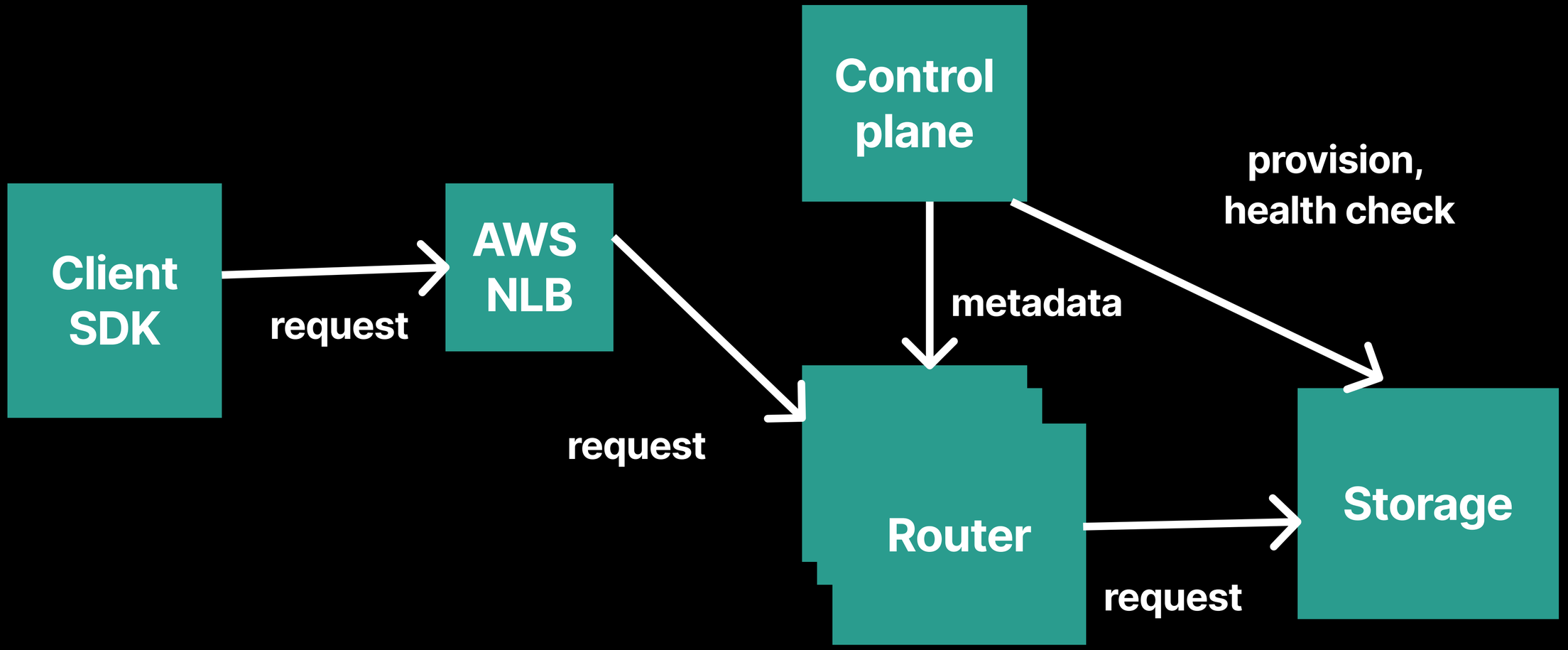
When HTTP is too heavy: Designing a minimal RPC layer with protosocket at Momento

Pratik Agarwal

Software Engineer
Figma
Previously: Momento, AWS

momento

Momento architecture

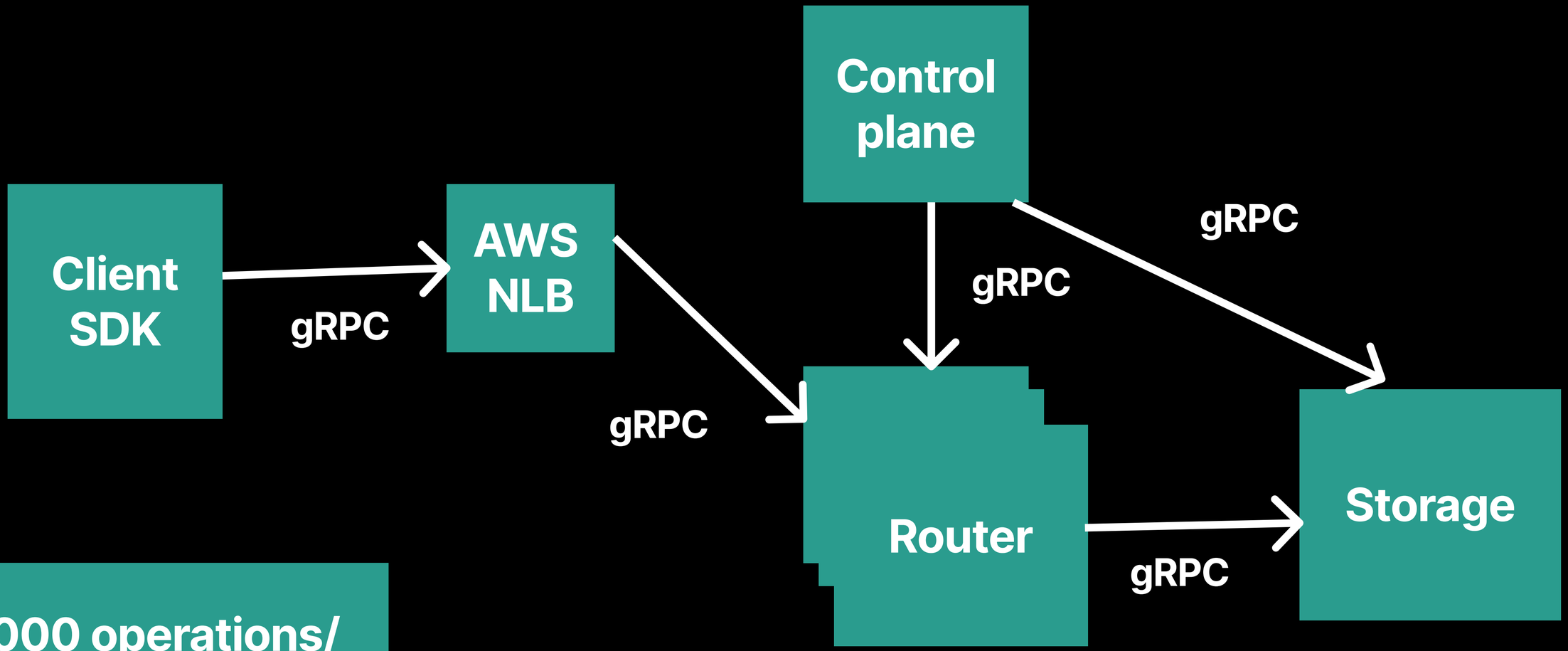


Benchmark setup

Client	Router	Storage	Item size	R/W ratio
c7g.4xlarge	c7g.2xl	r7g.xl	1KB	1:1

https://github.com/twitter/rpc-perf

Initial architecture



~20,000 operations/ second, p999 < 5ms

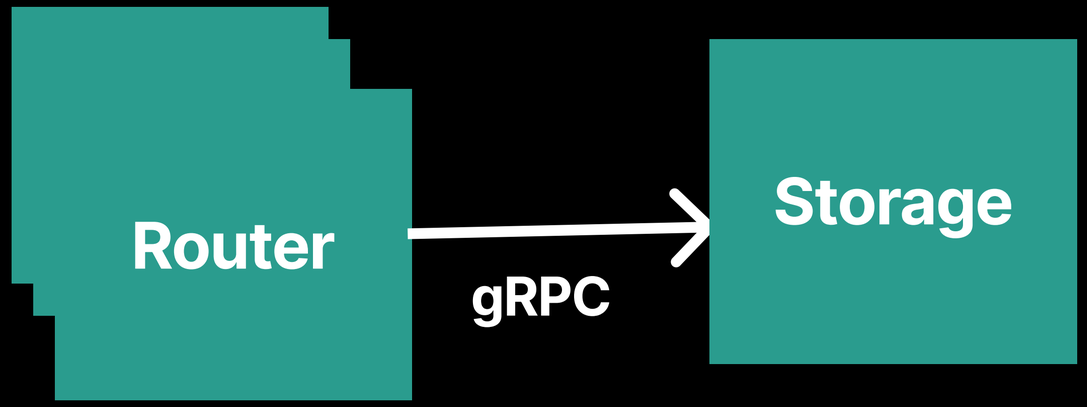
Router <> Storage communication

gRPC unary requests

- Network overhead - ~100 bytes per request
  - HTTP/2 frames + gRPC headers + protobuf envelope
- CPU usage/computation per request
  - Interceptors
  - gRPC invariants (state machines, error handling)

Results in:

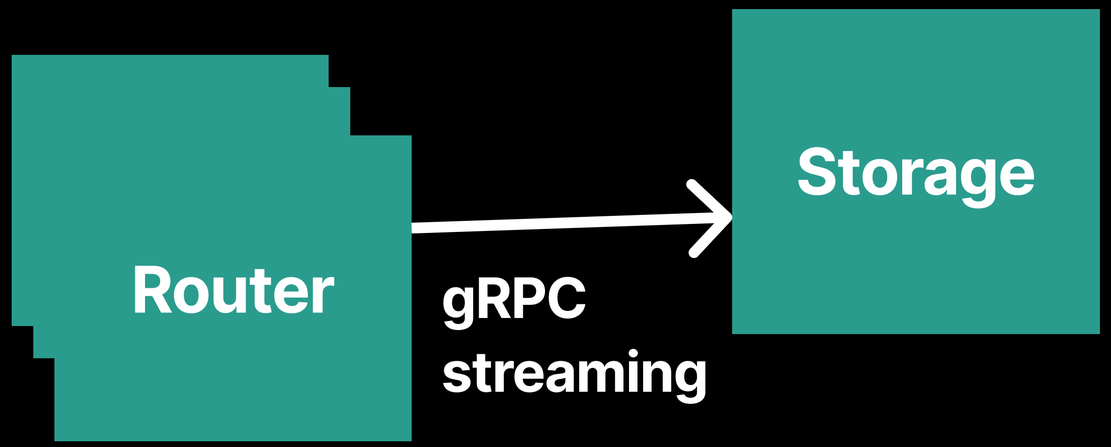
- Added latency
- Less throughput
- More ec2 cost; less margins



Router <> Storage communication

gRPC streaming

- Switched to gRPC streaming between router and storage
  - stream reuse in addition to connection reuse
  - less header/bytes overhead
  - interceptors are still per message
- Reduced network overhead
- Lower CPU usage in the interaction path
- ~20-30% more throughput
- Downside: more complex implementation



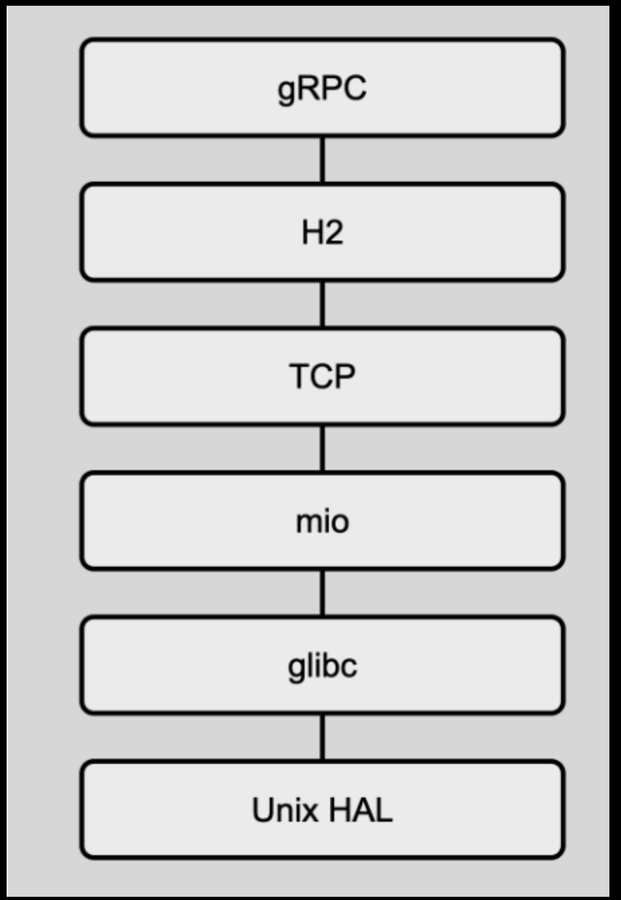
~25,000 operations/ second, p999 < 5ms

Rust’s gRPC ecosystem

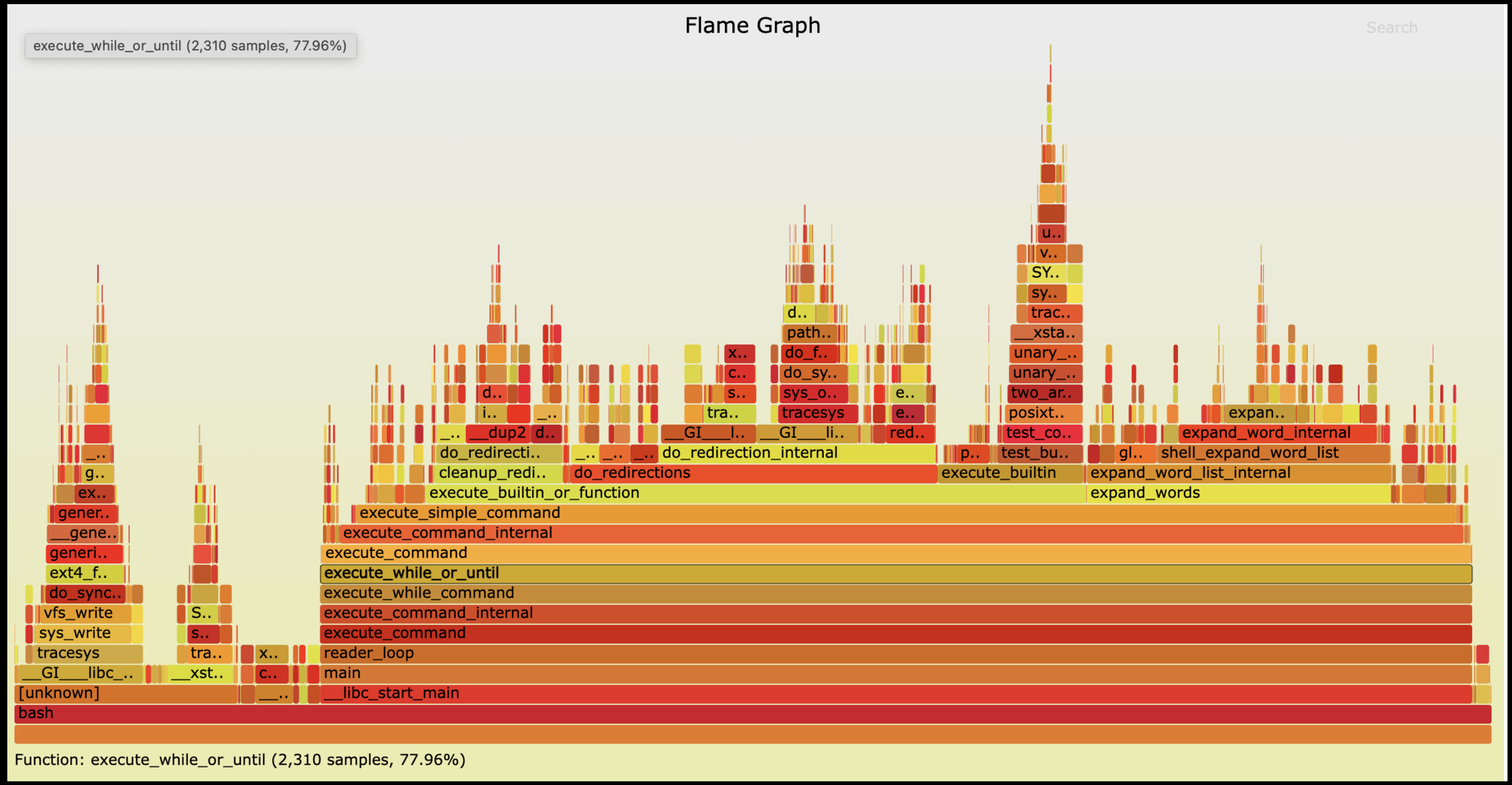
- H2 synchronization (Tokio’s Http/2 library)
- Causes task starvation and latency spikes
- https://github.com/hyperium/h2/issues/531

Library does not scale with multiple cores #531

Open bIgBV opened this issue on Apr 21, 2021 · 11 comments



Flamegraphs are awesome!



Protosocket

Message-oriented, low-abstraction tcp streams.

A protosocket is a non-blocking, bidirectional, message streaming connection. Providing a serializer and deserializer for your messages, you can stream to and from tcp servers.

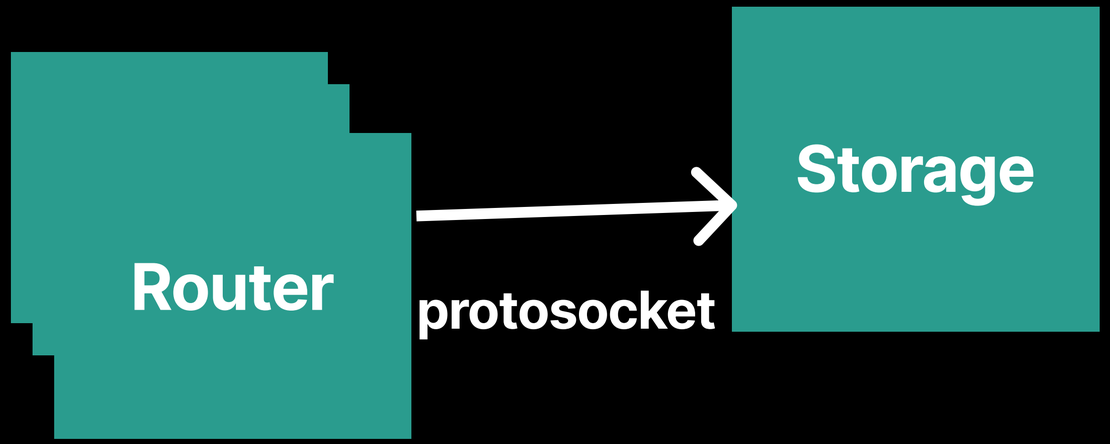
- No HTTP/2 layer - Direct TCP streams
- No shared state connection - per connection buffers and isolated state
  - Minimal synchronization points
- Multiple encoding formats supported

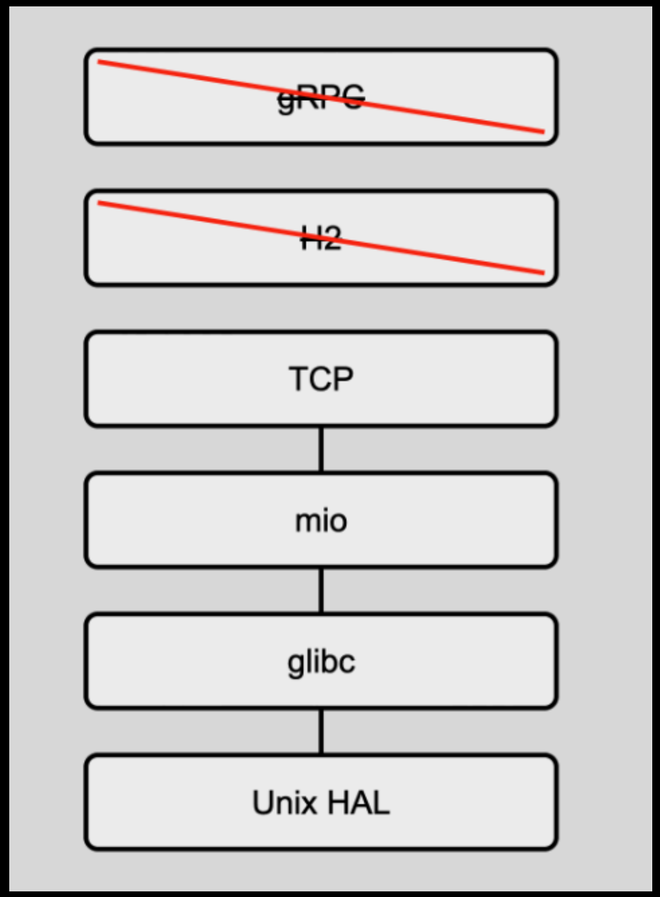
https://github.com/kvc0/protosocket

Router <> Storage communication

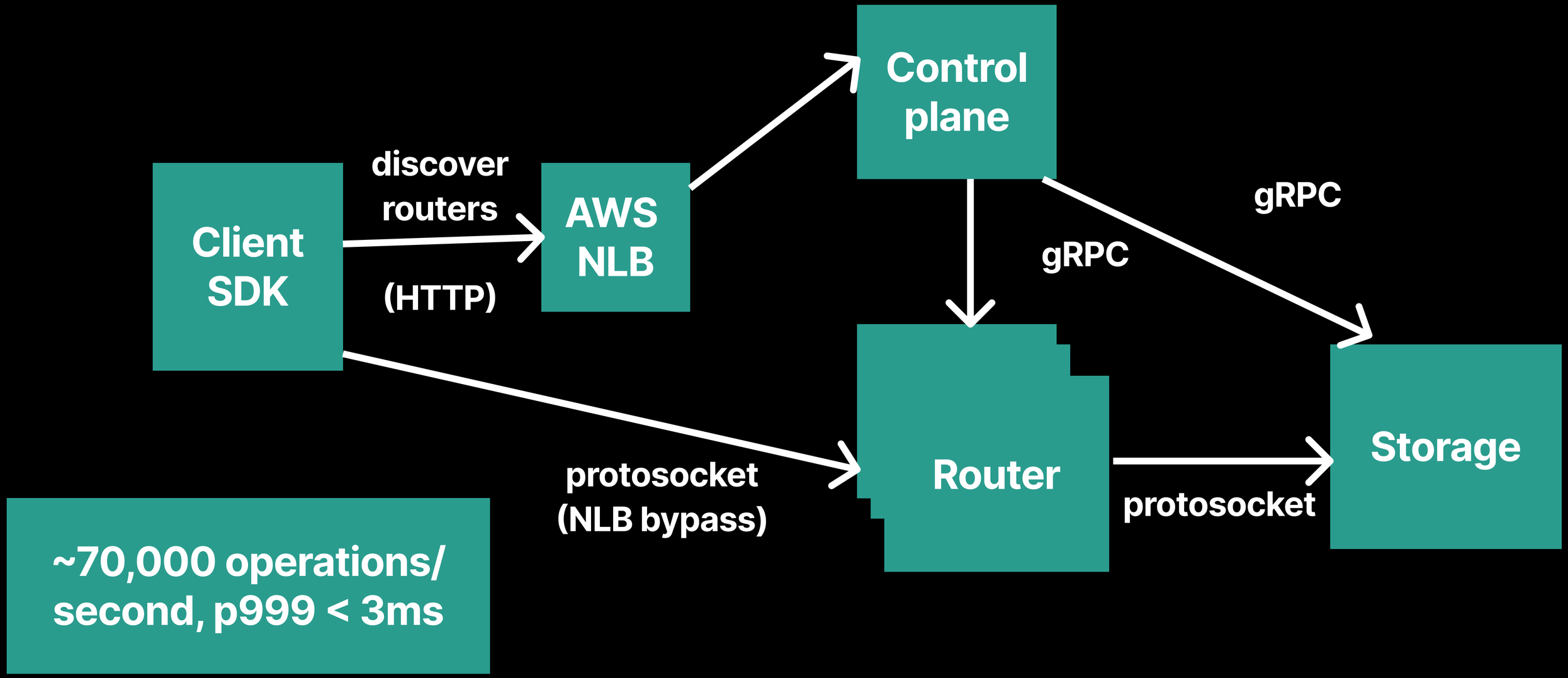
Protosocket

- Uses raw TCP to communicate b/w routers and storage servers
- prost for encoding and decoding protobuf messages
- mio provides the abstractions over low-level system calls for event driven programming





Final architecture



Numbers Recap

	gRPC Unary	gRPC Streaming	protosocket router → storage	protosocket client → router→ storage
ops/second	20,000	25,000	70,000	70,000+
p999	<5 ms	<5 ms	<5 ms	<3 ms

Tradeoffs

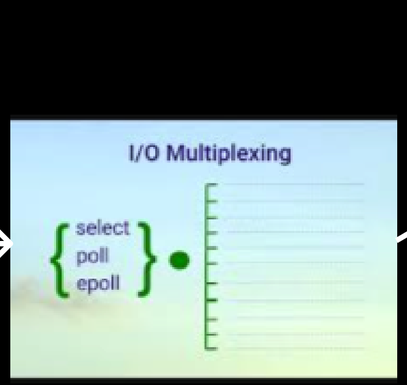
- More complex implementation
  - Although protosocket has its own rpc version
- HTTP/2 has better interoperatability
- Load balancers /proxies/firewalls are aware of HTTP!
  - Won’t be trivial to open it to customers behind corporate / custom proxies
- HTTP-aware tools (Envoy dashboards, WAF logs, distributed tracing via gRPC interceptors) don't work

Takeaways

- Measure, then optimize
  - Measure can take various forms
    - Look beyond traditional request metrics and logs
- Community maintained ecosystem might need more love
- But, std::lib is not always the best either!
  - We rewrote std::lib::mutex to get a 15% boost on the initial architecture
- Such undertakings are worth it only if your bottom line depends on it!

Async Rust

(Personal experience)



Resources

- https://pages.cs.wisc.edu/~remzi/OSTEP/
- https://mara.nl/atomics/
- https://www.brendangregg.com/linuxperf.html
- https://www.youtube.com/@6.824

Acknowledgement

Kenny Chamberlin

Khawaja Shams

Q & A