Using the Thrift RPC Framework in Zephyr

2023-06-28: Embedded Open Source Summit

Chris Friedt
Embedded SWE, Meta
Thrift Module Maintainer


gentoo linux™
Zephyr®
sched
Kconfig
timers
cc26xx
cc13xx
posix
spi
atomics
riscv
ieee802154
net
dns_sd
pthread
tls
libc
thread
sensor
clock
soc
pinctrl
emul
mdns
release
cc1352r1_launchxl
boards
cc1352r_sensortag
greybus
Devicetree
gpio
logging
ring_buffer
perf
bluetooth
LTS
kernel
arm
i2c
C++
yocto
PROJECT

https://bit.ly/442kJrJ

Agenda

01 What is Thrift?

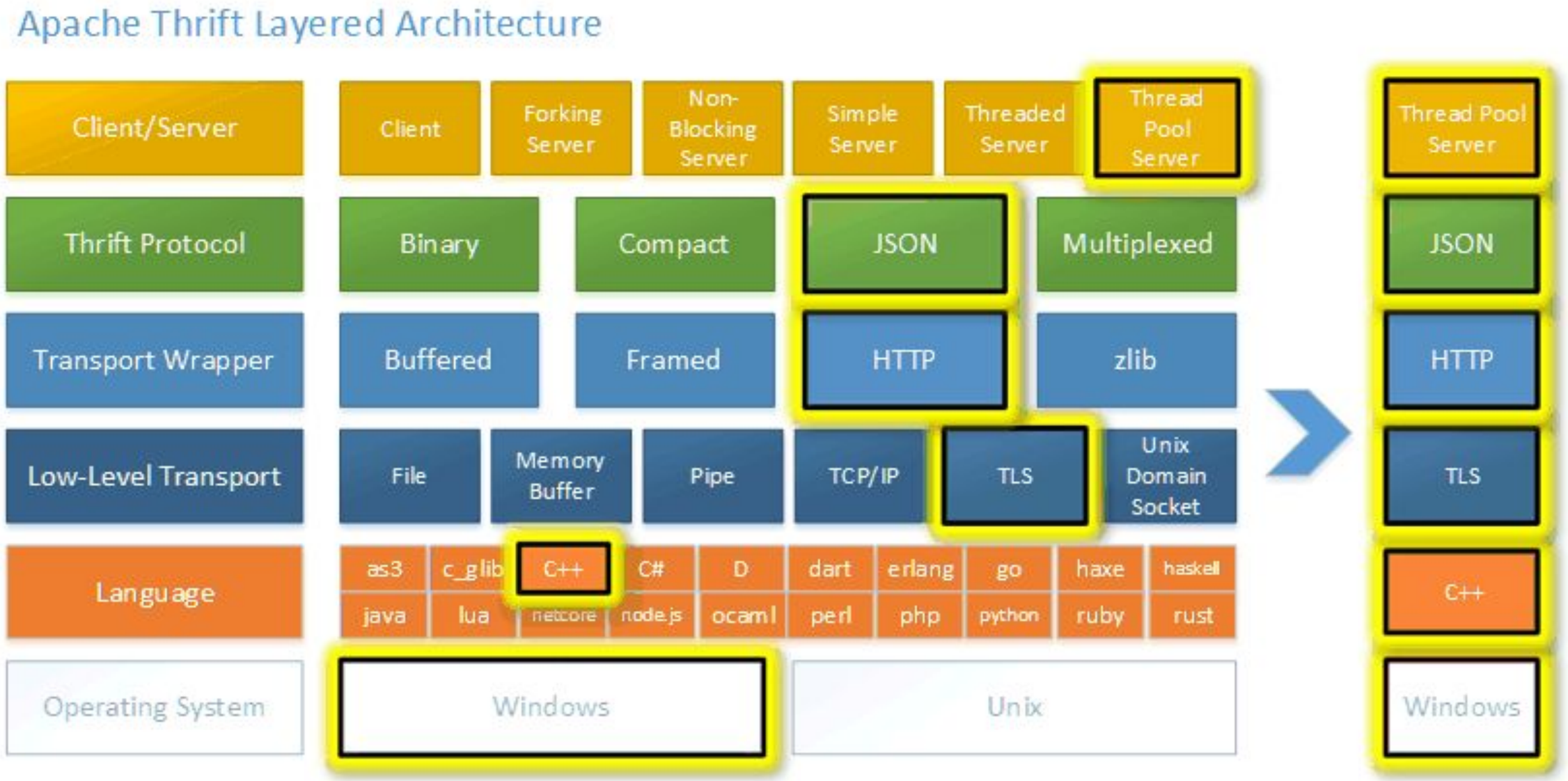
02 Thrift in Zephyr

03 Status Update & Additional Work

01 What is Thrift?

https://bit.ly/44nXnwq

https://bit.ly/433nA2p


Apache Thrift Layered Architecture
Client/Server
Client
Forking Server
Non-Blocking Server
Simple Server
Threaded Server
Thread Pool Server
Thrift Protocol
Binary
Compact
JSON
Multiplexed
Transport Wrapper
Buffered
Framed
HTTP
zlib
Low-Level Transport
File
Memory Buffer
Pipe
TCP/IP
TLS
Unix Domain Socket
Language
as3
c_glib
C++
C#
D
dart
erlang
go
haxe
haskell
java
lua
netcore
node.js
ocaml
perl
php
python
ruby
rust
Operating System
Windows
Unix
Thread Pool Server
JSON
HTTP
TLS
C++
Windows

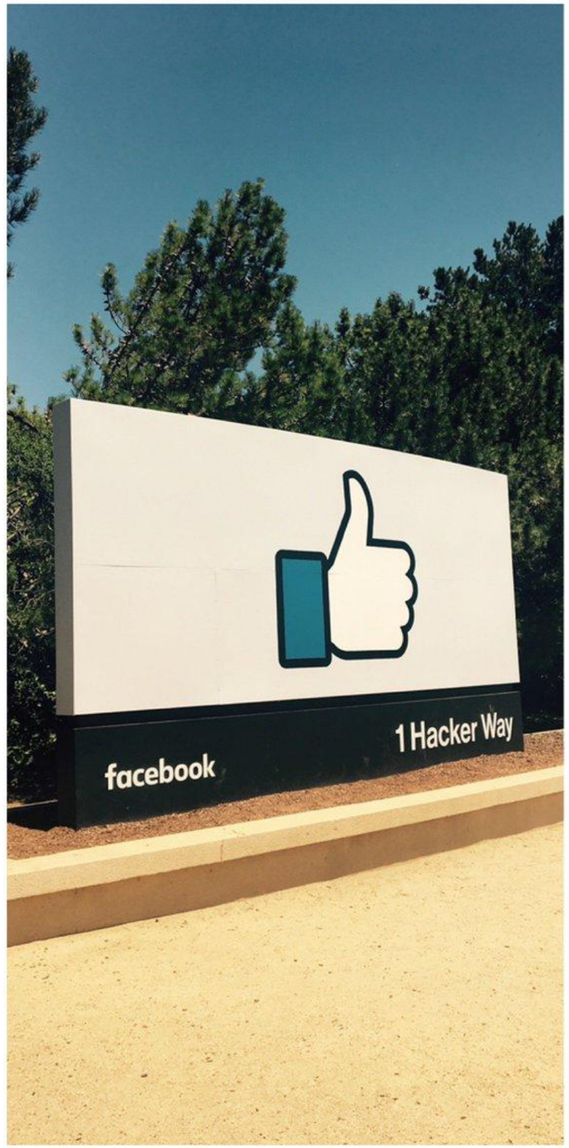
Remote Procedure Call Frameworks

- RPC is nothing new
  - Solution for ad-hoc, hand-rolled, bit-stuffed “protocols”
  - Solution for re-inventing TCP 1000x times
  - Reliable asynchronous communication is hard
  - ASN.1 (1984, ITU-T) now ISO/IEC 8825-1:2021
  - Sun RPC (1984-1986, part of NFS)
  - Part of glibc (and others) to this day (rpcgen)
  - TLV (Tag-Length-Value) representation
  - E.g. Netlink, MISB (oddly I did MISB in Thrift!)
- Some issues with SunRPC
  - every message (procedure) required a version
  - absolute nightmare for inter-version compatibility
- Protobufs, Thrift, and gRPC started around the “downfall of Sun”
  - Drop message versions (just use new Tag)
  - Do not reuse Tags
  - Guaranteed consistence +/- one rev (i.e. commit)

```
/*
 * The directory program definition
 */
program DIRPROG {
        version DIRVERS {
                readdir_res
                READDIR(nametype) = 1;
        } = 1;
} = 0x20000076;
```

https://bit.ly/4411BdP

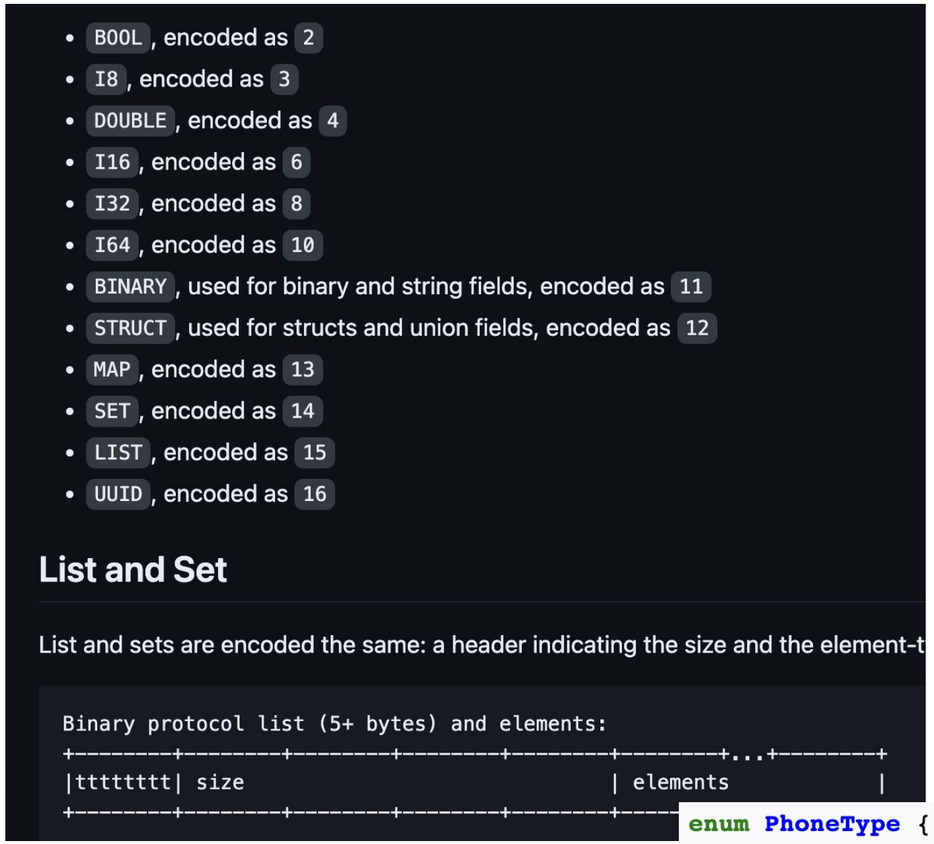
01 WHAT IS THRIFT?



[@codePrincess]

Thrift History / Internals

- Created at Facebook (now Meta) 2006-2007
- Inspired by Protocol Buffers
- Used extensively throughout Meta infra
- Released to the ASF in 2007, original whitepaper
- Adopted by several cloud companies, some OSS projects
- Not only primitive types such as *bool*, *i32*, and *double*, but also rich types such as *enum*, *list*, *map*, and *struct*
- Not only Binary, Compact Binary, but also e.g. JSON
- Procedures (methods) can *throw* any kind of *struct* and take any number of arguments
- All OS's, 27 Languages,
- Facebook forked Thrift in 2014 😅 to create fbthrift



https://bit.ly/3CS4PV8

https://github.com/facebook/fbthrift

```
enum PhoneType {
  HOME,
  WORK,
  MOBILE,
  OTHER
}

struct Phone {
  1: i32 id,
  2: string number,
  3: PhoneType type
}

service PhoneService {
  Phone findById(1: i32 id),
  list<Phone> findAll()
}
```

https://bit.ly/443gpse

02 Thrift in Zephyr

https://bit.ly/3JxrVUs

https://bit.ly/3JvDyeP

Zephyr® Project

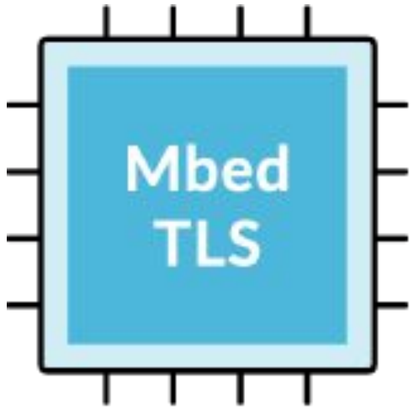
https://zephyrproject.org/store/

https://bit.ly/3Xv60D9

Google Summer of Code, 2022

- Thrift ported to Zephyr by Young Mei (StdElectronics)
  - Zephyr Blog Post
  - C++ Code Generation
- Supported Features
  - Binary Protocol
  - Compressed Binary Protocol
  - Zlib Transport (via uzlib, muzic)
  - TLS sockets via MbedTLS
- All Zephyr Architectures Supported
- All Thrift features (ThriftTest.thrift, Zephyr testsuite)
- Multi-OS, Multi-language samples
- Released in Zephyr v3.3.0 (-zlib) `[EXPERIMENTAL]`

https://bit.ly/43aC5le



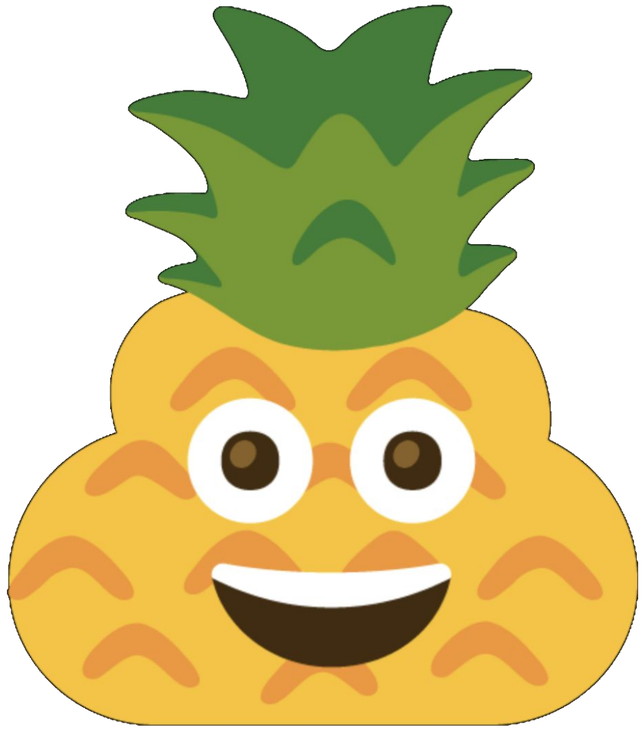
How Can I use Thrift in Zephyr?

Nice features about Thrift is it is Transport Agnostic, but also ways to stitch things together

- TFDTransport
  - anything that can be represented as a file descriptor
  - socket (TCP, UDP, socketpair), text file,
  - In Linux, “everything is a file”
  - /dev/mem, /dev/gpu0, /dev/accel0, …
- TMemoryBuffer
  - e.g. dual-port SRAM, DMA buffer…
  - In Zephyr, I wonder if it’s possible to wrap an arbitrary *const struct device** with an integer file descriptor… 🤔
  - fdtable
  - What if I wanted to use Thrift over a UART? 🤔

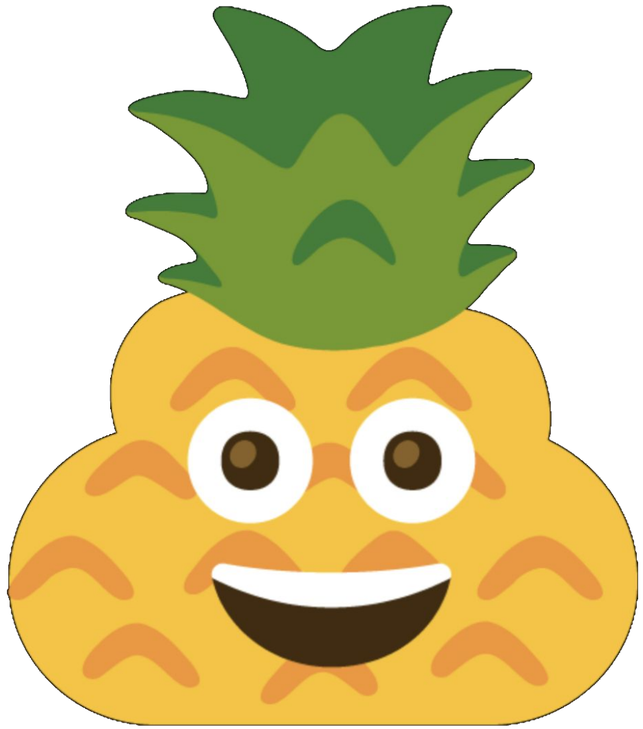
Integration Challenges

- Partial C++ Support in Zephyr
  - See C++ Roadmap (#45785)
  - Support for std::thread and std::this_thread (#25569)
  - Get std::mutex, std::condition_variable, … free
  - No async (due to missing *std::thread* support)
- C++ Requirements
  - Zephyr SDK needs gthr-posix.h (#43729)
  - Support pthread_create() dynamic stacks (#25973)
  - socketpair() (testing) needs subsys/net (#51211)
- POSIX requirements
  - Well, dynamic thread stacks benefit all languages
  - Fix at kernel level
- Several workarounds (no Mutex synchronization)



Integration Challenges

- We used eventfd() for notification in TFDServer
  - eventfd_read() / eventfd_write() deadlock (#58790)
  - Fixed (with 10x performance improvement) 👍
- No Compression / Decompression subsys yet 🤷‍♂️
- Code size is huge - can run / test in Qemu but very difficult to fit onto practically size
- SRAM requirements were quite large (for zlib) but reduced dramatically by Young during GSoC 2022



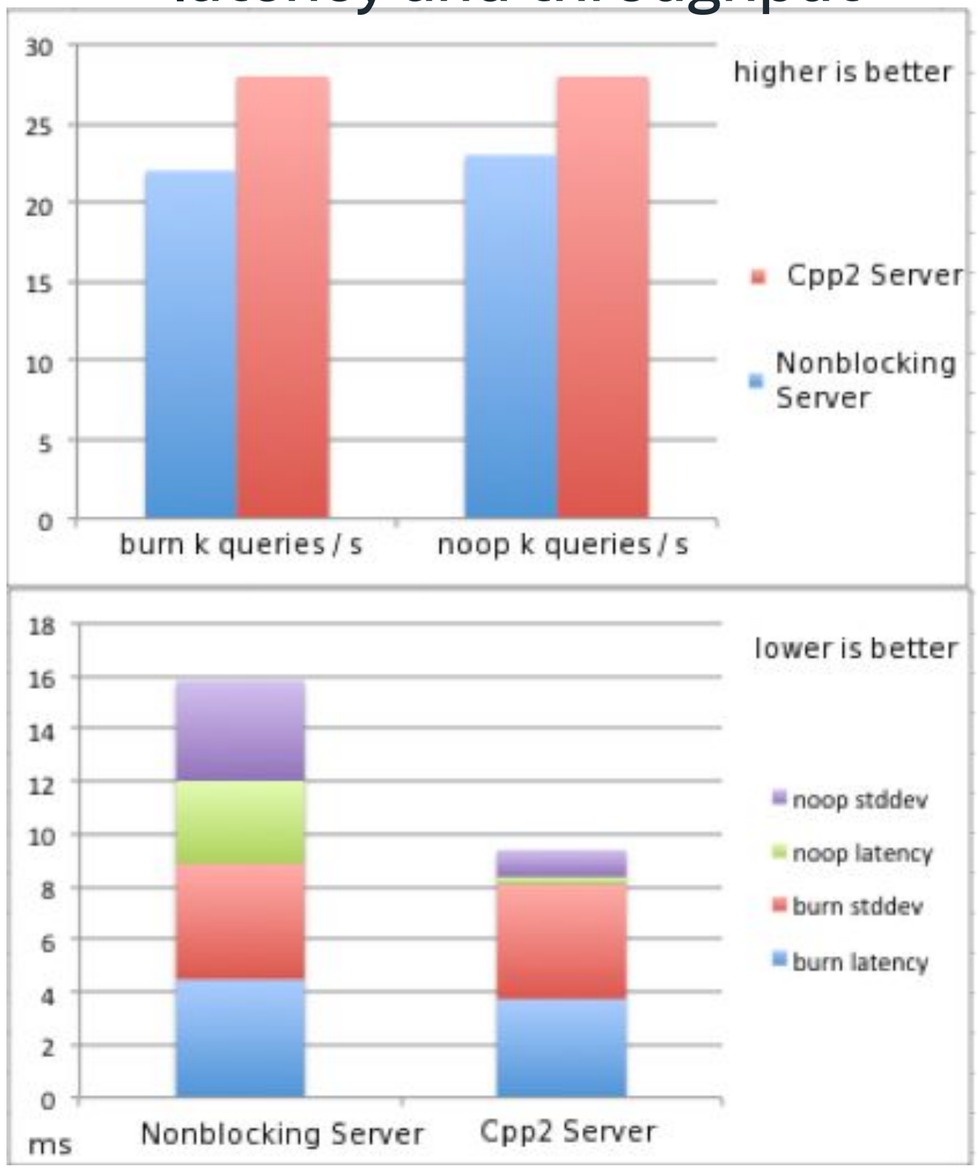
03 Status Update & Additional Work

fbthrift

- https://github.com/facebook/fbthrift
- Major C++ enhancements over Apache Thrift
- Fully asynchronous
- Server performance gains of 5-10k queries / s
- Switch to Google Test Framework
- Now uses std::coroutine (C++20)
- Dramatically lower latency across the board
- Code size reduction
- Use std::string_view (C++17) where possible
- Zero-copy buffers,

😁 👍

latency and throughput

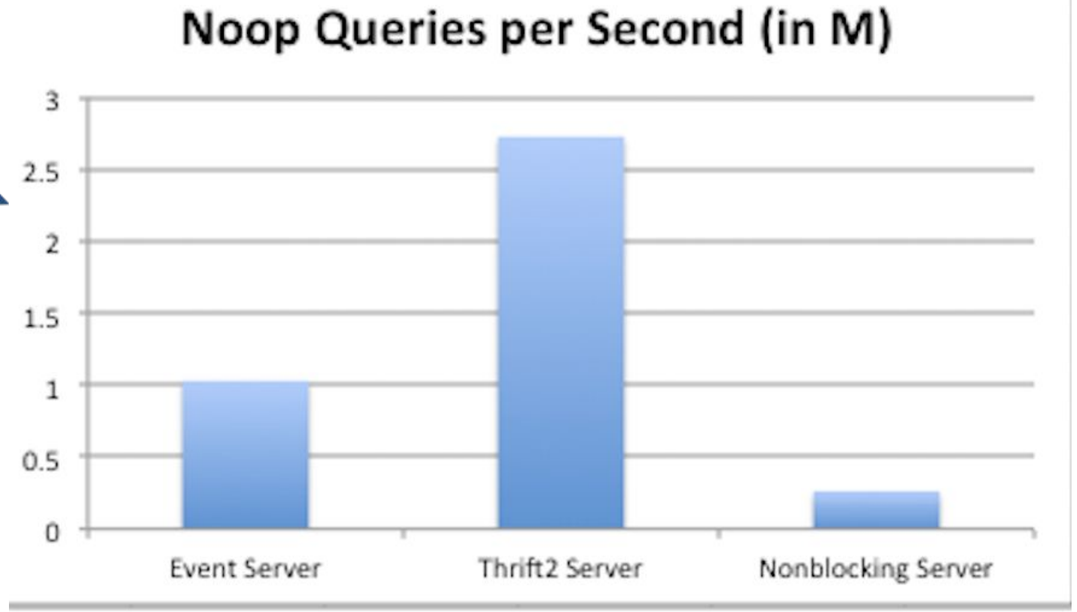


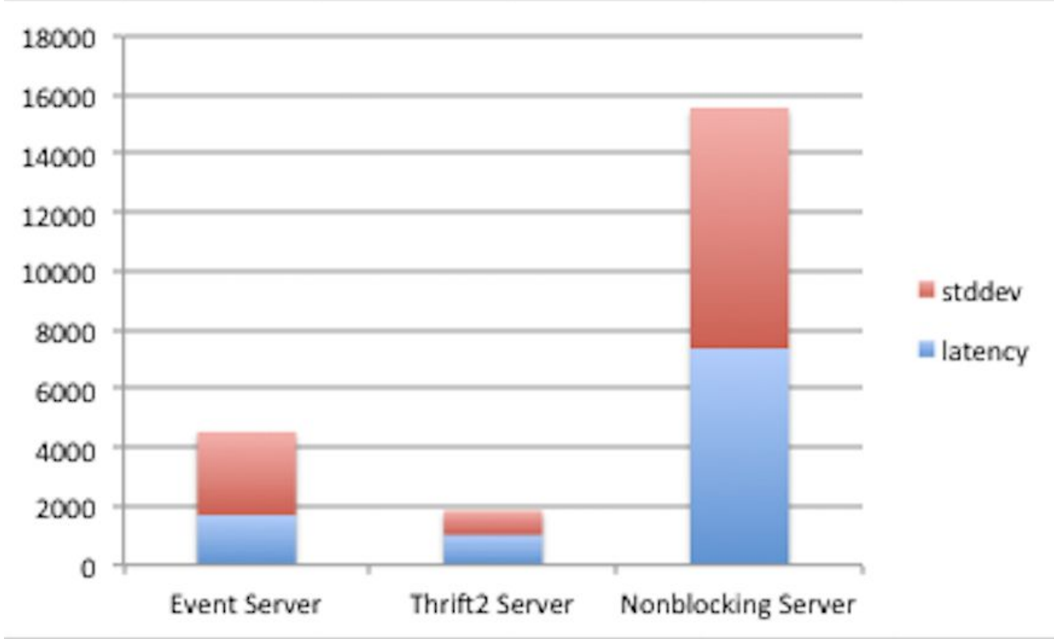
https://bit.ly/3r67vf6

fbthrift

- Even more recent improvements..

🙌 🥳 🎉



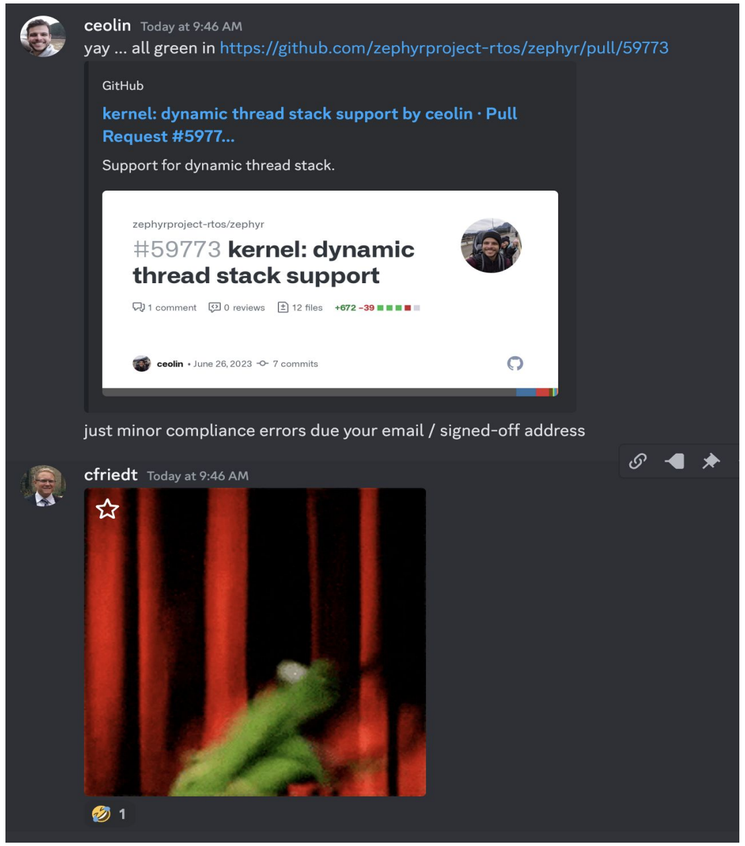


https://bit.ly/3r67vf6

Dynamic Kernel Thread Stacks Working!!11!

- After 4 years of “hacking in my spare time”
- Received this from Flavio **literally this morning**!!
- With / without MMU, with / without Userspace
- Collaboration from Andy, Flavio, Daniel, Anas and others at Intel, Stephanos Ioannis, Keith Packard

🎉 🥳 🙌



Questions? / Feedback


Meta