



KubeCon



CloudNativeCon

———— North America 2018 ————

gRPC Performance

Tuning Applications and Libraries

Noah Eisen (email, github, twitter handle: ncteisen)



Intended Audience

- Anyone interested in performance!

Agenda



KubeCon



CloudNativeCon

North America 2018

- gRPC Overview
- Tooling, Benchmarks, and Data
- Tuning the gRPC Library
 - Undoing Death by 1000 Paper Cuts
 - Case Study
- Breaking Down the Layers
- Tuning gRPC Applications
 - Low Hanging Fruit
 - Case Study



KubeCon




CloudNativeCon

North America 2018


gRPC Overview

gRPC Overview - History

Borg → 
Kubernetes


TensorFlow

Blaze → 
Bazel

Stubby → 

gRPC Overview - Basics

gRPC stands for **g**RPC **R**emote **P**rocedure **C**all.

★ Unstar	18,115	🔗 Fork	4,189
----------	--------	--------	-------

A **high performance**, open source, standards based, general purpose, polyglot, feature-rich RPC framework.

Actively developed and production-ready.



gRPC Overview - Generic Stack

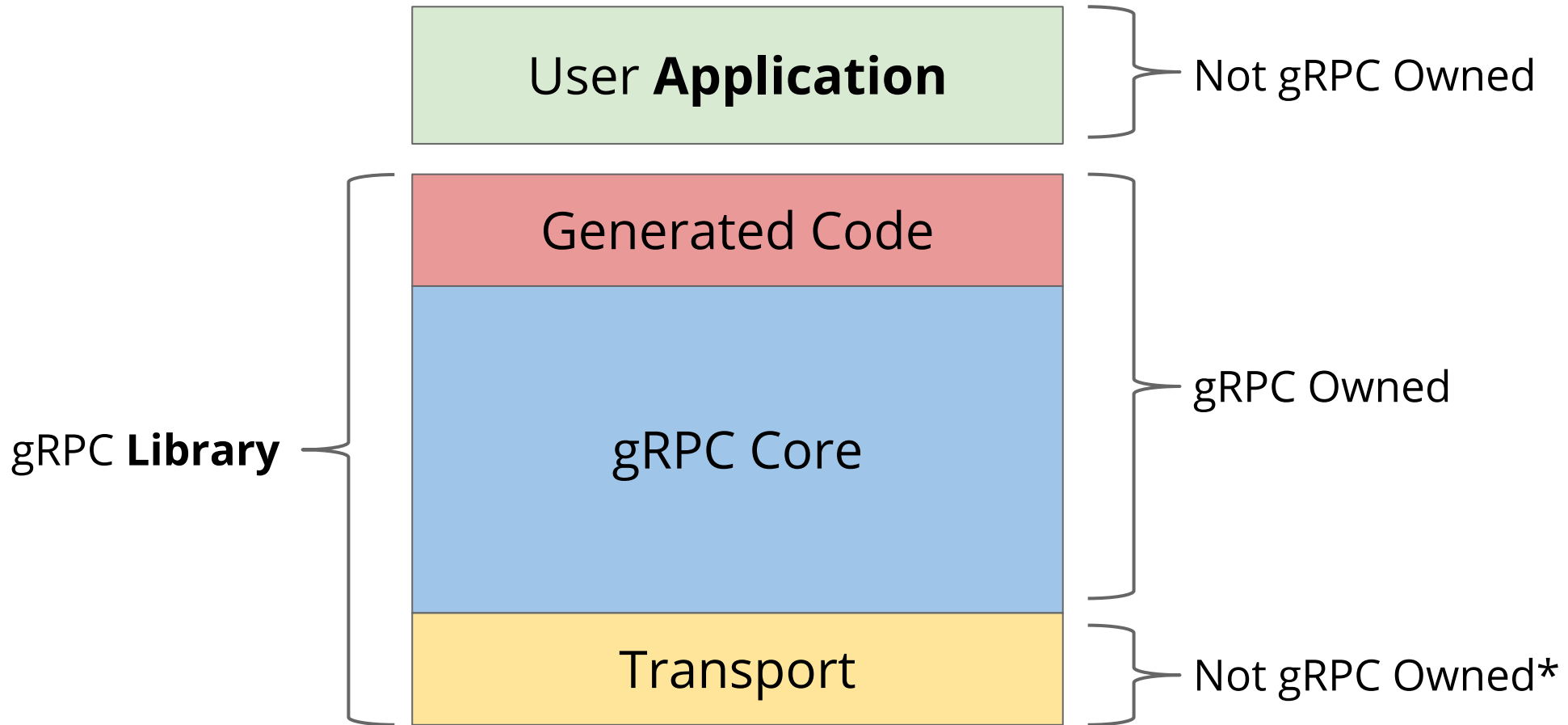


KubeCon



CloudNativeCon

North America 2018



gRPC Overview - Go Stack

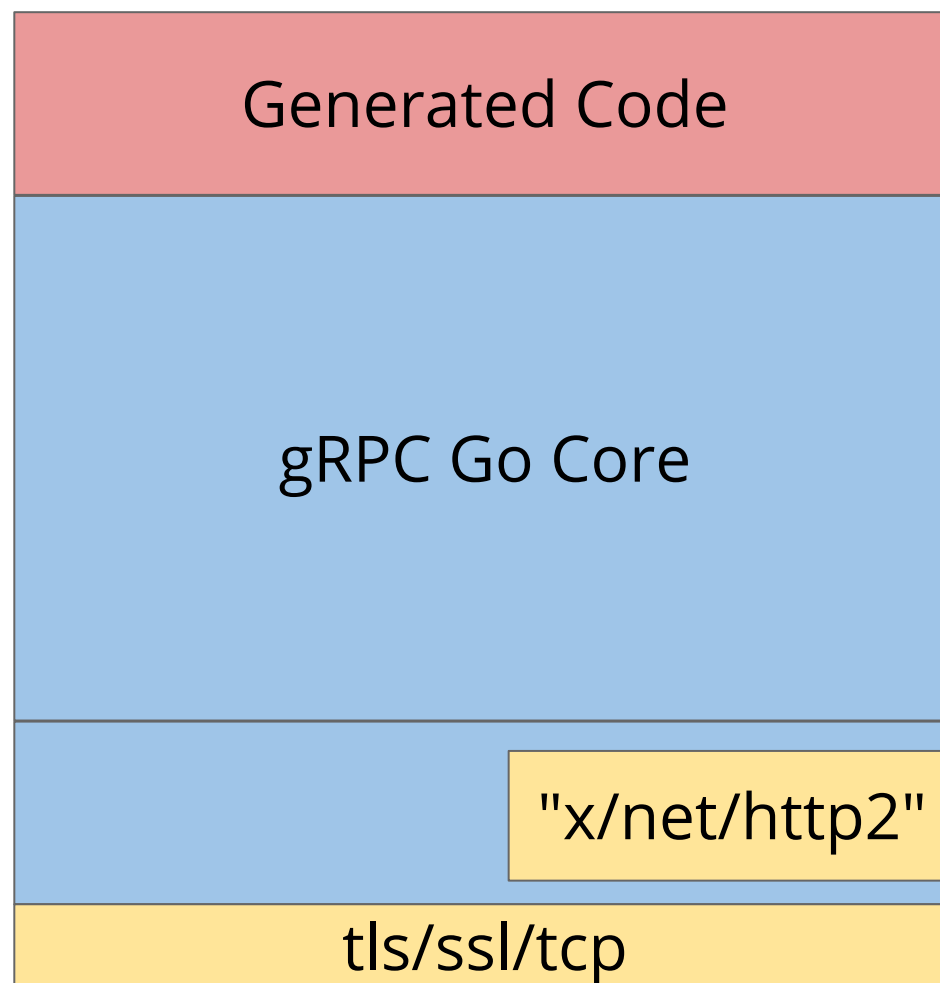


KubeCon



CloudNativeCon

North America 2018



gRPC Overview - Java Stack

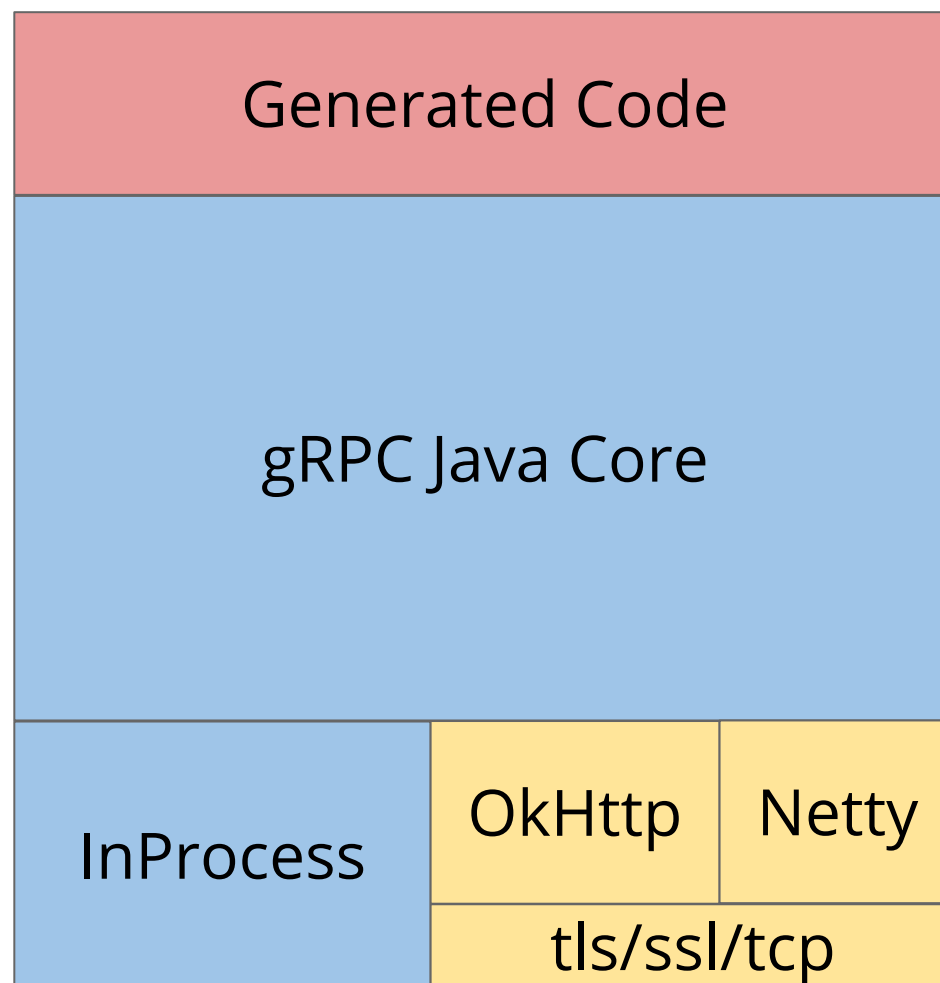


KubeCon



CloudNativeCon

North America 2018



gRPC Overview - C Stack

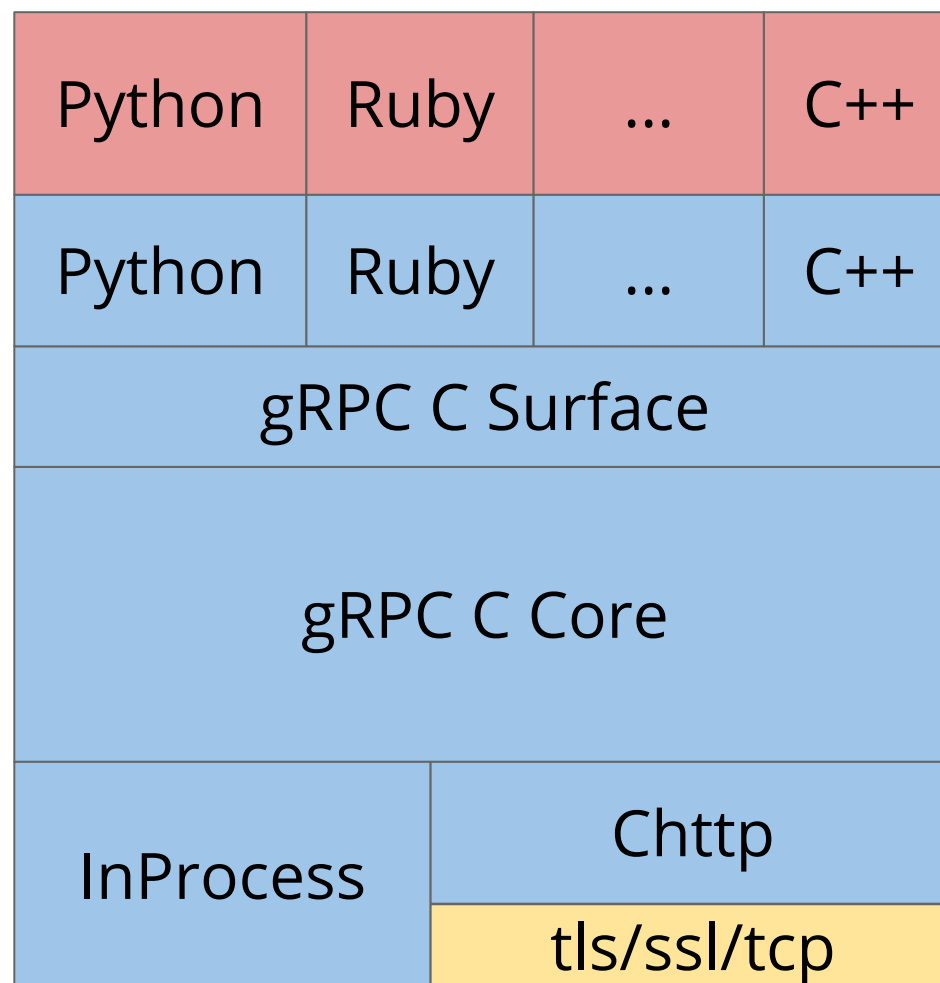


KubeCon



CloudNativeCon

North America 2018



Tuning Libraries - Key Points



KubeCon



CloudNativeCon

North America 2018

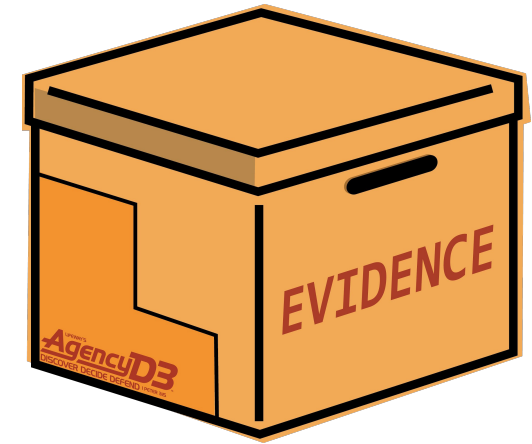
Tooling



Benchmarks



Data



Tooling



KubeCon



CloudNativeCon

North America 2018

- **In order to optimize, know where to look!**
- Tooling narrows problem scope.
- No such thing as "perfect tool".

Where are my
microseconds
going??



Tooling - Latency Traces



KubeCon

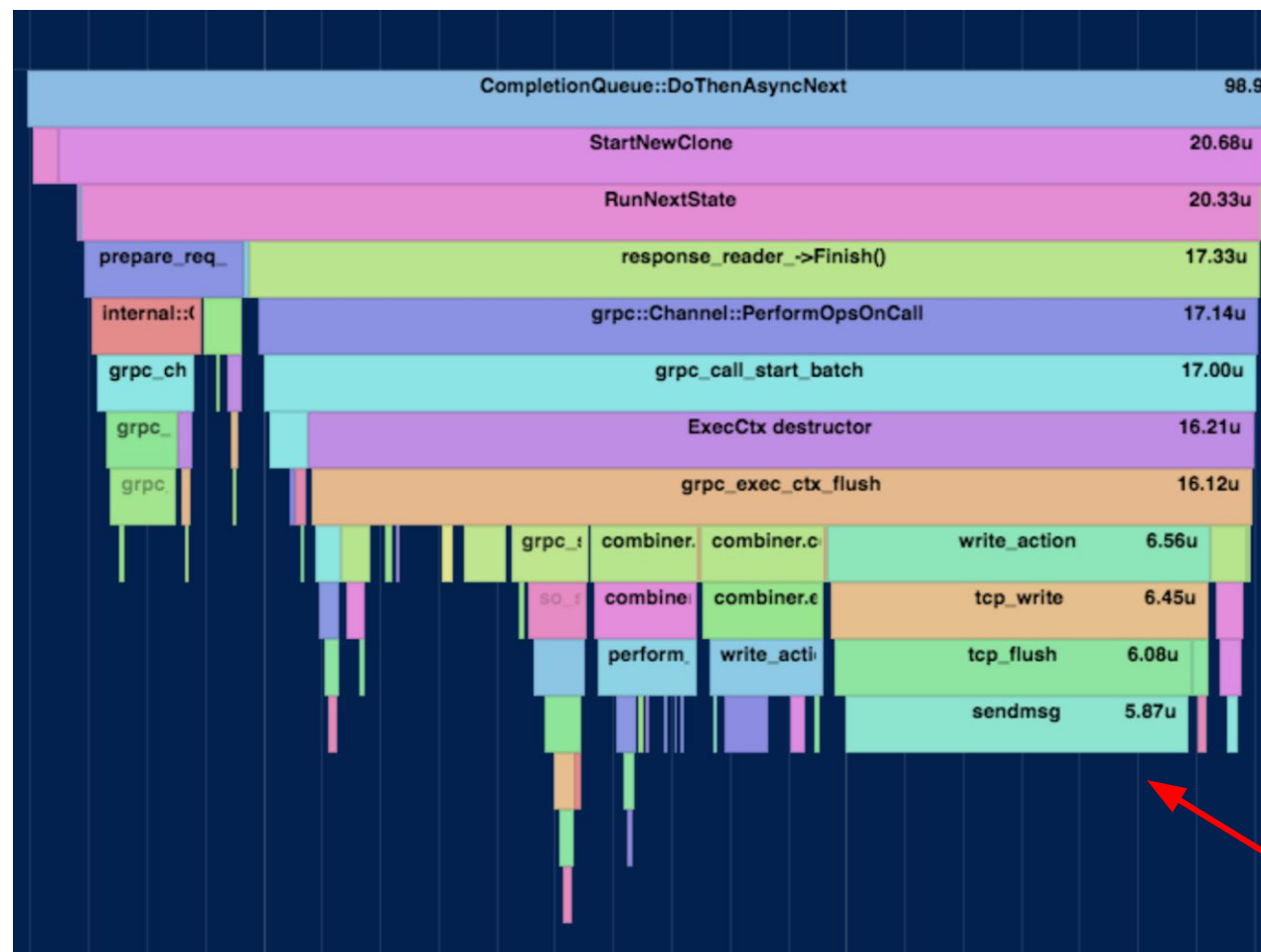


CloudNativeCon

North America 2018

```
void foo() {  
    GPR_TIMER_SCOPE("foo");  
    bar();  
}  
  
void bar() {  
    GPR_TIMER_SCOPE("bar");  
    do_more_work();  
}
```

gRPC's custom trace annotations: [here](#)



- North America 2018



Tooling - Other Tools

- Lock contention measuring tools (valgrind, mutrace)
- Customs counters for allocs, atomics.
- Kernel tools:
 - perf (general analysis)
 - strace (syscall introspection)
 - pahole (c++ struct packing)

Tooling - Bottom Line



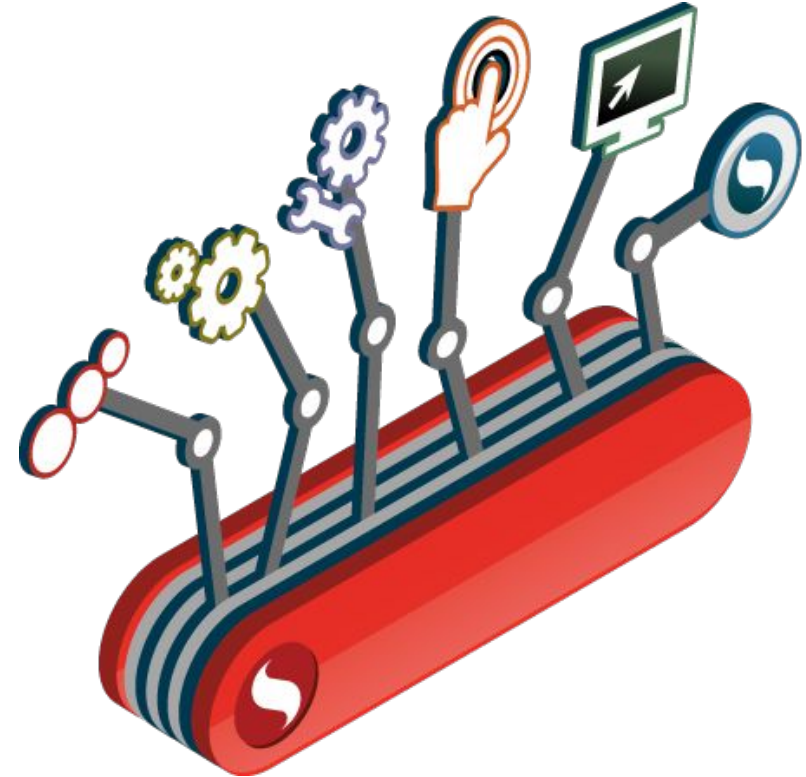
KubeCon



CloudNativeCon

North America 2018

- Obtain an **arsenal of tools**
- Grow your arsenal
- Use tools in conjugation



Benchmarks



KubeCon



CloudNativeCon

North America 2018

- **In order to optimize, know how to measure!**
- Benchmarks widen scope.



Was that really an optimization??

Benchmarks - Microbenchmarks



KubeCon



CloudNativeCon

North America 2018

```
static void BM_ErrorCreate(State& state) {  
    while (state.KeepRunning()) {  
        GRPC_ERROR_UNREF(GRPC_ERROR_CREATE("Error"));  
    }  
}  
  
BENCHMARK(BM_ErrorCreate);
```

Run on (12 X 3800 MHz CPU s)

Benchmark	Time	CPU Iterations
BM_ErrorCreate	119 ns	118 ns 5516723

Benchmarks - Synthetic



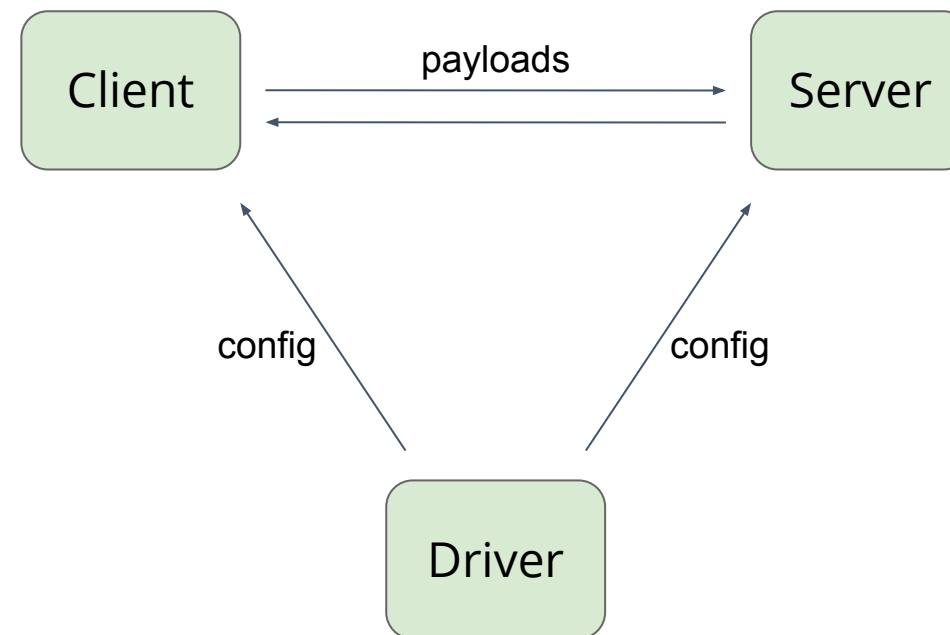
KubeCon



CloudNativeCon

North America 2018

```
{
  "name": "1-channel-1-byte",
  "warmup_seconds": 30,
  "benchmark_seconds": 120,
  "num_servers": 1,
  "server_config": {
    "async_server_threads": 1,
    "server_type": "ASYNC_SERVER"
  },
  "num_clients": 1,
  "client_config": {
    "client_type": "ASYNC_CLIENT",
    "payload_config": {
      "simple_params": {
        "resp_size": 1,
        "req_size": 1
      }
    },
    "client_channels": 1,
    "async_client_threads": 1,
    "rpc_type": "UNARY",
    "load_params": {
      "closed_loop": {}
    }
  }
}
```



Benchmarks - Application



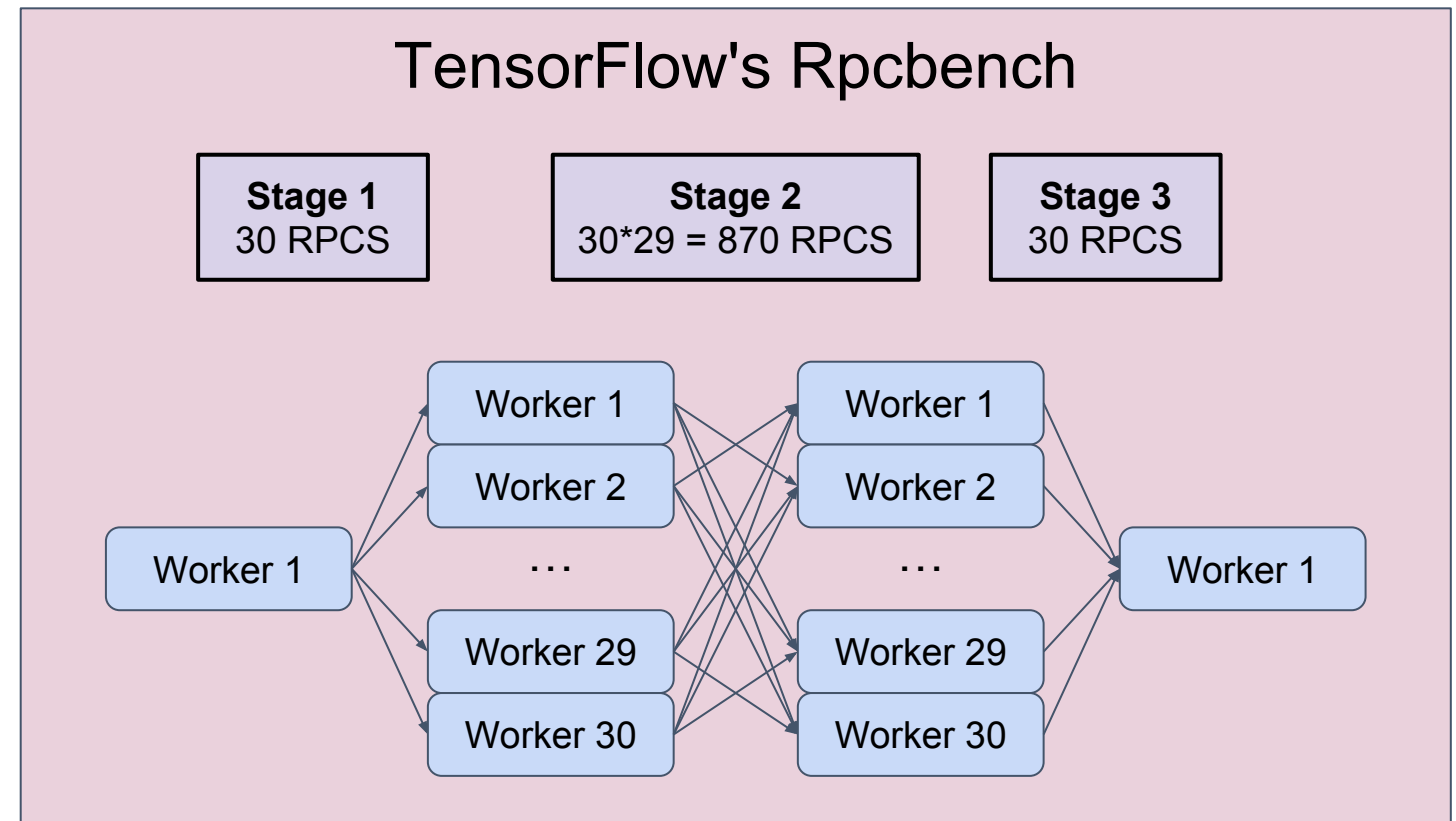
KubeCon



CloudNativeCon

North America 2018

- Written via another team's API.
- Exercises the stack in new ways.
- Only applies to libraries.



Data



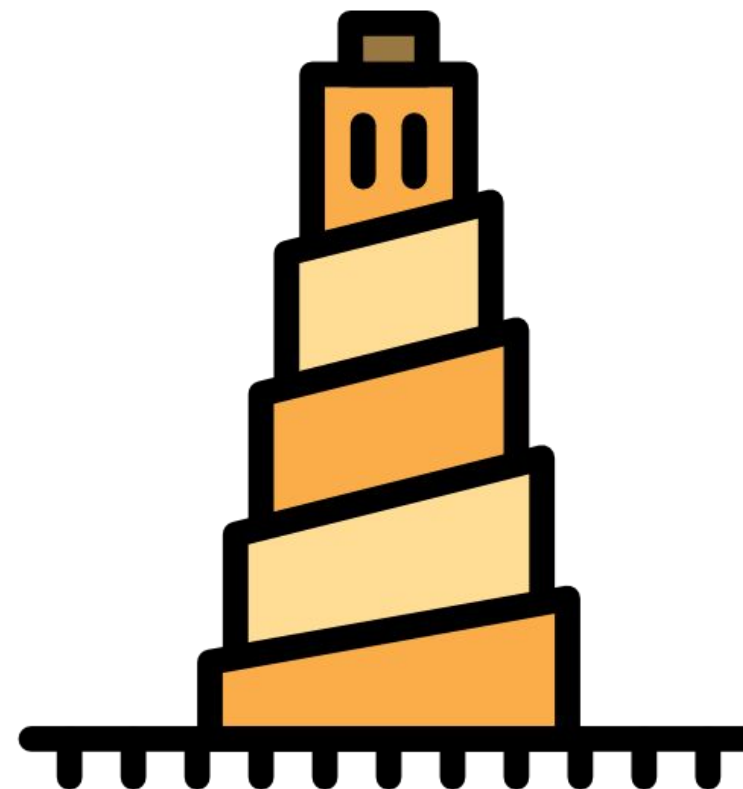
KubeCon



CloudNativeCon

North America 2018

- Team needs a *lingua franca*
- Optimizations come with:
 - data from tooling
 - data from benchmarks



Narrowing and Widening



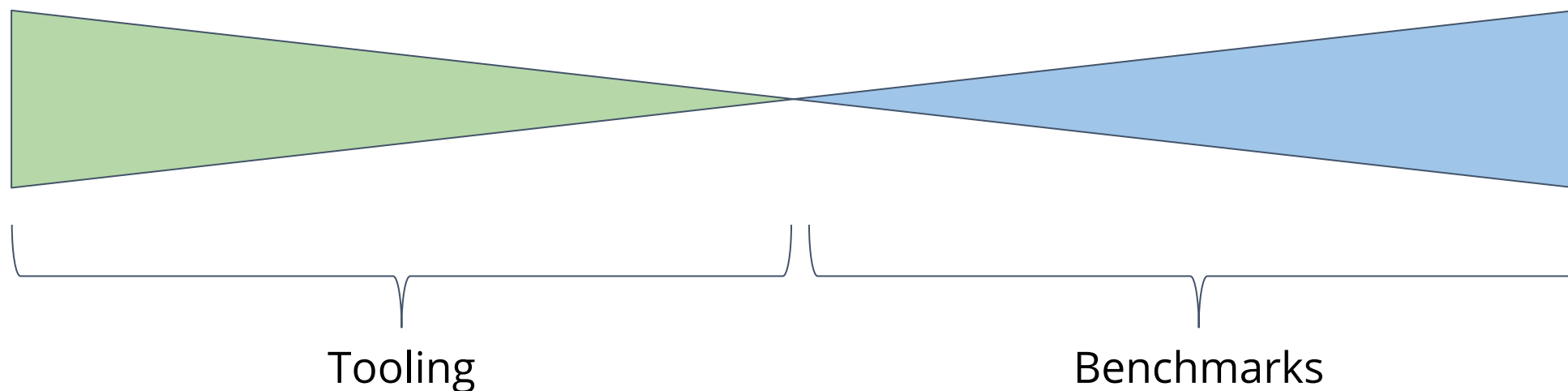
KubeCon



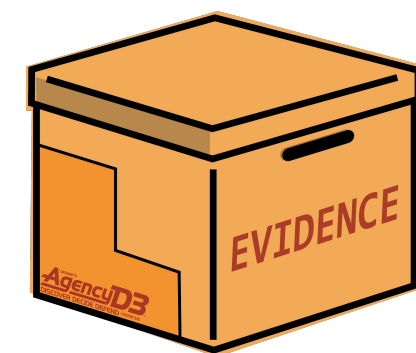
CloudNativeCon

North America 2018

- Tooling narrows scope
- Benchmarks widen scope
- End result is **data**



Data





KubeCon



CloudNativeCon

North America 2018

Tuning the gRPC Library

Undoing Death by 1000 Paper Cuts



KubeCon



CloudNativeCon

North America 2018

What to do once the "low hanging fruit" has been taken?

- Features can cause small regressions.
- Sometimes, *below margin of detection*.
- Consistent, slow, degradation of performance.

How do we reverse this process?

Undoing Death by 1000 Paper Cuts



KubeCon



CloudNativeCon

North America 2018

- New benchmark: **Minimal RPC**
 - Ping pong of 1 byte payloads
 - No security
 - No stats or tracing
 - Focused on median latency
- New tooling to use
- Noise reduction

Case Study

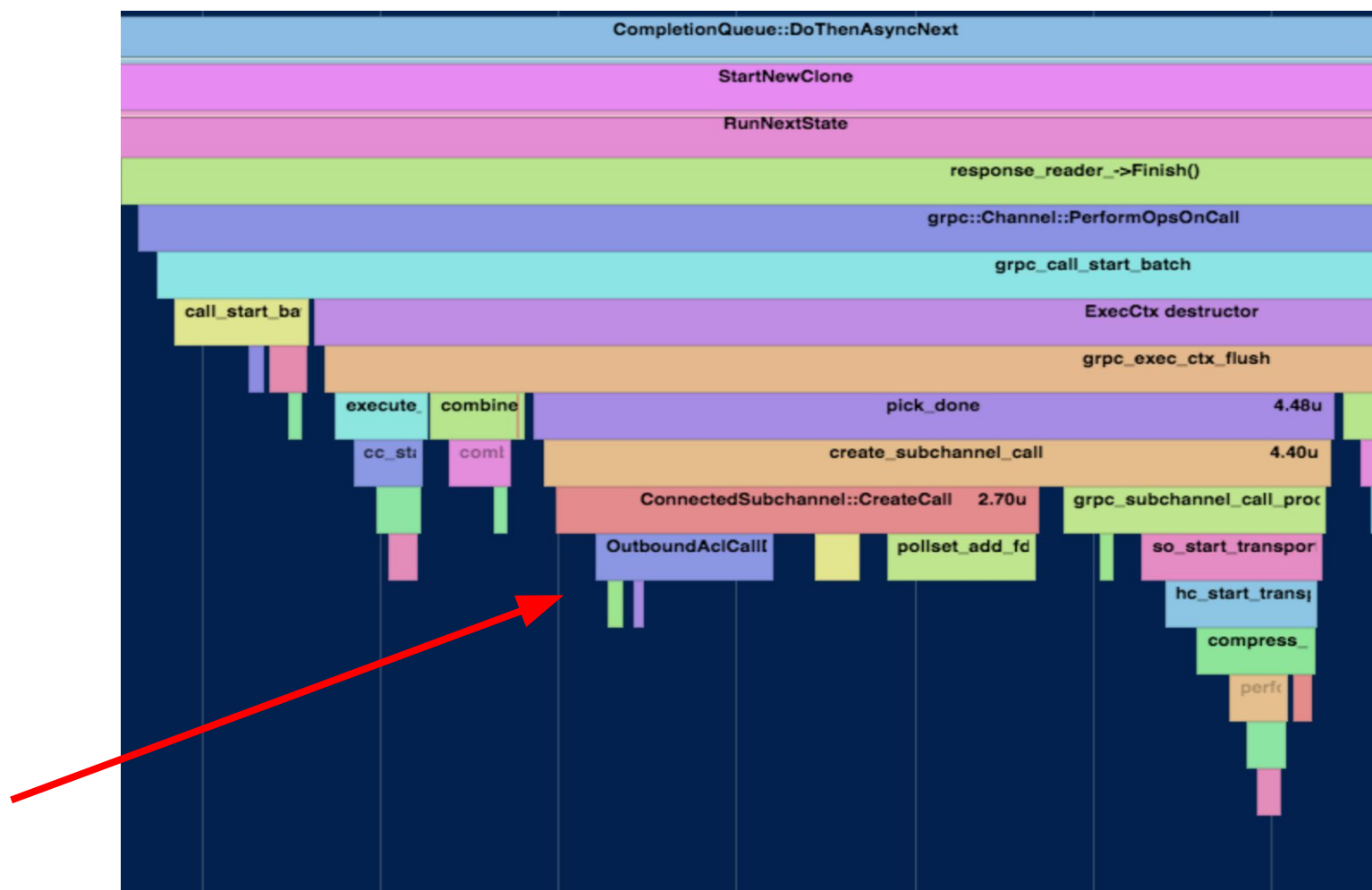


KubeCon



CloudNativeCon

North America 2018



Case Study



KubeCon



CloudNativeCon

North America 2018

```
__gnu_cxx::__s...
__gnu_cxx::__s...
__gnu_cxx::__s...
__gnu_cxx::__s...
__gnu_cxx::__v...
basic_string::...
basic_string::...
base::internal::FlagOps
Copy
InitFlagIfNece...
CommandLineFlag::Read
InitFl... Mutex:... base:... basic_...
base::GetFlag
PolicyManager::Initialize
Mutex:... Mutex:... Policy... Policy... std::_... __gnu_cxx::__s... size_c...
production_dependency_rpc::outboundacl::(anonymous namespace)::InitializePolicy
Mutex:... __gnu_cxx::__v... tc_mem...
production_dependency_rpc::outboundacl::IsAllowed
__gnu_cxx::__v... absl::string_v... __memc... grpc_s...
OutboundAclCallData::Init
grpc_call_stack_init
ConnectedSubchannel::CreateCall
create_subchannel_call
pick_done
exec_ctx_run
ExecCtx::Flush
ExecCtx::~ExecCtx
grpc_call_start_batch
Channel::PerformOpsOnCall
non-virtual thunk to grpc::Channel::PerformOpsOnCall(grpc::internal::CallOpSetInterface*, grpc::internal::Call*)
Call::PerformOps
ClientAsyncResponseReader::Finish
ClientRpcContextUnaryImpl::RunNextState
ClientRpcContextUnaryImpl::StartNewClone
grpc::testing::AsyncClient::ThreadFunc(unsigned long, grpc::testing::Client::Thread*)::(lambda)#1::operator()
CompletionQueue::DoThenAsyncNext
AsyncClient::ThreadFunc
Thread::ThreadFunc
execute_native_thread_routine
__clone
(root)
```

Case Study

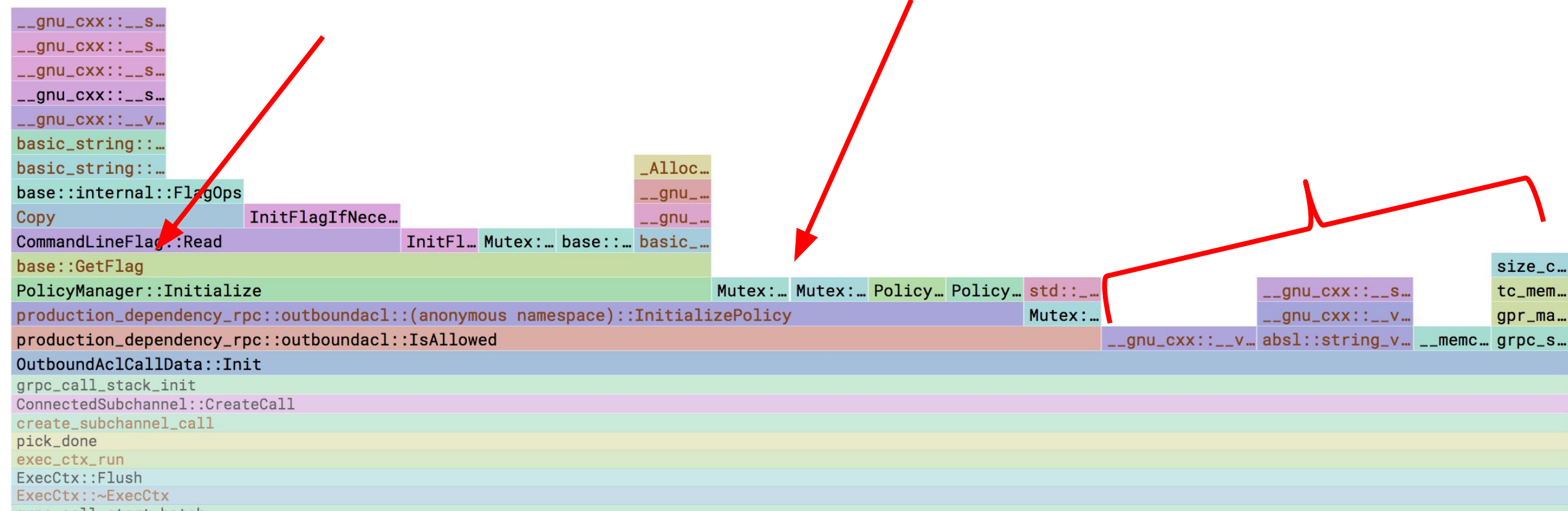


KubeCon



CloudNativeCon

North America 2018



Case Study

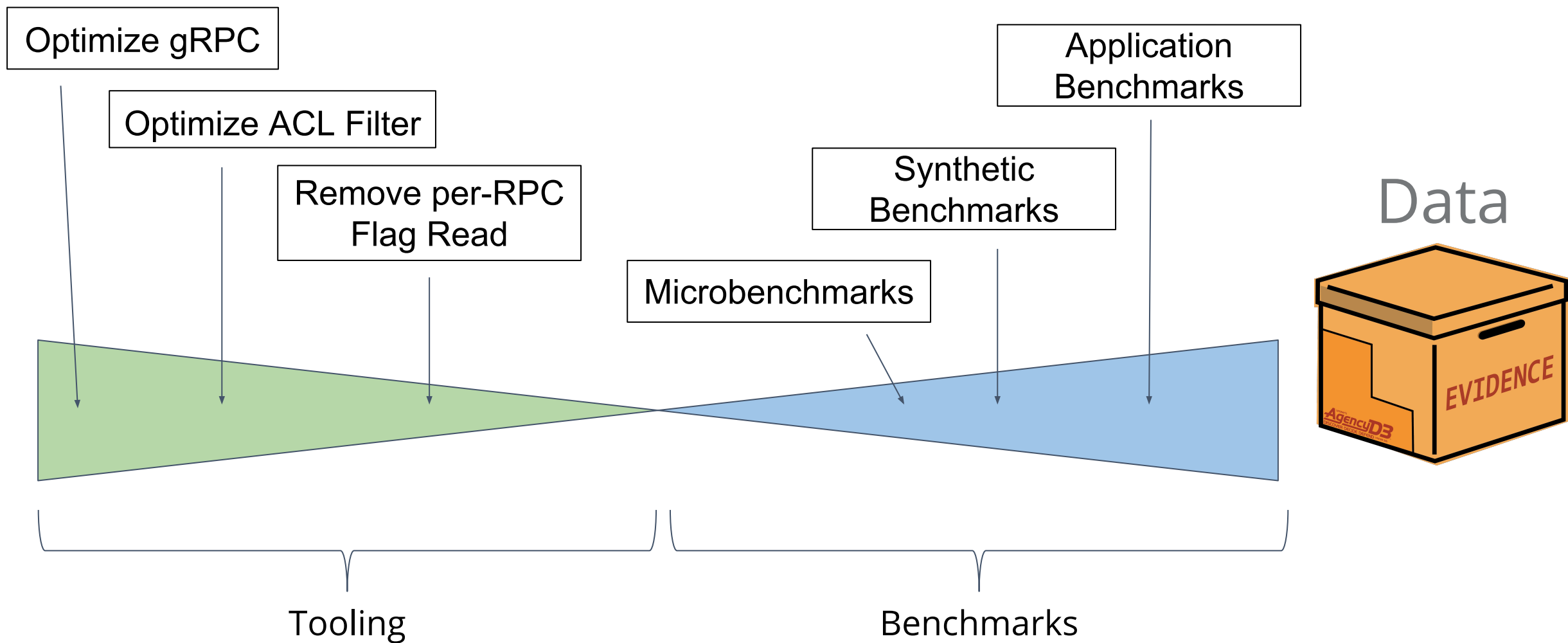


KubeCon



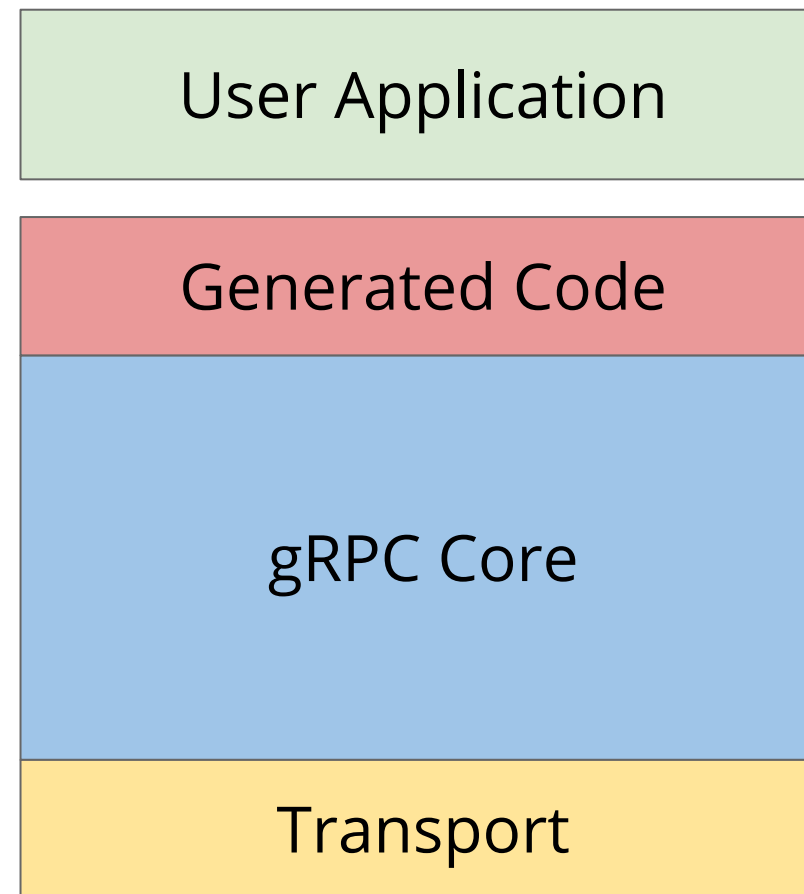
CloudNativeCon

North America 2018



Breaking Down the Layers

- Tuning below:
 - Contributions to Netty, OkHttp.
 - Contributions to x/net/http.
 - Tuning work with TCP team.
- Tuning above:
 - Next part of this talk.





KubeCon



CloudNativeCon

North America 2018

Tuning gRPC Applications

Low Hanging Fruit



KubeCon



CloudNativeCon

North America 2018

- All Language Stacks
 - Reduce allocations
 - Reduce copies
 - Reduce syscalls
 - Reduce contention
- Java Stack
 - Use async API
 - Tune thread pools
 - Tune Netty direct memory
 - Consider Netty Epoll / kqueue
- C++ Stack:
 - Use async API
 - Tune threading model
 - # of completion queues
 - # of outstanding RPCs
- Go Stack
 - Parallelize with goroutines
 - Tune Read/Write buffer size
 - # of outstanding RPCs

And of Course



KubeCon



CloudNativeCon

North America 2018

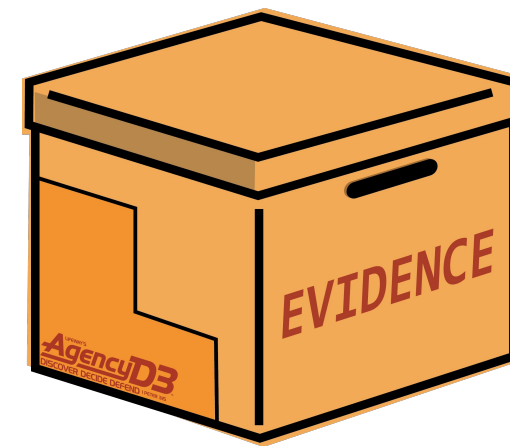
Tooling



Benchmarks



Data



Case Study



KubeCon



CloudNativeCon

North America 2018

- Distributed TensorFlow
 - Needs network layer
 - Uses gRPC
- **Goal:** improve TensorFlow-over-gRPC performance



Case Study



KubeCon



CloudNativeCon

North America 2018

- Learned TensorFlow tracing system:



Case Study



KubeCon

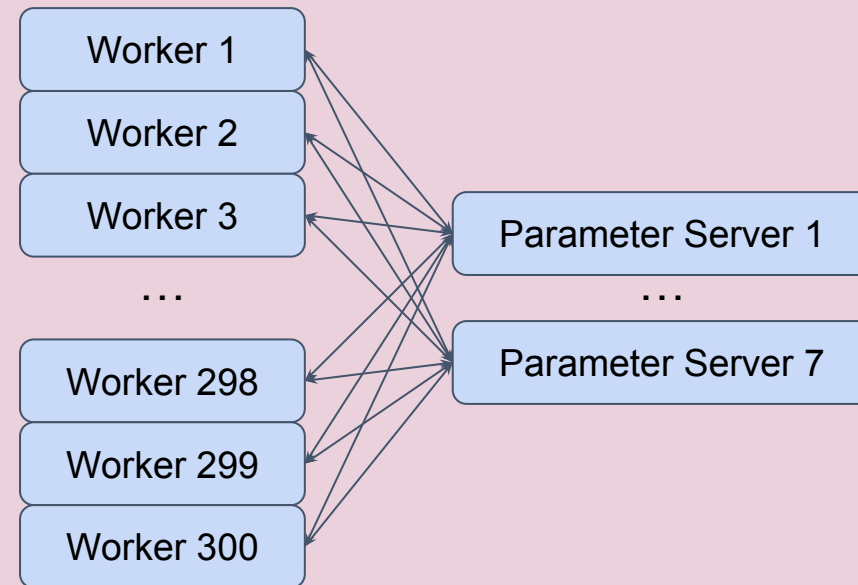


CloudNativeCon

North America 2018

- Benchmarks:
 - Rpcbench
 - Real TensorFlow training tasks
- Impactful changes:
 - better threading
 - serialization threadpool

Realistic TensorFlow RPC Topography



Breaking Down the Layers (again)



KubeCon



CloudNativeCon

North America 2018

- Tuning below:
 - Contribute to gRPC!

Your Application

gRPC Library

Thank You!



KubeCon



CloudNativeCon

North America 2018

- gRPC Resources:
 - <http://grpc.io>
 - <http://grpc.io/contribute>
 - <https://github.com/grpc>
 - <https://github.com/grpc-ecosystem>
- Personal Contact:
 - Email: ncteisen@google.com
 - GitHub: <https://github.com/ncteisen>
 - Website: <http://noaheisen.com>





KubeCon

CloudNativeCon

———— North America 2018 ————





KubeCon



CloudNativeCon

North America 2018

Appendix

Optimization: DoThenAsyncNext



KubeCon



CloudNativeCon

North America 2018

- **Author:** [kpayson64@](#)
- **Change:** [#13084](#)
- **Location:** gRPC Core.
- **Context:** gRPC has a asynchronous completion queue API. Work is placed on the queue by the application, driven by calls to AsyncNext, and then completion events are returned to application.
- **Optimization:** New API in which application can pass a lambda to be executed before AsyncNext. If this lambda triggers a completion event, it is returned by the call to AsyncNext.
- **TL;DR:** Reduced thread hops in a common case.

Optimization: TF Threading

- **Author:** ncteisen@
- **Change:** 0d5fb10
- **Location:** TensorFlow application layer.
- **Context:** TensorFlow has GrpcWorker class, which is responsible for encapsulating the gRPC network layer from the TensorFlow application.
- **Optimization:** Allow multiple threads to service the GrpcWorker's completion queues.
- **TL;DR:** More parallelism.
-

Optimization: Epoll Exclusive

- **Author:** ctiller@
- **Change:** #12789
- **Location:** gRPC Core.
- **Context:** gRPC has an internal polling system to efficiently interact with network I/O. It has gone through several iterations and optimizations.
- **Optimization:** New polling system, epollx, that relies on the EPOLLEXCLUSIVE flag for epoll_ctl.
- **TL;DR:** Thread are woken up more efficiently.

Optimizations to the Minimal RPC



KubeCon



CloudNativeCon

North America 2018

- hcaseyal@
 - #15839, #15879, #15883 (moving allocations to call arena)
- kpayson64@
 - #13947 (adds fd cache to avoid epoll_ctl)
- ncteisen@
 - #15578 (compile out spammy tracer)
- yashykt@
 - #15280 (compile out stats machinery in opt builds)
 - #15200 (adds new closure scheduling mechanism)
 - #15044 (adds compiler hints)