
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023


KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023

Building proxy with new asynchronous I/O API: Exploring Envoy's io_uring Integration

Xu, Hejie / Xie, Zhihao

Who are we

Xu, Hejie (Alex)

Cloud Software Engineer, Intel / Envoy Maintainer /
OpenStack Nova/Placement Core Reviewer

Alex is a Cloud Software Engineer at Intel and Maintainer of Envoy. Currently, he focus on the Service Mesh data plane and Envoy community. He also has 10 years of experience with OpenStack and IaaS in the past.

Xie, Zhihao

Cloud Software Engineer, Intel

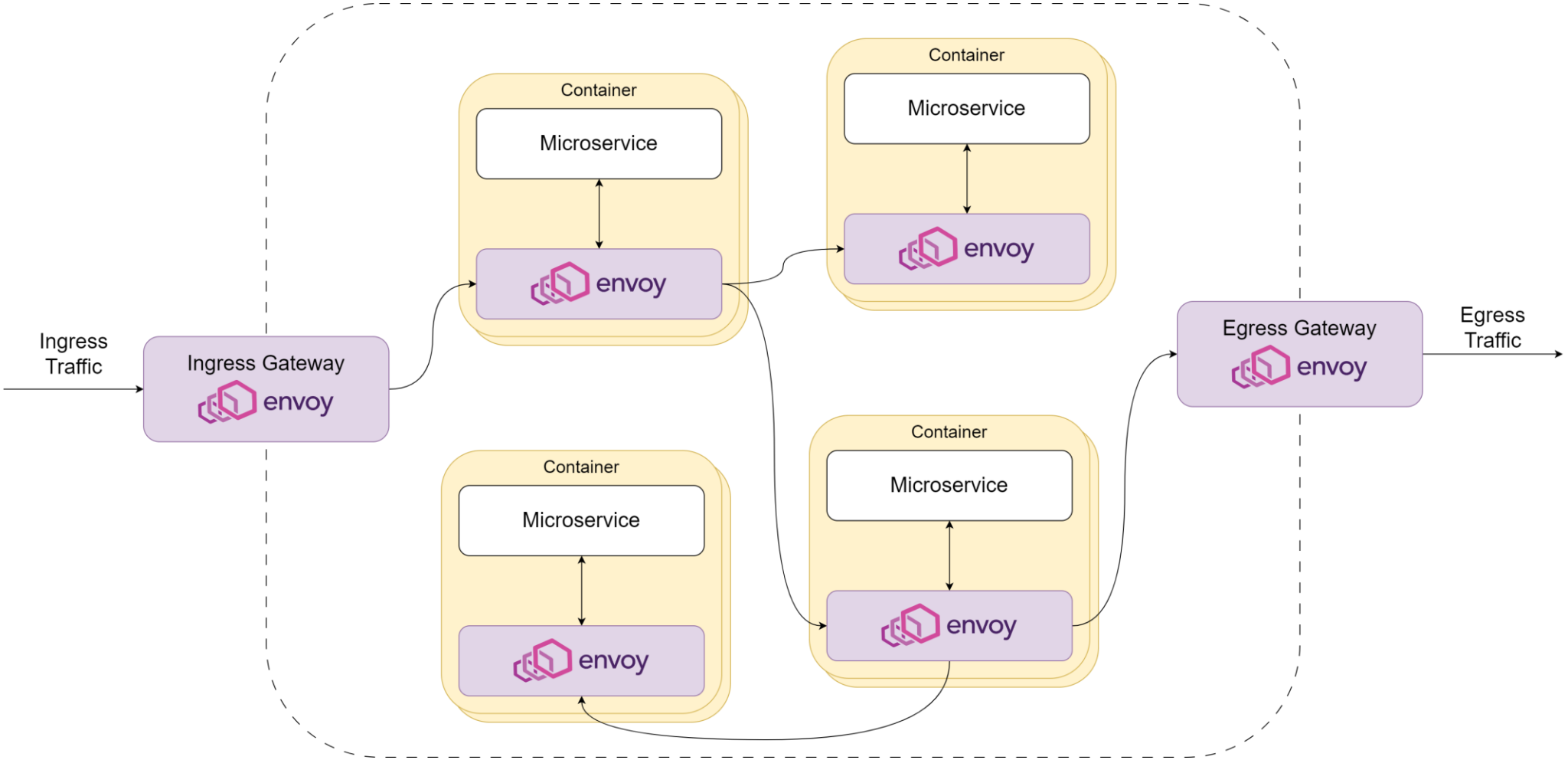
Zhihao is a Cloud Software Engineer at Intel in the Service Mesh team. He works on Envoy, with a focus on optimizing performance in Networking, Load Balancing, Routing and Access Control.

Agenda

- Backgrounds
- Envoy’s I/O Architecture
- Integrate io_uring into Envoy’s I/O Architecture
- API and Benchmarking
- Status Update
- Q & A

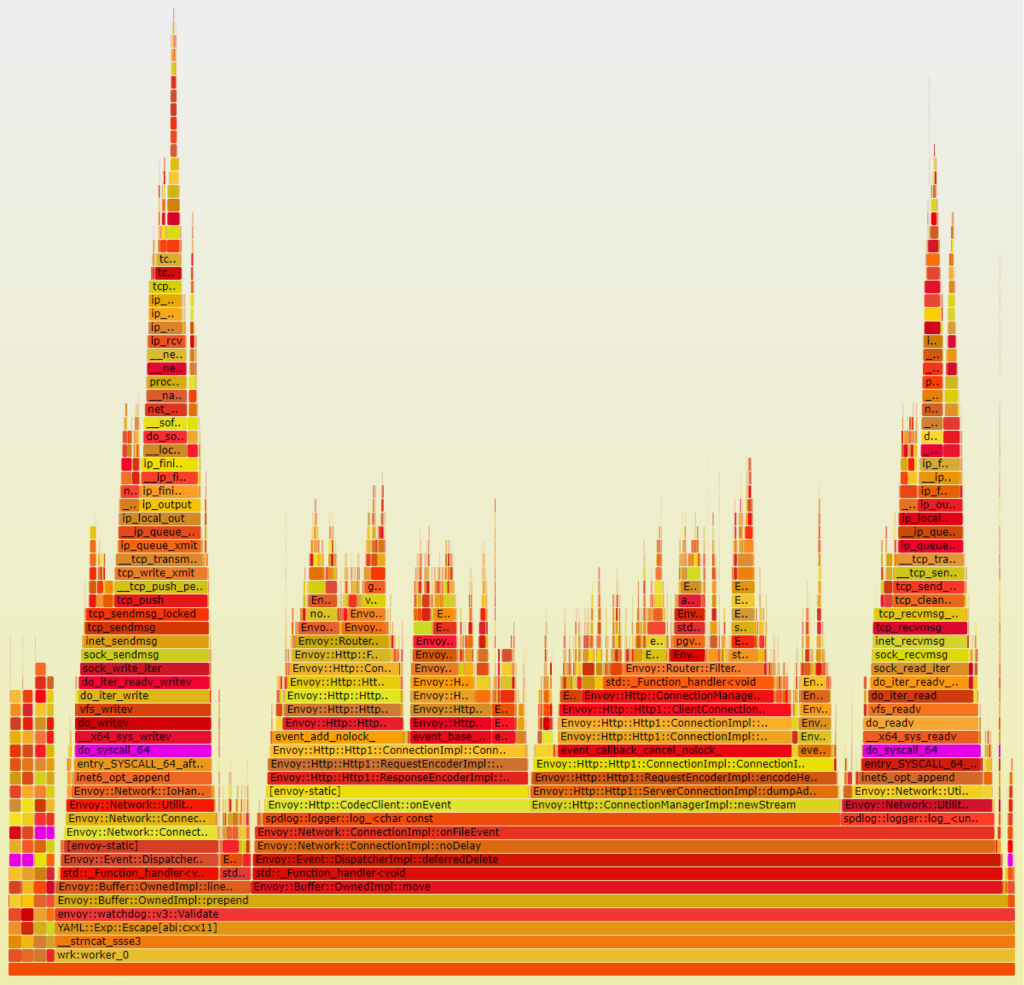
Backgrounds

Envoy is everywhere



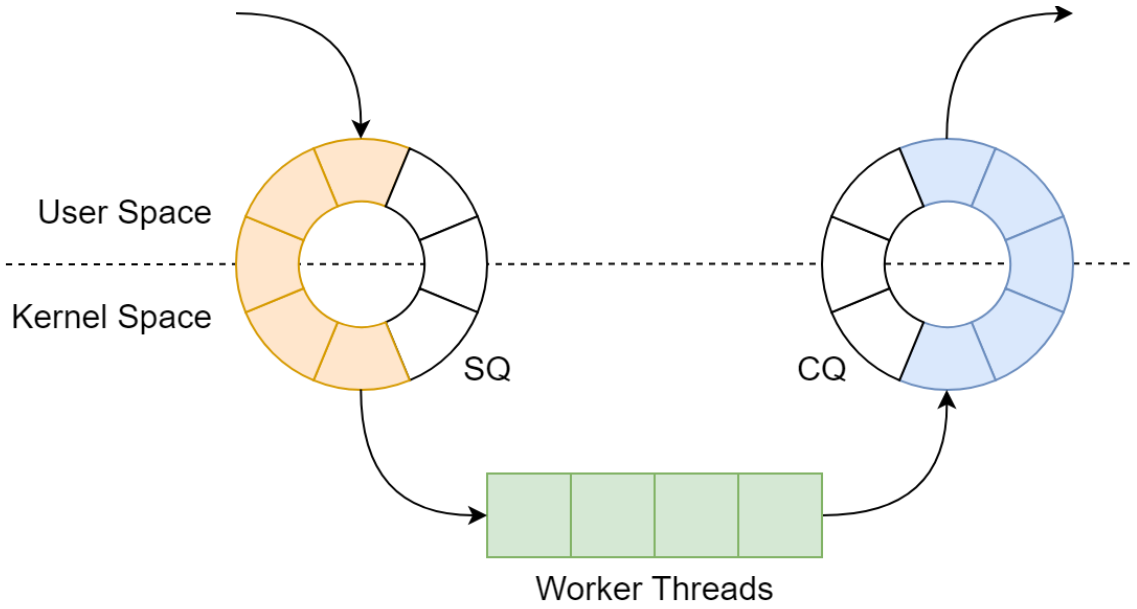
What we found previously

- Nearly 30% of the CPU usage is allocated to read-and-write-related syscall in a simple HTTP router use case profiling
- You can utilize VCL within Envoy but using VCL requires support from an external VPP process which involves extensive tuning efforts



The benefit of integrate with io_uring

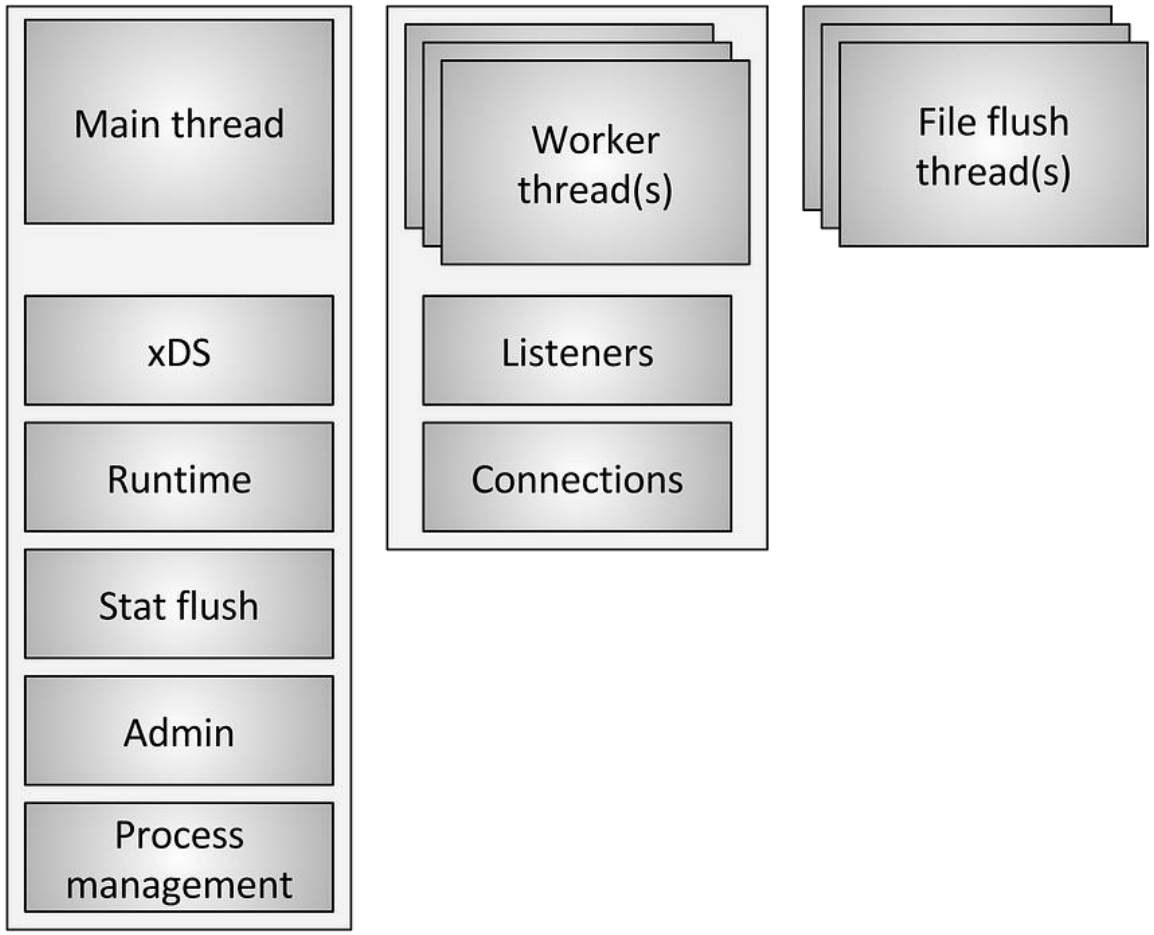
- io_uring is a new asynchronous I/O API where 2 ring buffers are responsible for submission (SQ) and completion (CQ) exchange
- Less syscall, less memory copy
- io_uring operates in a fully asynchronous manner, which is different from Envoy's I/O model



Envoy’s I/O Architecture

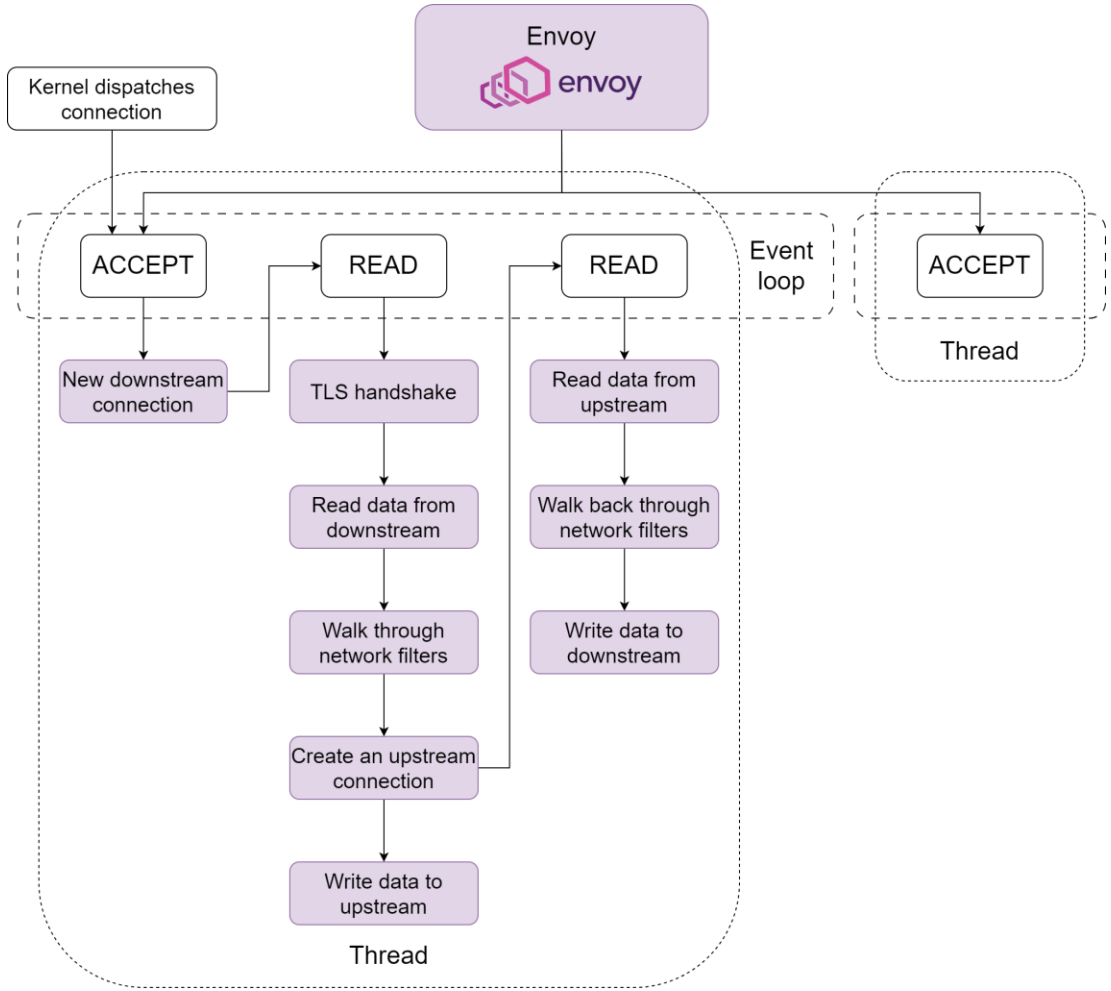
Threading model

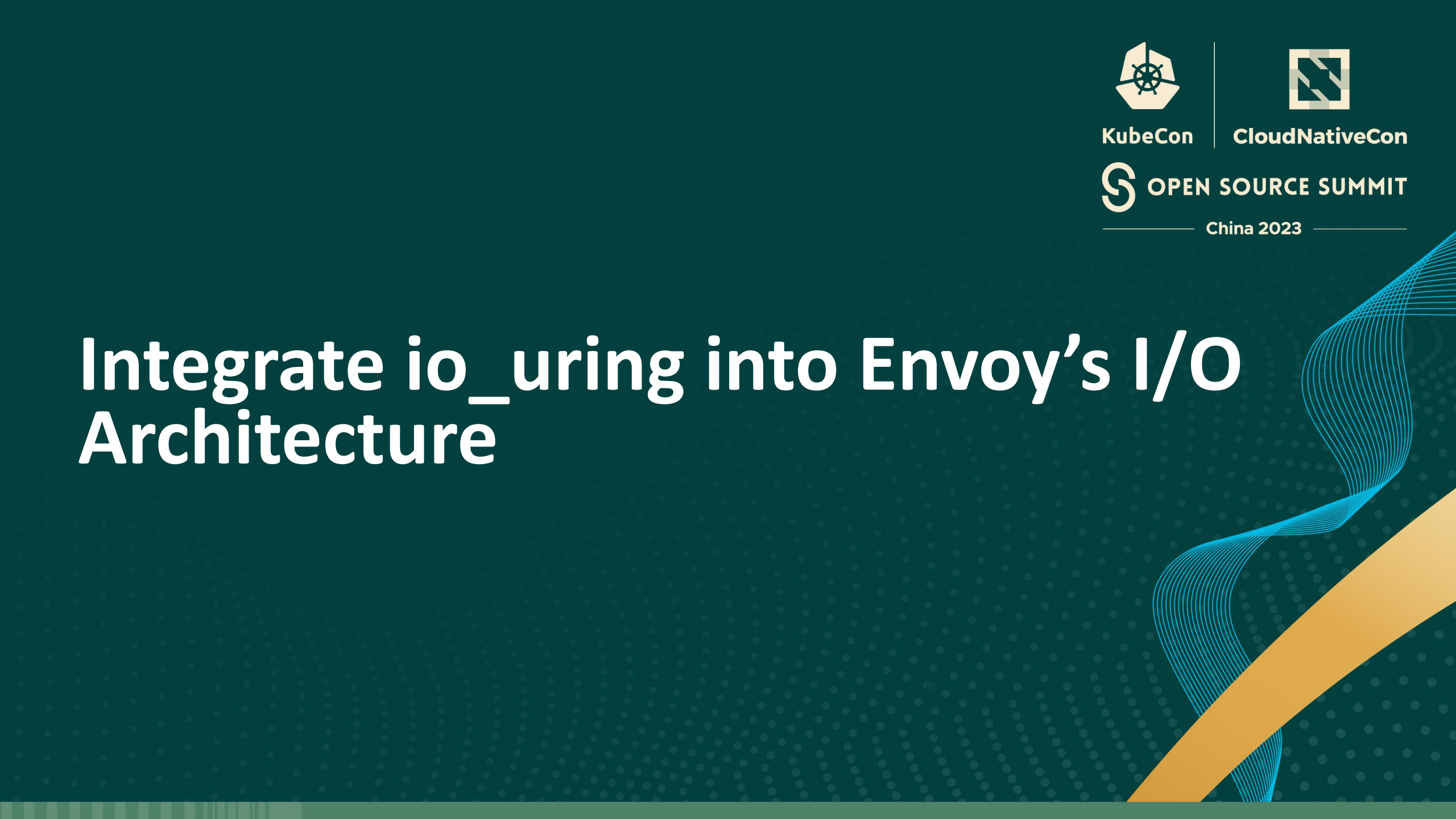
- Single process, multiple threads
- Main thread for control path
- Worker threads for data path
  - Connections are processed by worker threads



I/O model

- The kernel will pick a thread to dispatch new connection, and the connection will be bind to the thread in the whole lifetime
- Each thread is running an event loop for I/O multiplexing
  - libevent-based
- Basically, the connection binds to a single thread in the whole lifetime
  - Sockets register to the event loop of the current thread

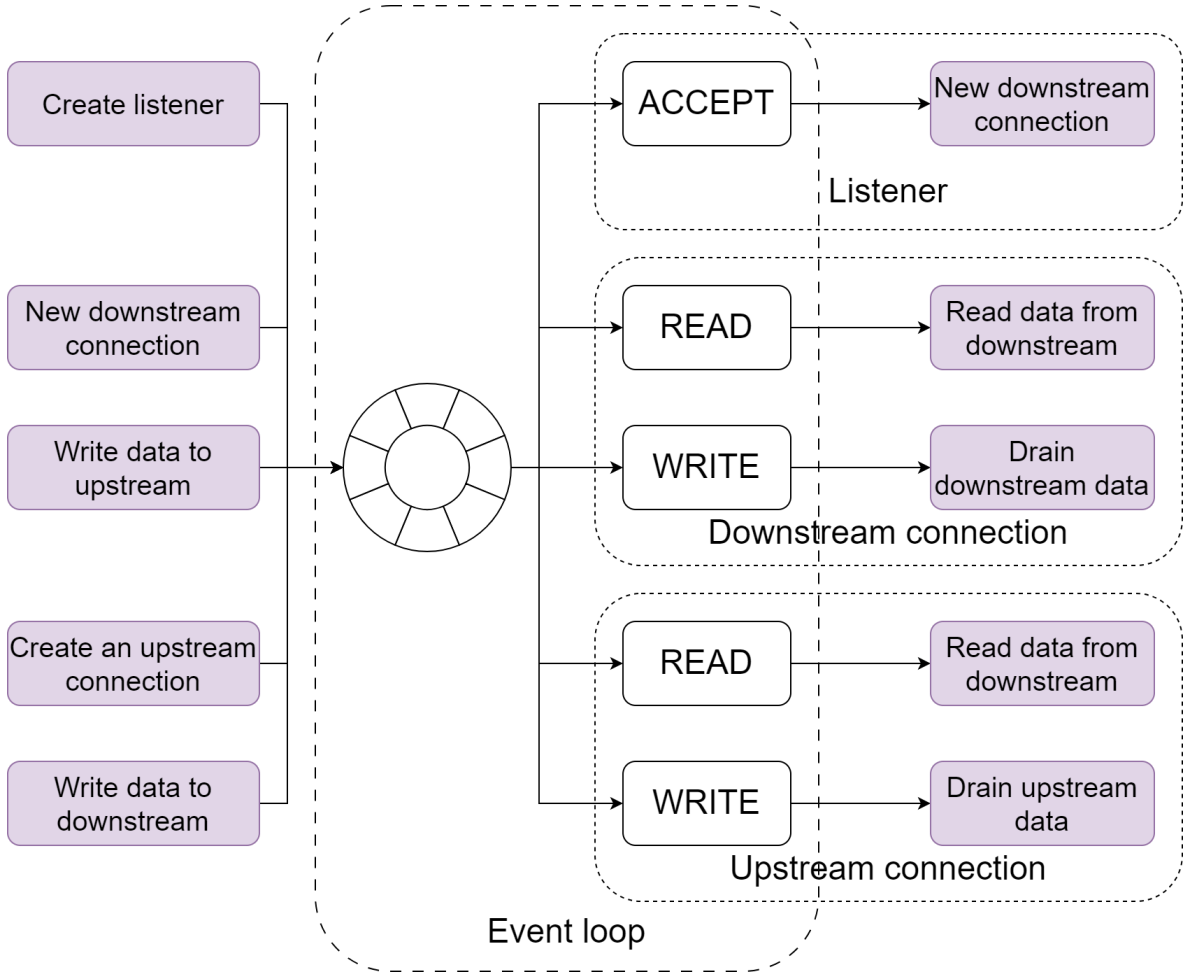




Integrate io_uring into Envoy's I/O Architecture

Integrate io_uring into the event loop

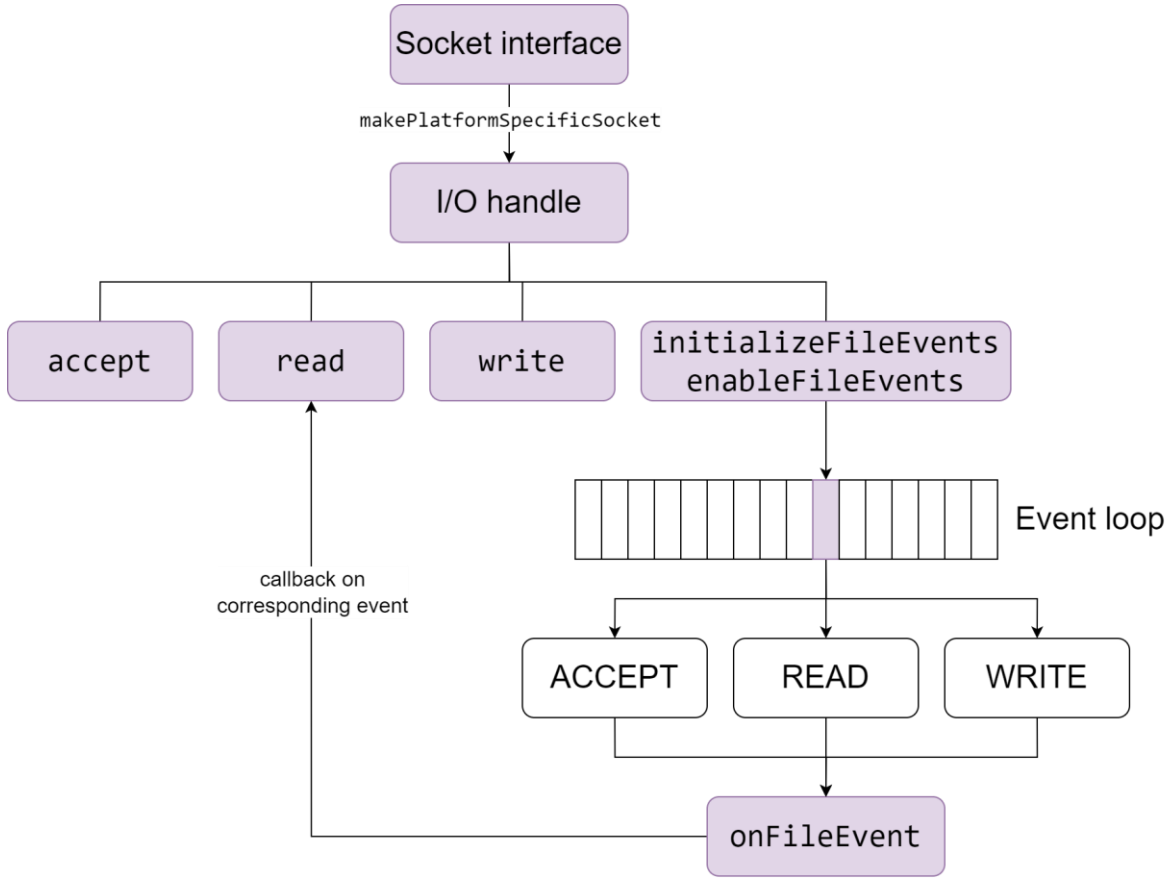
- libevent is responsible for the event loop in each thread which does not support io_uring natively
- liburing supports eventfd and whenever completions are posted to the CQ, then we know there are completion queue events (CQE) need to be processed
- Using the eventfd to bridge the libevent event loop and io_uring



https://github.com/libevent/libevent/issues/1019

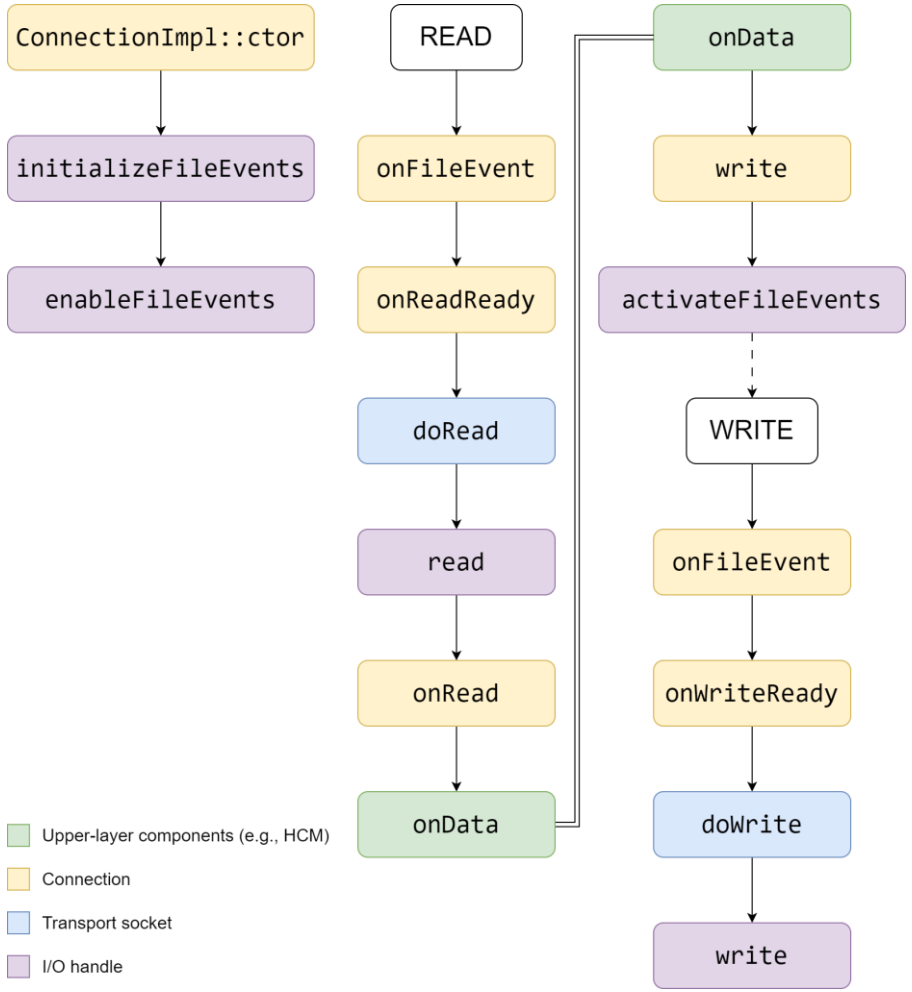
Socket interface

- Socket interface determines which I/O handle should be used
  - Default
  - VCL
  - Userspace
- We change the default socket interface to support the io_uring
  - Detecting the Linux kernel capabilities to enable the io_uring
  - Initialize the io_uring for each thread



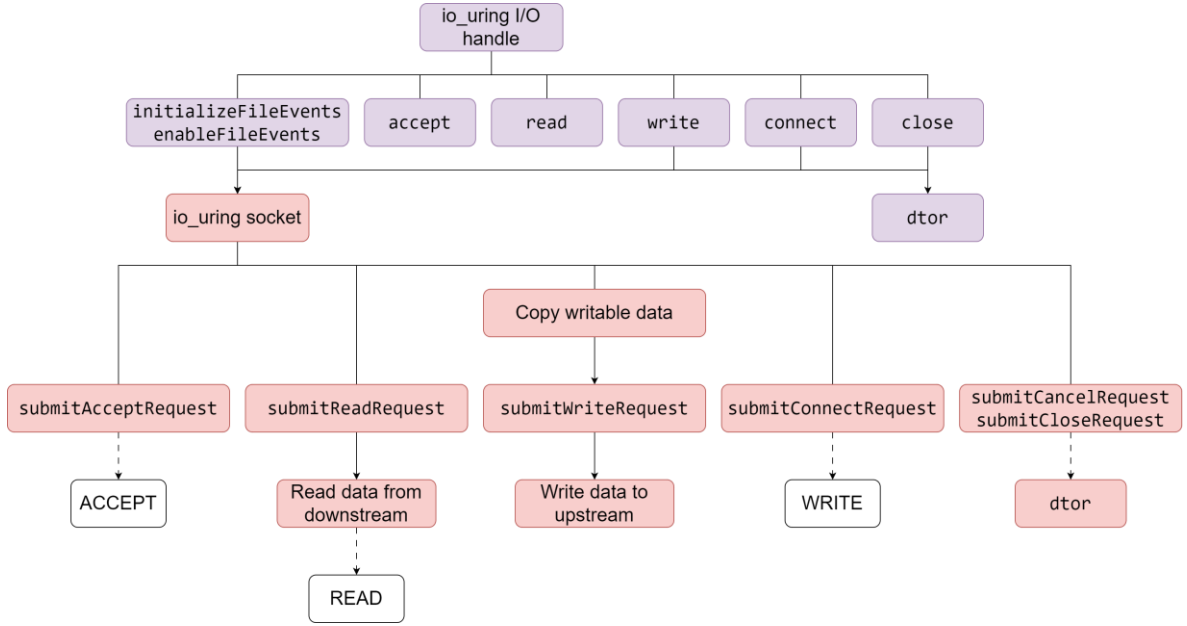
I/O handle

- I/O handle is the abstraction of I/O API
- Complicated in the full workflow, but socket-agnostic for all upper-layer components
- I/O models between socket API and io_uring API are different
  - E.g., write in socket API will return bytes written in time synchronously while io_uring is always asynchronous



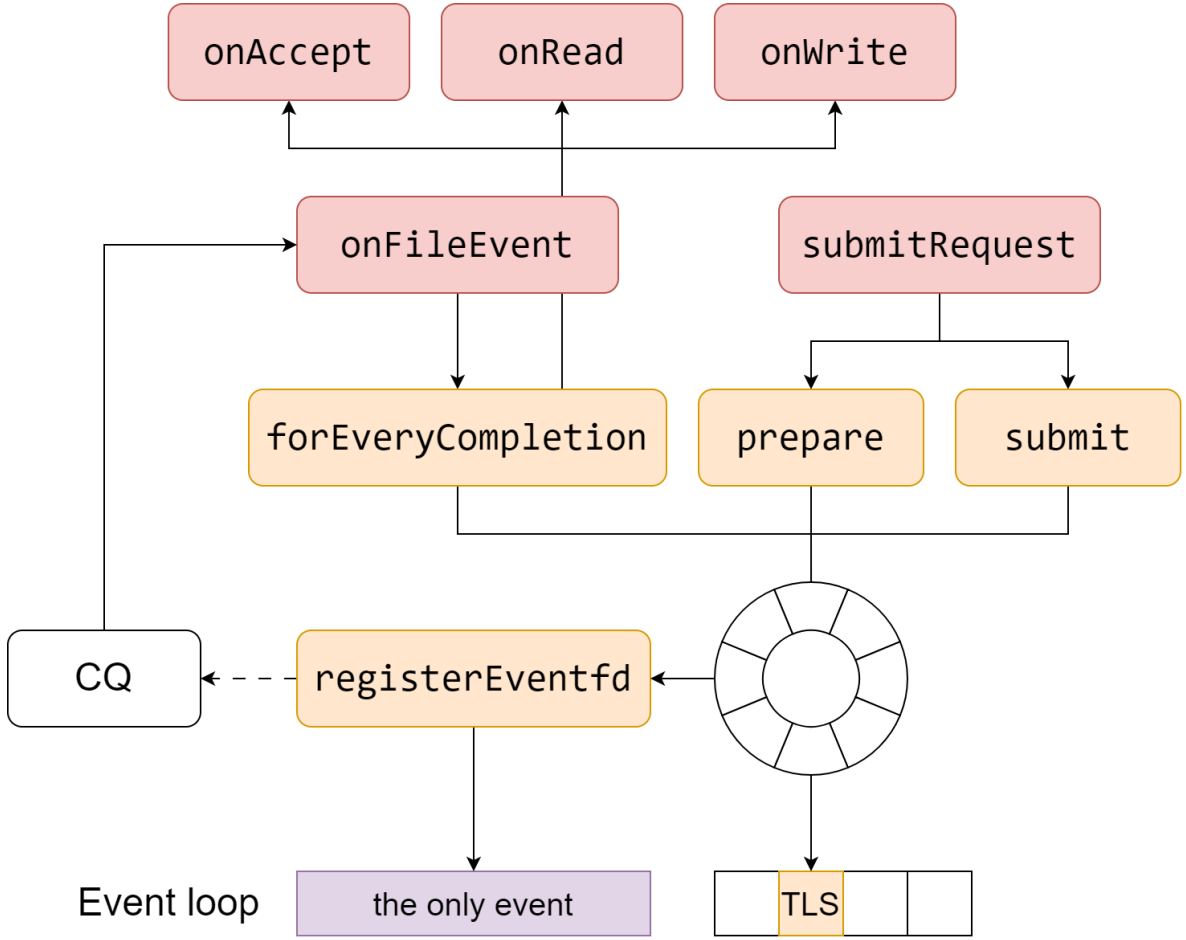
io_uring socket

- io_uring I/O handle aligns with I/O handle interface and communicates with io_uring socket
- io_uring socket manages the lifetime of its inner socket and lives longer than I/O handle, and is responsible for submit request to and process completion from io_uring

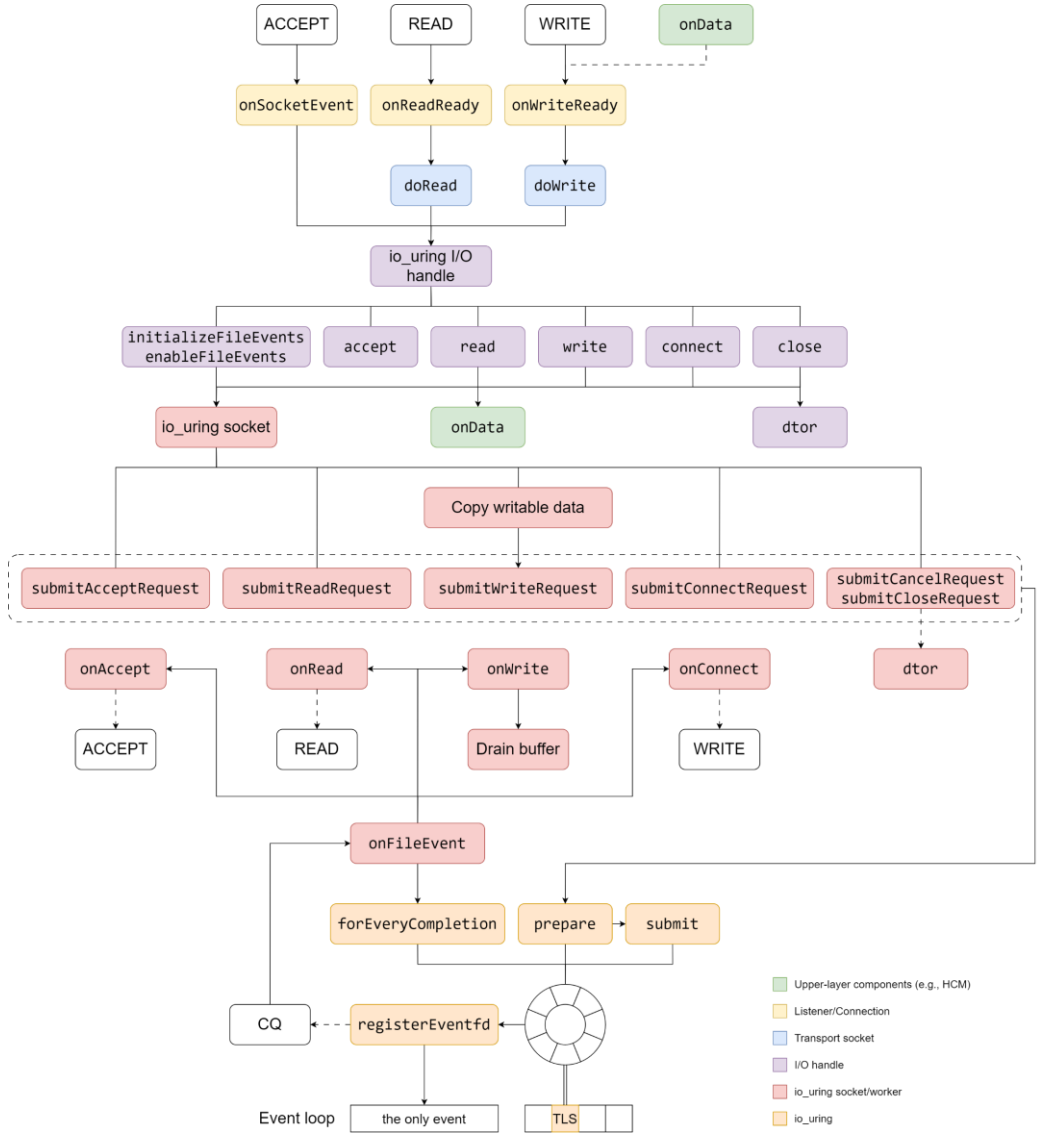


io_uring worker

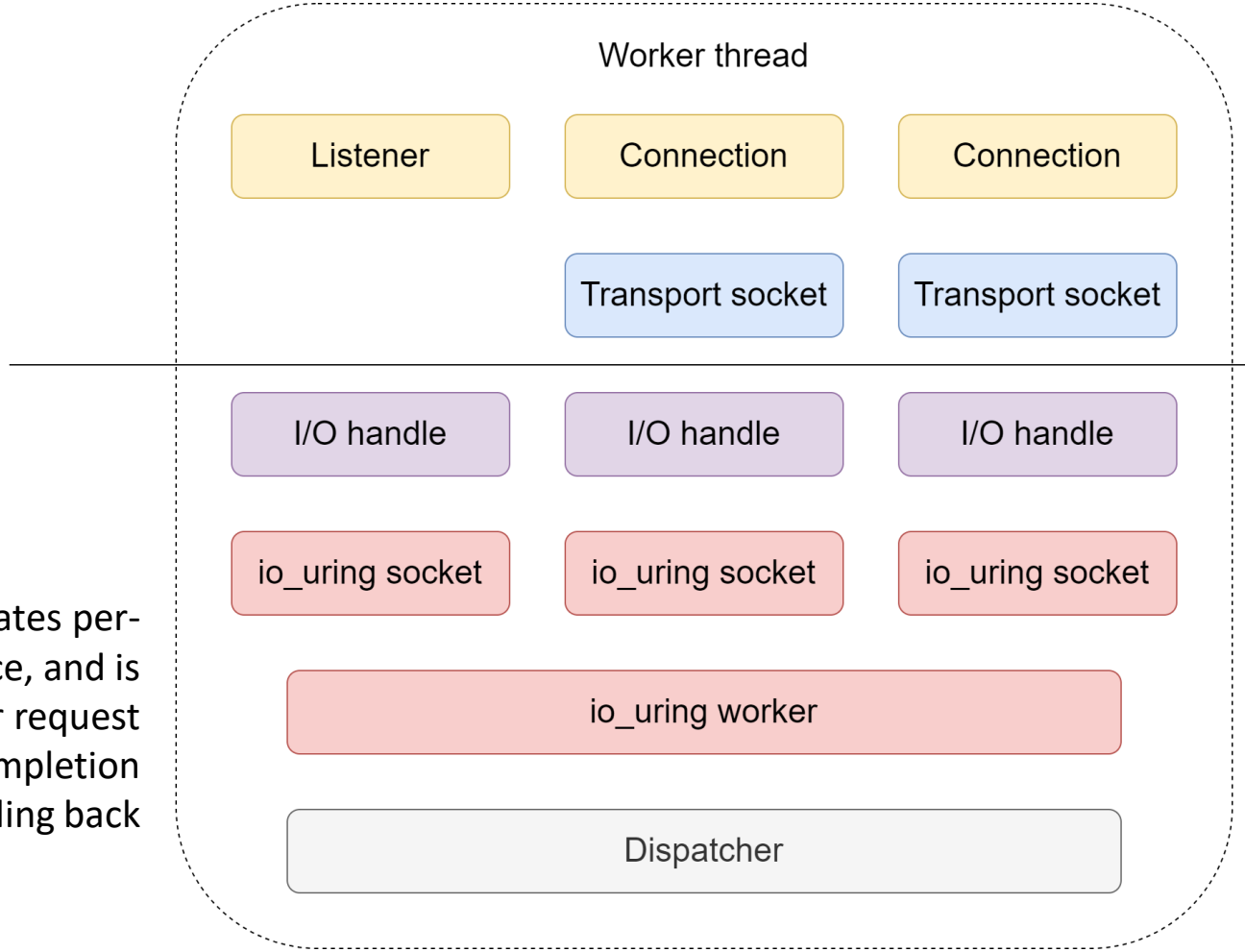
- It is a kind of event loop of io_uring
- Each thread has an io_uring worker and an io_uring instance, guaranteed by TLS, which registers the only event in the event loop
- io_uring worker manages and communicates io_uring sockets in the thread



Put everything together…



...In a higher level



The upper layer does not aware of io_uring

I/O handles mimic socket API behavior

io_uring sockets manage sockets lifetime and requests

io_uring worker creates per-thread io_uring instance, and is responsible for request submission and completion calling back

io_uring worker registers itself as an event in the dispatcher

Design doc at https://docs.google.com/document/d/17M15J6vh7ZzoJPCY65j6HNoLSqNEoWnQ3Oa9oKIru0o

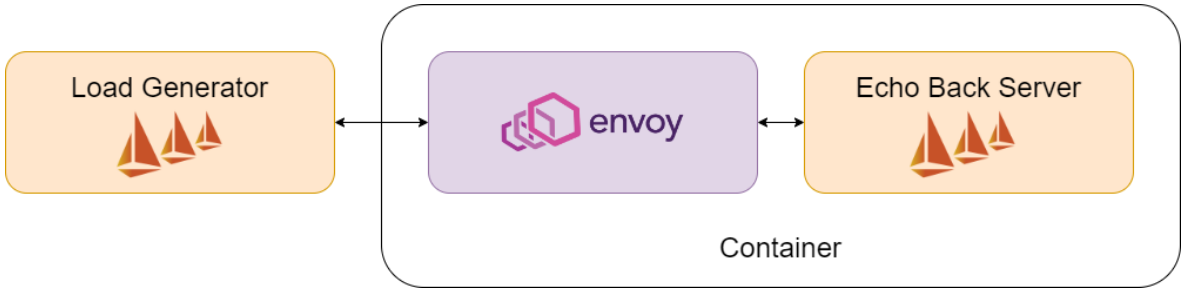
API and Benchmarking

How to enable io_uring in Envoy

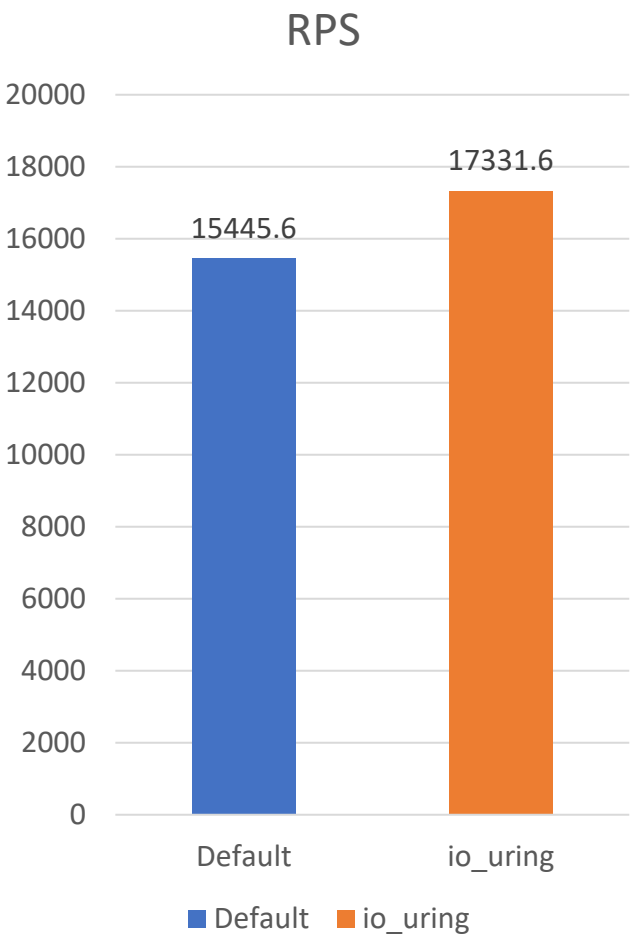

```
default_socket_interface: "…default_socket_interface"
bootstrap_extensions:
- name: …default_socket_interface
  typed_config:
    "@type": …DefaultSocketInterface
    enable_io_uring: true
    io_uring_size: 300
    accept_size: 5
    read_buffer_size: 8192
    write_timeout_ms: 1000
    use_submission_queue_polling: false
```

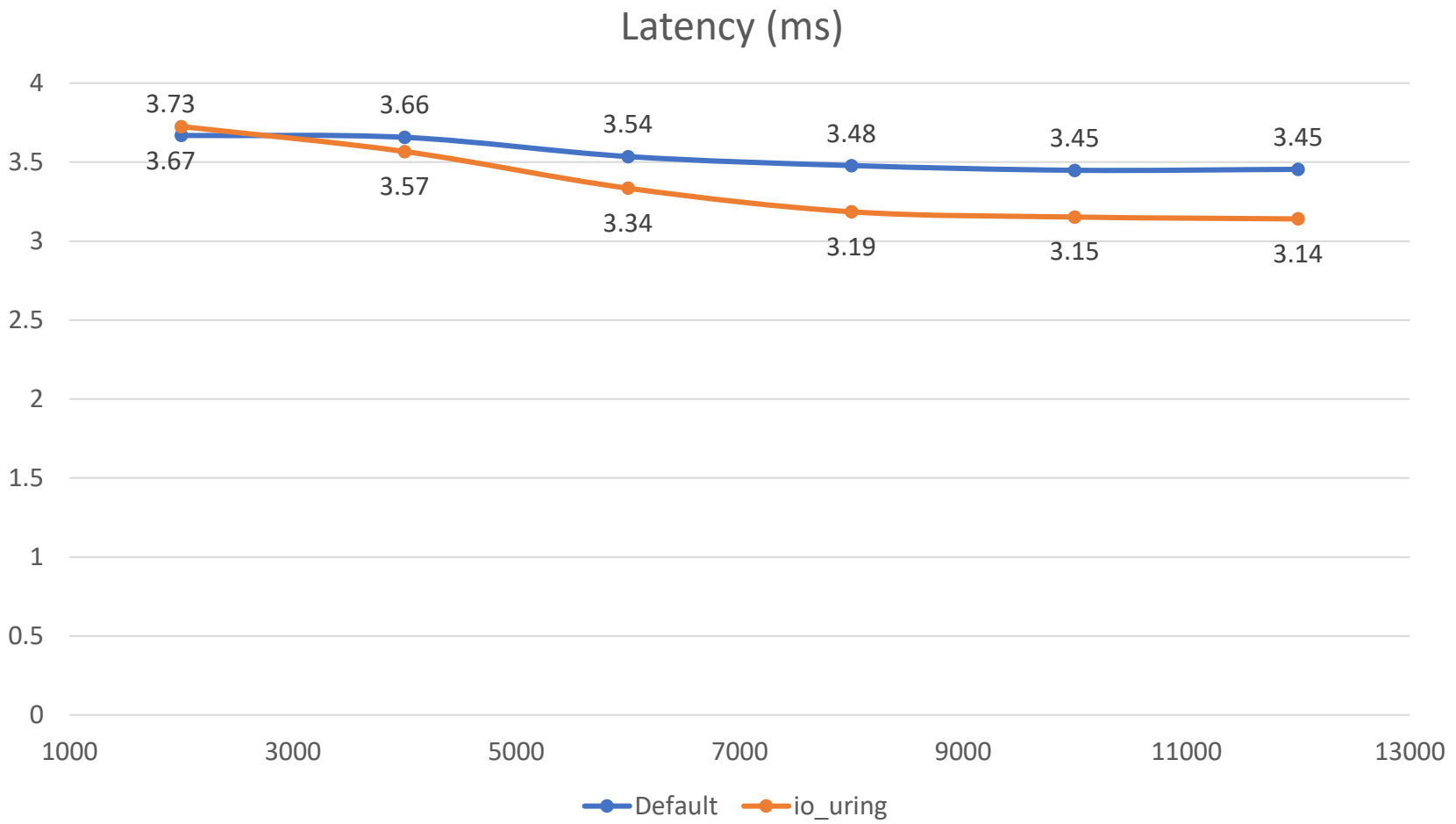
Workloads and configurations

- Envoy 6050489 / 1.27.0
  - Single thread
- Fortio load 1.34.1
  - 64 connections, unlimited QPS, 4kB payload POST
  - 64 connections, QPS ranging from 2000 to 12000, 4kB payload POST
- Fortio server 1.59.0
  - Echoing back



Performance





Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex

Status Update

Community status

- All the issues leading to unstable status have been addressed, and the CI shows all green now in the third iteration
  - Envoy has over 800 integration tests, and in the first iteration, we failed 90% of all. Most of them are segment fault and timeout
  - Thanks to the abstraction of io_uring socket, we can isolate the implementation and test them separately
  - Some tests do not work well for io_uring and need to be rewritten
- Begin the merge process in the community. Have merged 30% code into the codebase so far 🎉
  - We target to support TCP in the first complete implementation

Pull Request at https://github.com/envoyproxy/envoy/pull/28824

What’s next

- Community
  - Continue merging the rest of the code
  - There are some known issues need to be addressed
- Performance
  - Parameter tunning
  - SQ polling, fixed buffers, zero copy, and features in the latest kernel
- Dispatcher replacement
  - Replace libevent with io_uring


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Q & A

Notices and Disclaimers