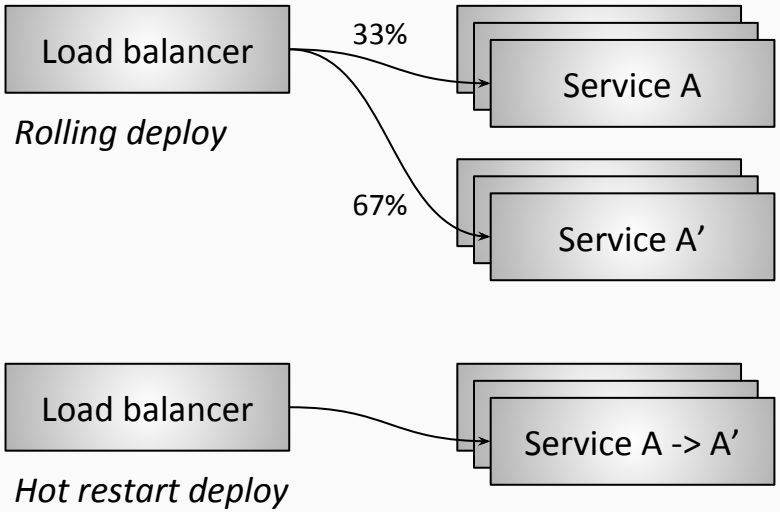
Envoy threading model (cluster updates example)

- Complete example of TLS and RCU for cluster updates



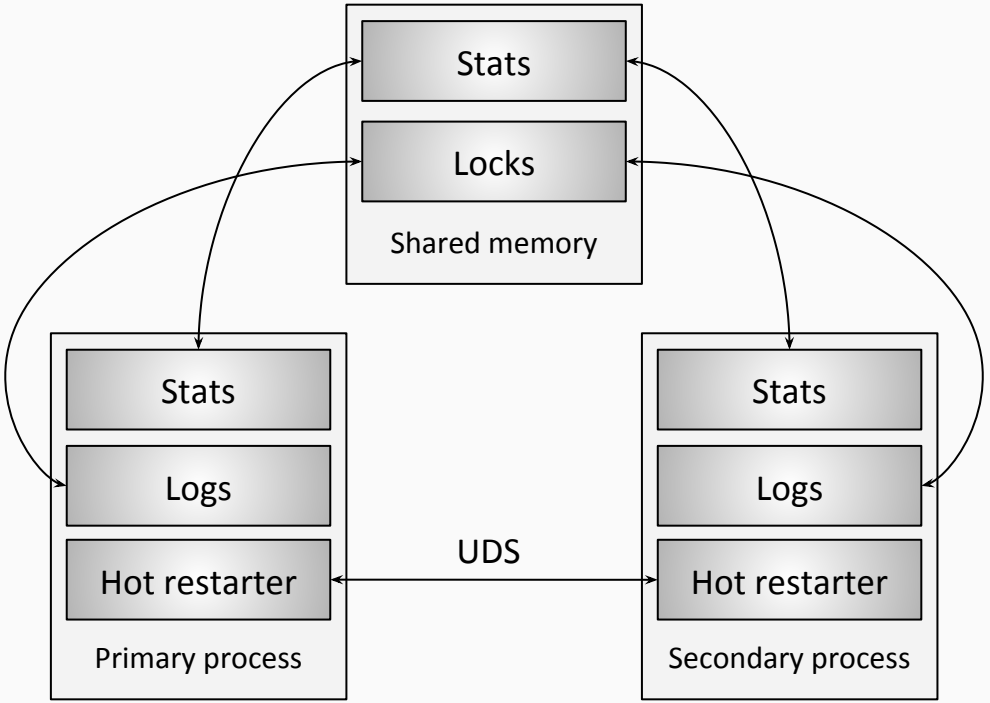
Envoy hot restart (overview)

- Full binary reload without dropping any connections
- Very useful in legacy/non-container scheduler worlds



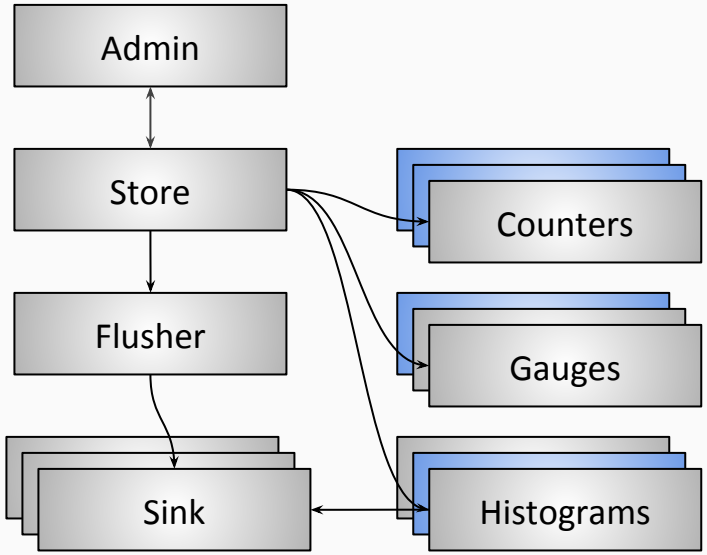
Envoy hot restart (mechanics)

- Stats/locks kept in shared memory
- Simple RPC protocol over unix domain sockets (UDS)
- Sockets, some stats, etc. passed between processes
- Built for containers



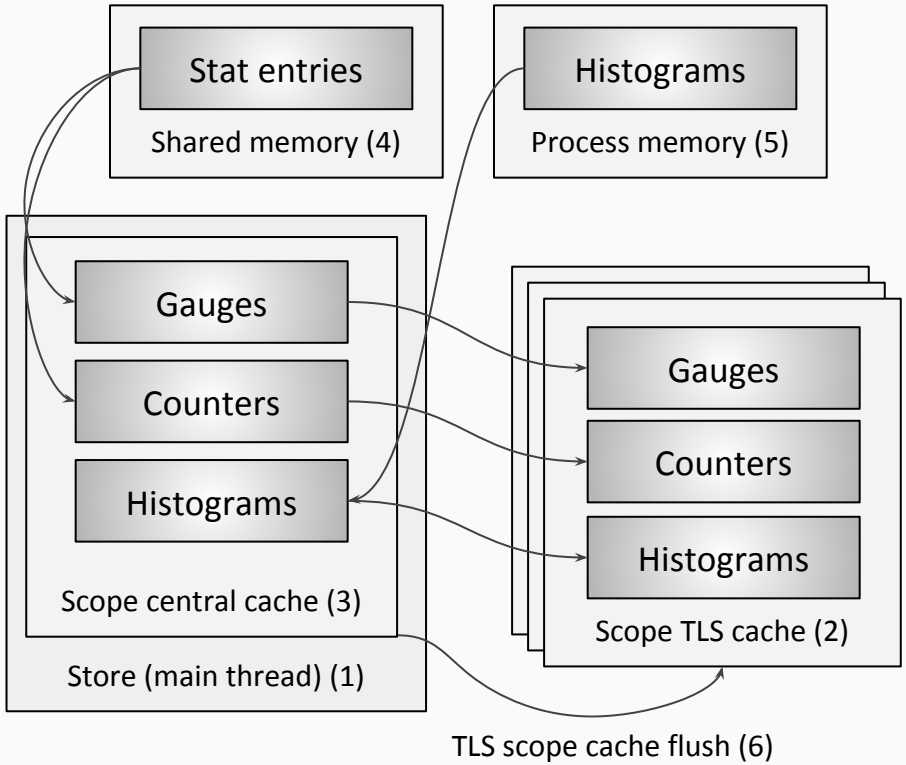
Envoy stats (overview)

- **Store**: holds stats
- **Sink**: protocol adapter (statsd, gRPC, etc.)
- **Admin**: allows pull access
- **Flusher**: allows push access
- **Scope**: discrete grouping of stats that can be deleted. Critical for dynamic xDS on top of hot restart shared memory

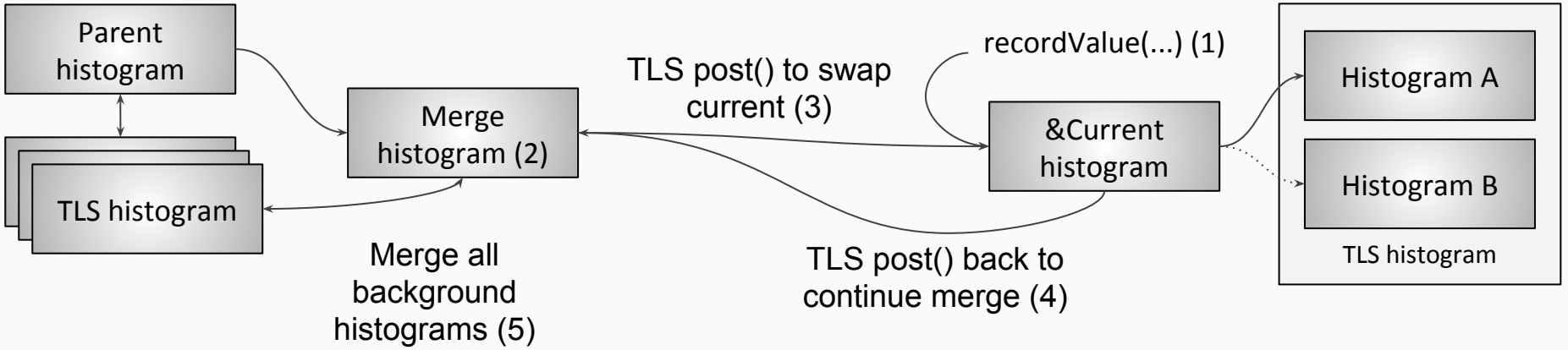


Envoy stats (TLS scopes)

1. **Store** is global
2. Stats first looked up in **TLS** cache
3. Not found, allocated in central cache, added to TLS cache
4. Counters/gauges in shared memory
5. Histograms in process memory
6. Scope deletion causes a TLS cache flush on all threads



Envoy stats (TLS histograms)



(1) TLS histogram values recorded into “current” without locks

(2) Period merge/flush

(3) Post to each worker to swap current histogram (record now happens on alternate)

(4) Post back to main thread to continue merge

(5) Merge all TLS histograms without locks

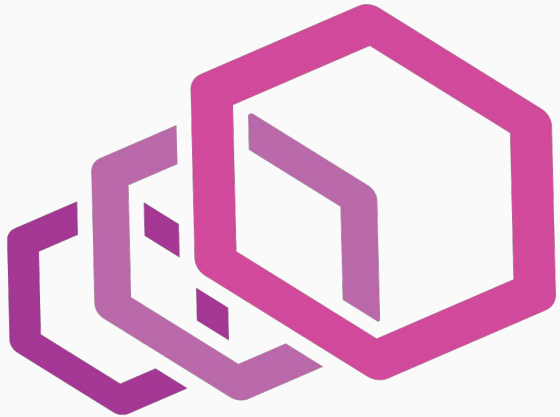
Summary

- Bias for developer productivity without sacrificing high throughput and low latency
- Architecture embarrassingly parallel and designed for mostly lock free scaling across high HW thread count
- Heavy use of RCU locking paradigm and TLS
- Design for containerized world
- Extensibility is key

Cloud native summary

Q&A

- Thanks for coming! Questions welcome on Twitter: **@mattklein123**
- We are super excited about building a **community** around Envoy. Talk to us if you need help getting started.
- https://www.envoyproxy.io/
- **Lyft is hiring!**



envoy