Under the hood:

Exploring Istio's lock contention and its impact on Expedia's Compute Platform

KubeCon | CloudNativeCon

North America 2023

expedia group™

Timothy Ehlers

Senior Engineer

Started with Orbitz in 2009

OpsDev

Raghav Grover

Senior Software Engineer

Dev first

Contributed to Crossplane, Karpenter and Istio

KubeCon | CloudNativeCon

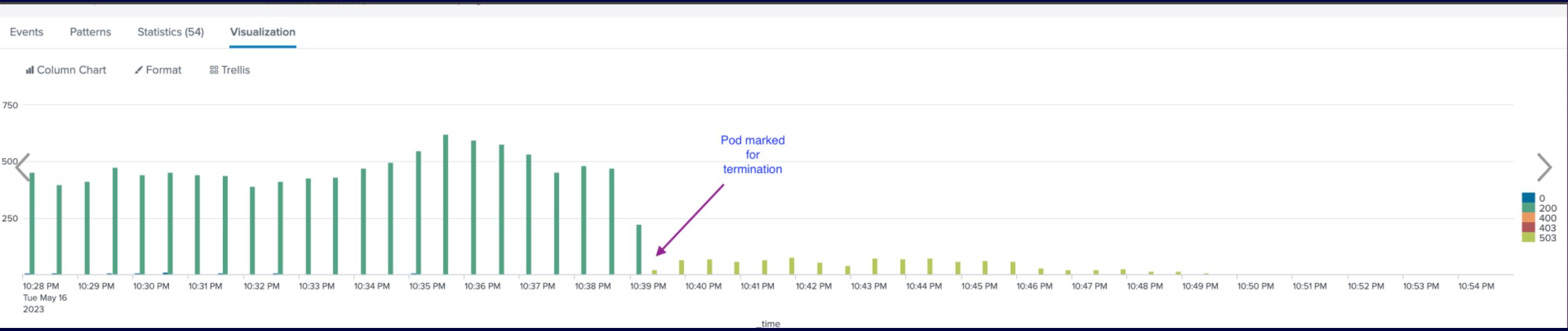
North America 2023

expedia group™

agenda

- Issue and its cause
- Istio architecture
- Plausible bottlenecks
- Istiod deep dive
- Remote debugging
- Lock contention and fix
- Workaround
- Telemetry

expedia group



The issue

Ingress routed traffic to a dead pod even after 9 mins of Pod's termination resulting in 503s

expedia group™

Cause of the issue

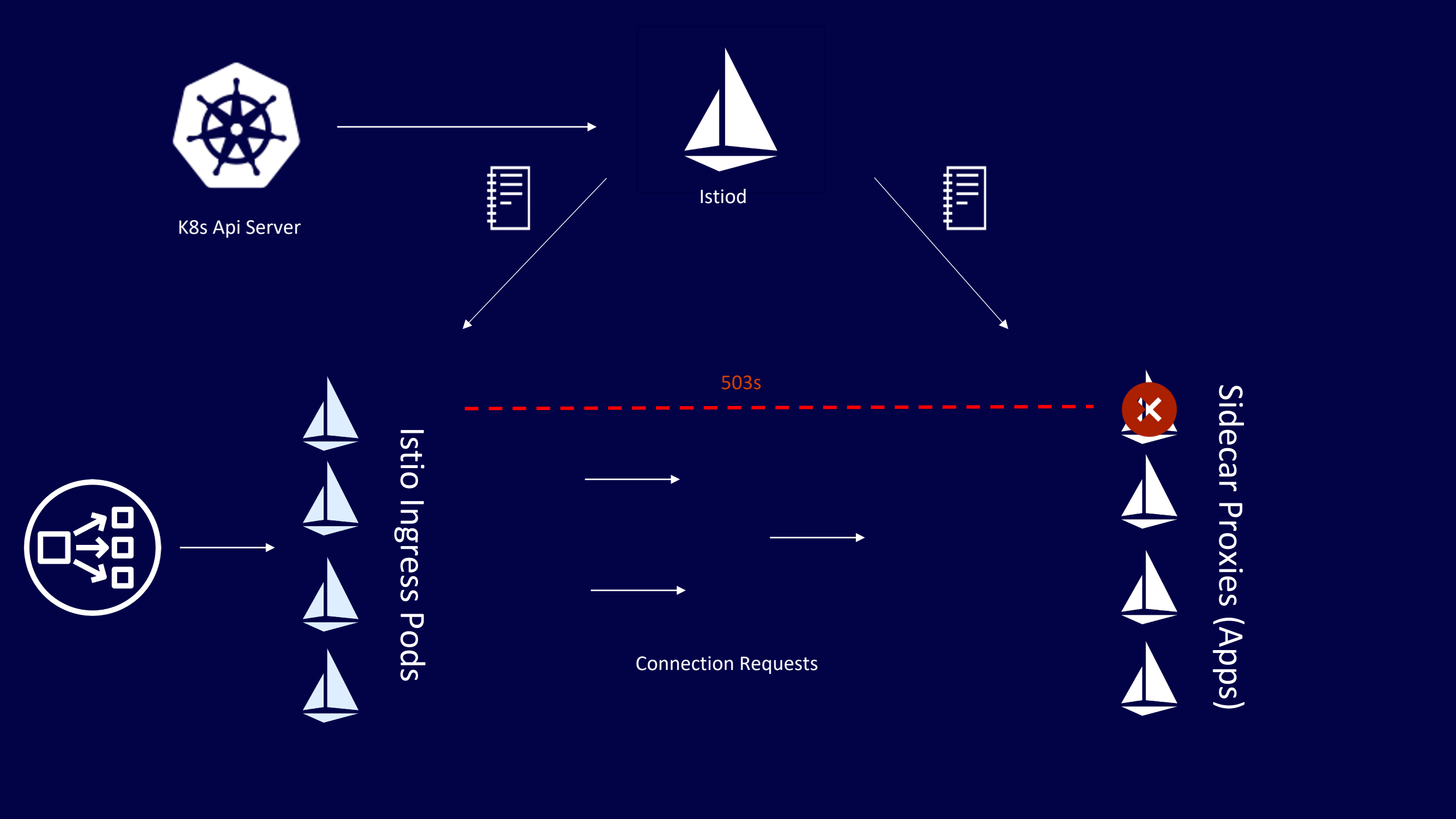
- Scale down of a HPA of a big deployment
- Accompanied by a release of deployment containing 1000+ replicas.

Pod churn

expedia group™

Istio architecture

Step by Step


K8s Api Server
Istiod
503s
Istio Ingress Pods
Connection Requests
Sidecar Proxies (Apps)

Pushing the limits

- 23,000 pods
- 2000 nodes
- 500 -> 1500 istiod
- Only 15 pods assigned per istiod
- Istiod burning 2500 cores
- And it still didn’t work!!!

expedia group™

Plausible bottlenecks

What all we considered could be the root cause but didn’t turn out to be one.

expedia group™

Are K8s informers delayed ?

Delayed informers would cause the configuration
and state of the cluster to go stale.

```
if pod.Name == "test-pod" {
    dt := time.Now().Unix()
    labels := pod.GetLabels()
    i, _ := strconv.ParseInt(labels["createtime"], base: 10, bitSize: 64)
    dif := dt - i
    fmt.Printf( format: "Pod Event took %ds to show up\n", dif)
}
```

- We setup a container with an informer to watch for a test pod
- Test pod was re-created every 20s
- The informer reported the pod within milliseconds

Is endpoint controller impacted ?

Affected endpoint controller would not be able to process the changes to events fast enough!

- We fed AWS audit logs through a custom script to plot the number of healthy endpoints at any given point in time.
- That matched with our Dashboards of pod count in the cluster, indicating no lag from Endpoint controller!

expedia group

Is proxy config too big ?

Let's try namespace isolation!!

- Namespace isolated the biggest deployments
- Looked at EndpointSlice labels. (we have 12 of them)
  - More labels == more processing by the parser in istiod
- Why does a restart resolve things immediately?

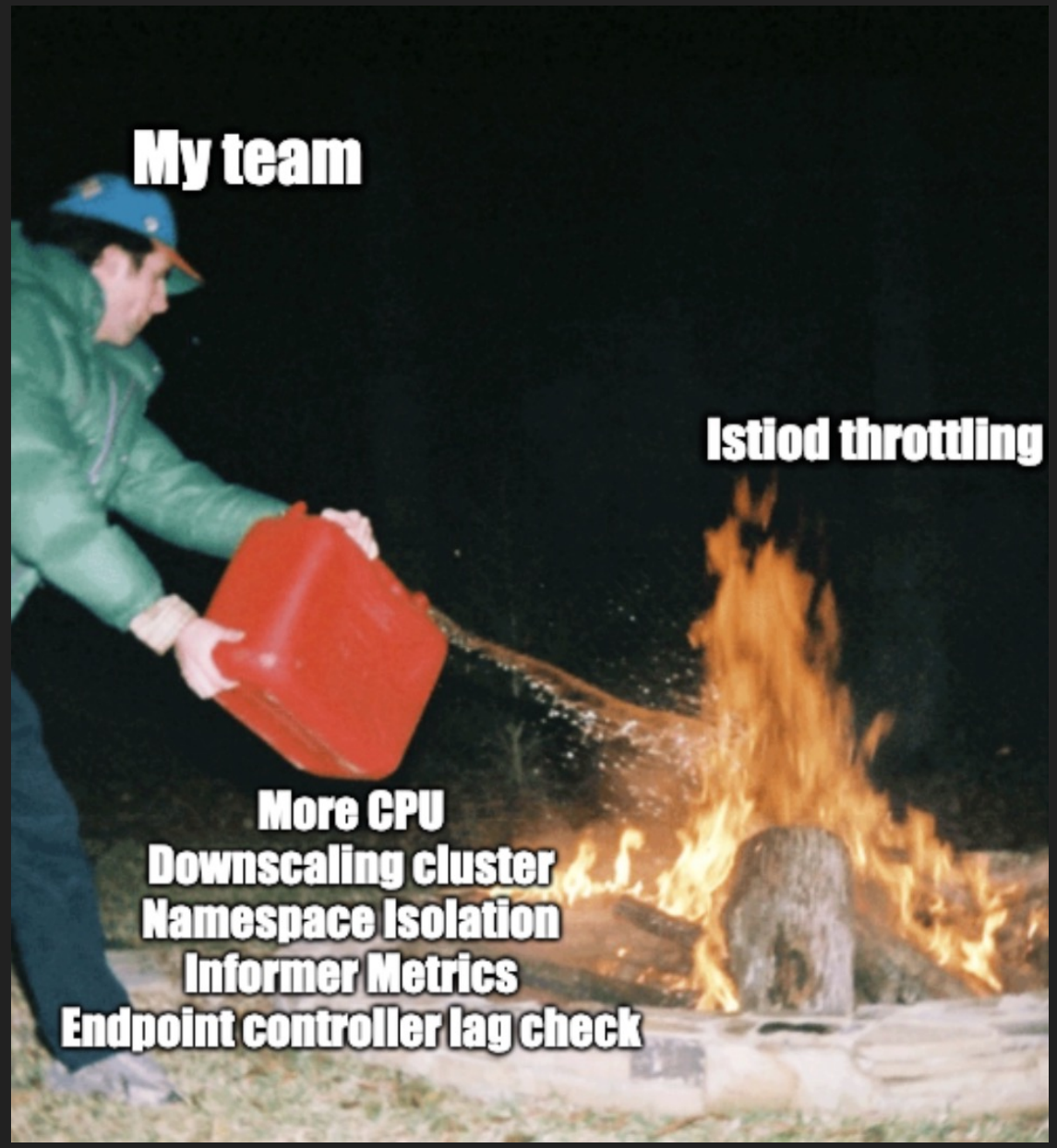
expedia group

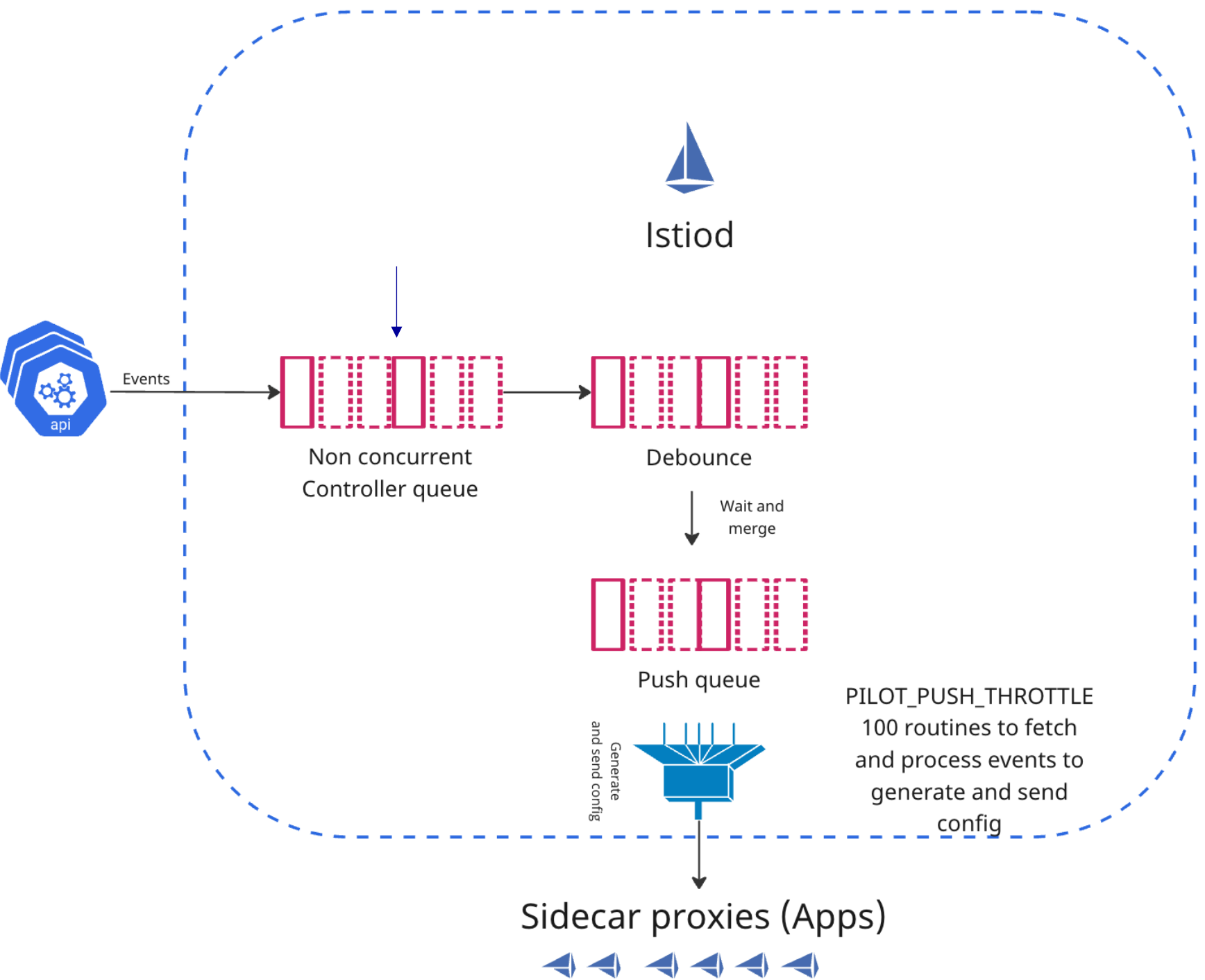
Still burning!

After ruling out all other things, all eyes on

Istiod.

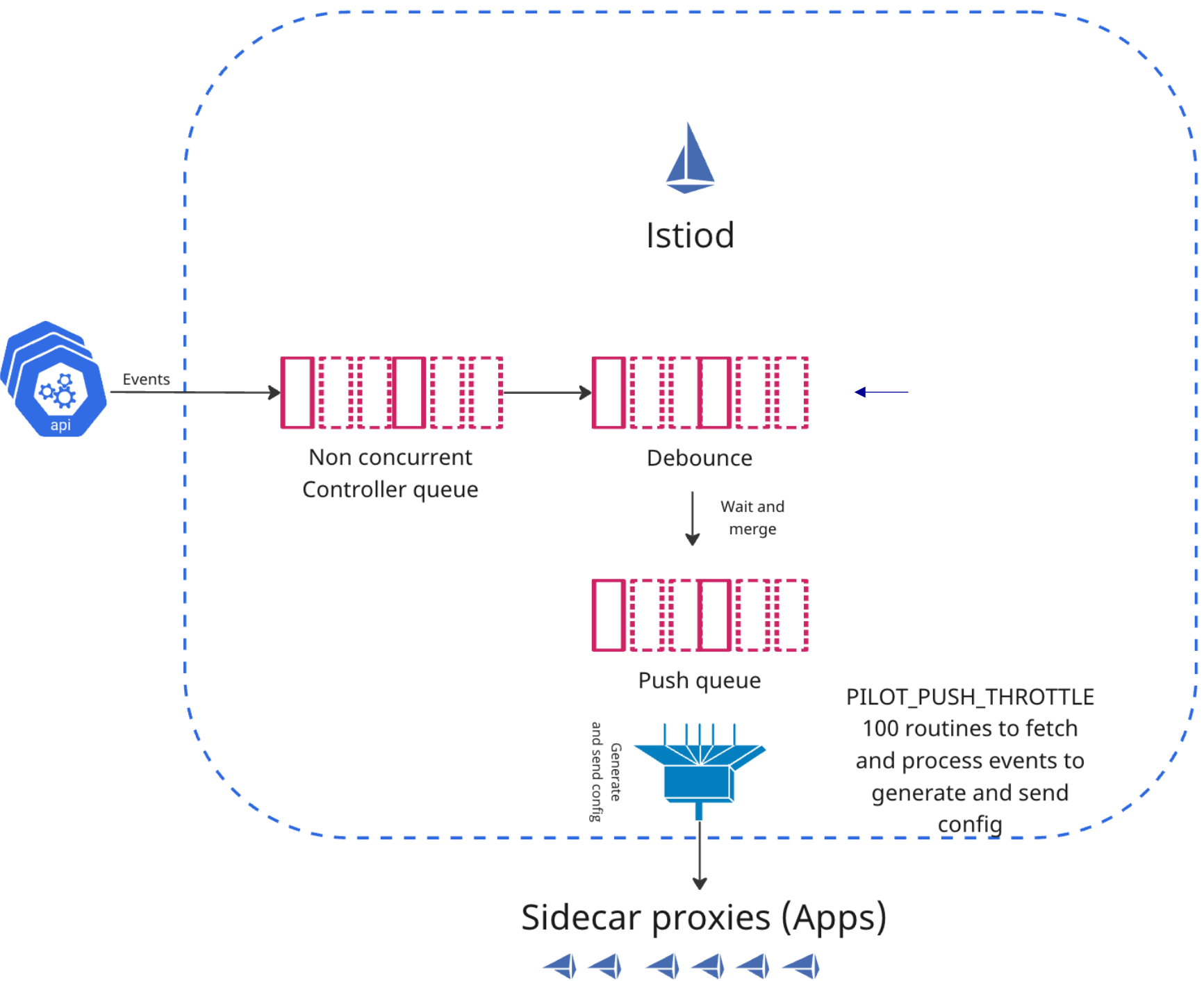
Let’s dive into its internals.





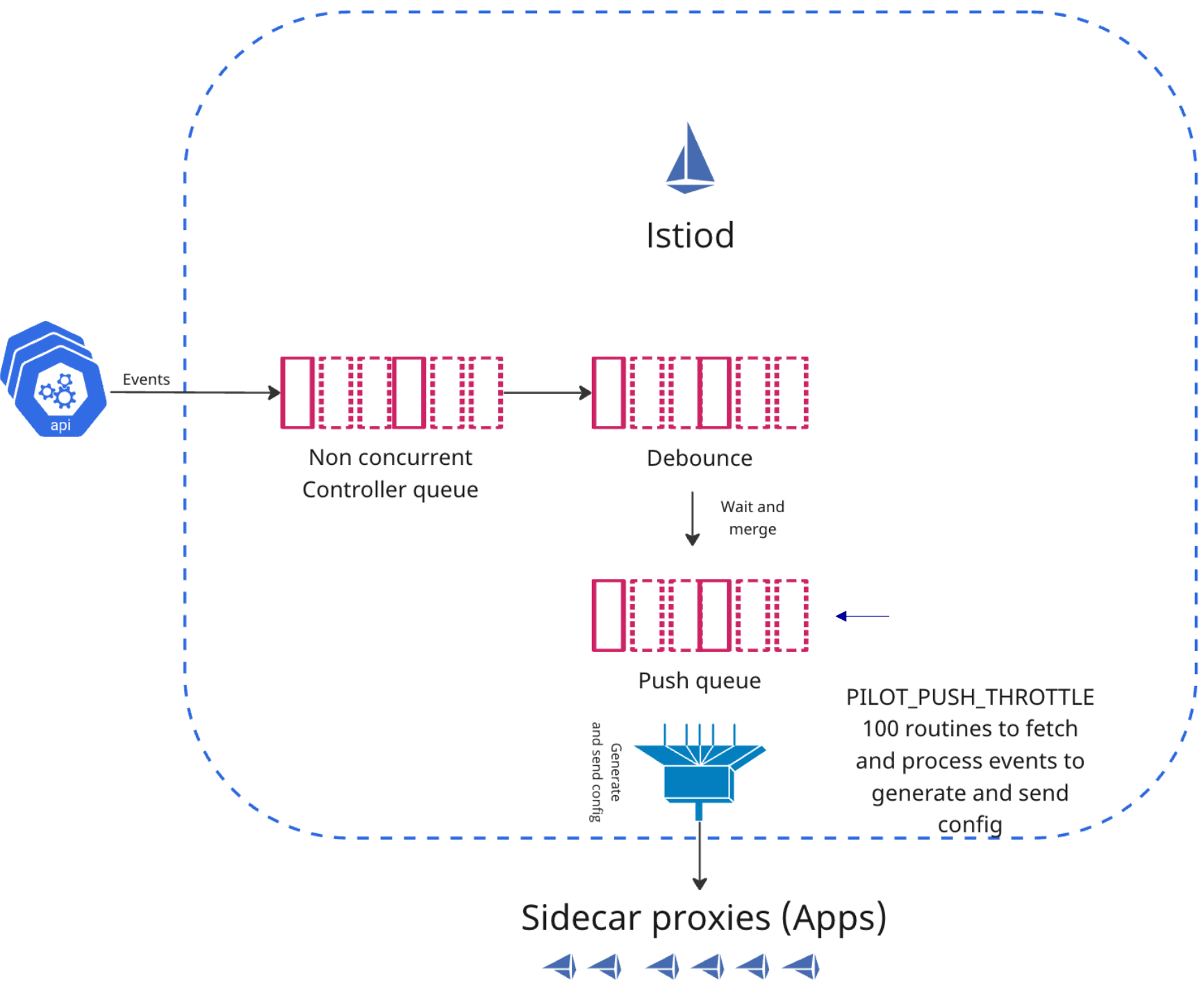
Controller Queue

Events from K8s API server via informers are put into this non-concurrent queue. The informers include watches on Pods, services, endpoint slices, nodes, etc.



Debounce

Events are grouped together to limit the number of Push requests, generating proxy config is CPU intensive task!



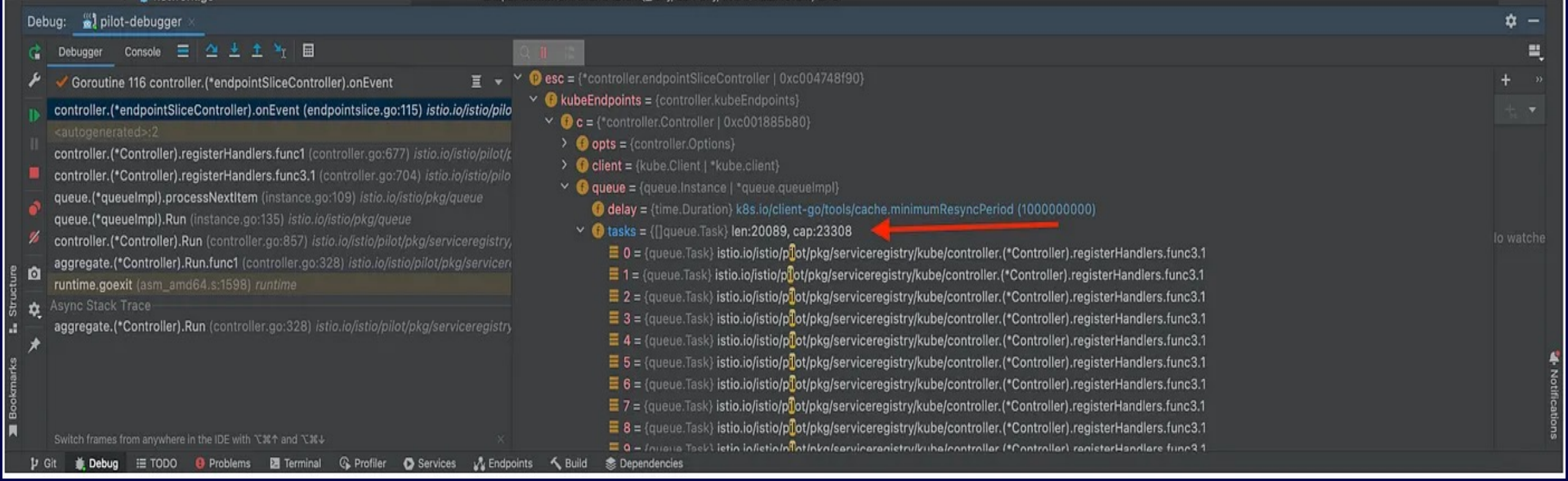
Push Queue

After context gathering, events are put into this queue as Push Requests. By default, 100 go routines are processing to clear off the Push queue and send the generated proxy config to sidecars containing all the routing information!

Remote debugging

Istio documentation is awesome. We built our own Istiod custom image and deployed it to our clusters, but this time we could remotely debug pods right from our IDE!

expedia group™



Controller Queue

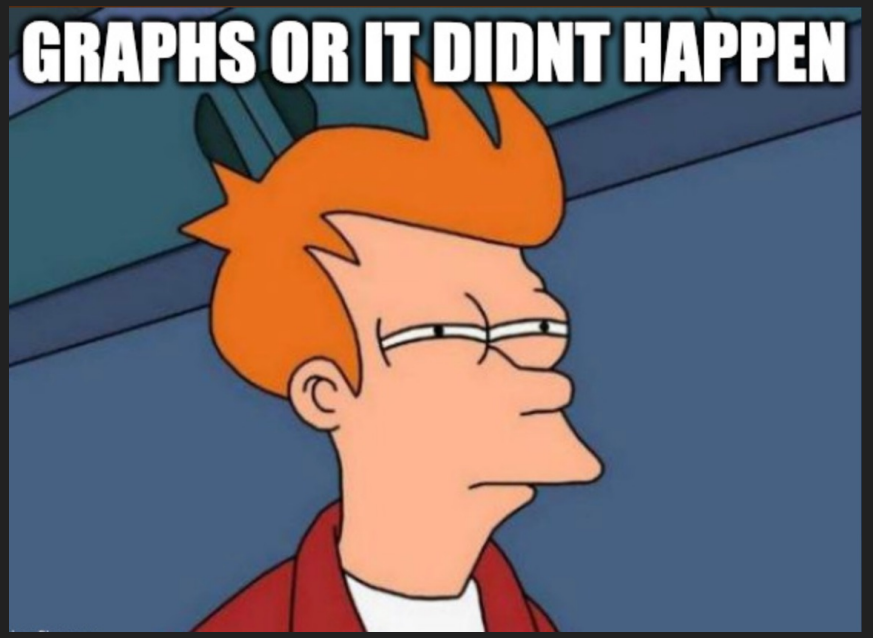
Observed around 20K items in the controller queue which took a lot of time to clear signifying a contention!

expedia group

Limited observability

Controller queues had limited observability, it had logging on errors but no metrics which could have pointed us to this direction.

But now we know where the blockage is so let’s dive into some code.



expedia group™

What does Controller queue exactly contain ?

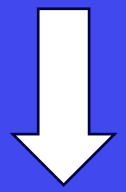
- Contains callbacks generated from informer’s change events.
- Callbacks are functions which get executed and contains logic to process each item depending whether item is pod , node , endpoint, service or something else.
- We had no way of telling the type of event callbacks that were piled up.
- Pod / Node / Service / Endpoint / other ?

expedia group™

Uncovering Lock contention!

We modified Pilot code to push enriched version of the callback with the type of the object and the timestamp when it was inserted in the queue.

```
c.queue.Push(func() error {
    return wrappedHandler(nil, obj, model.EventAdd)
})
```



```
c.queue.Push(&queue.EnrichedTask{
    //Event push time
    Start: time.Now(),
    // Unchanged Task field
    Task: func() error {
        return wrappedHandler(nil, obj, model.EventAdd)
    },
    //otype is the type of the K8s object - Pods, service, endpointslice, node
    Type: otype + "-create"})
```

Enriching the callbacks with a struct on push.

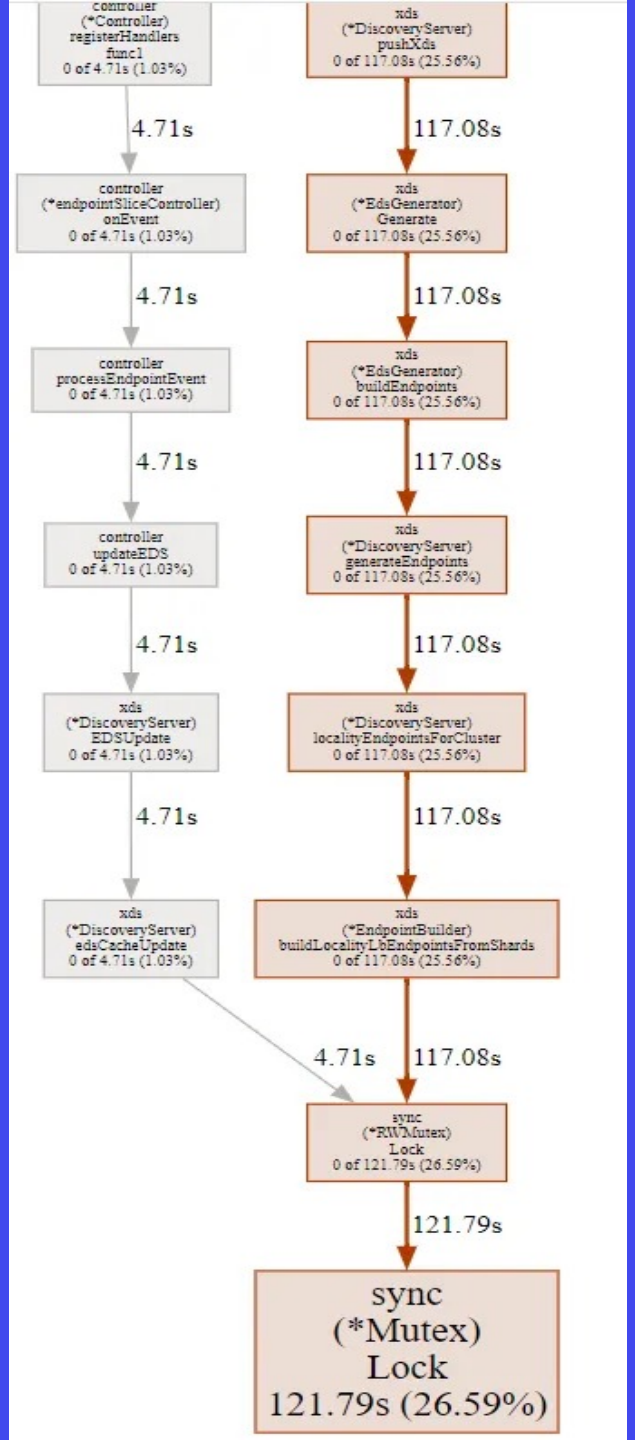
While dequeuing
we took a note of time taken to start processing,
time taken to execute the callback, type of the callback and logged it.

expedia group

Endpoints slices taking max time to get processed!

800ms to process a single Endpoint slice event during Pod churn was observed with additional logging we put.
If there are 500 Endpoint slices, it would take 6 mins before the last event is even dequeued. Here is our delay !

expedia group™



Fight for the

Profiling Istiod showed the same code blocks fighting for the lock, related to processing of an EndpointSlice.

The lock was shared between two processes, one on the side of dequeuing the controller Endpoint slice event, the other while pushing the proxy config!

expedia group

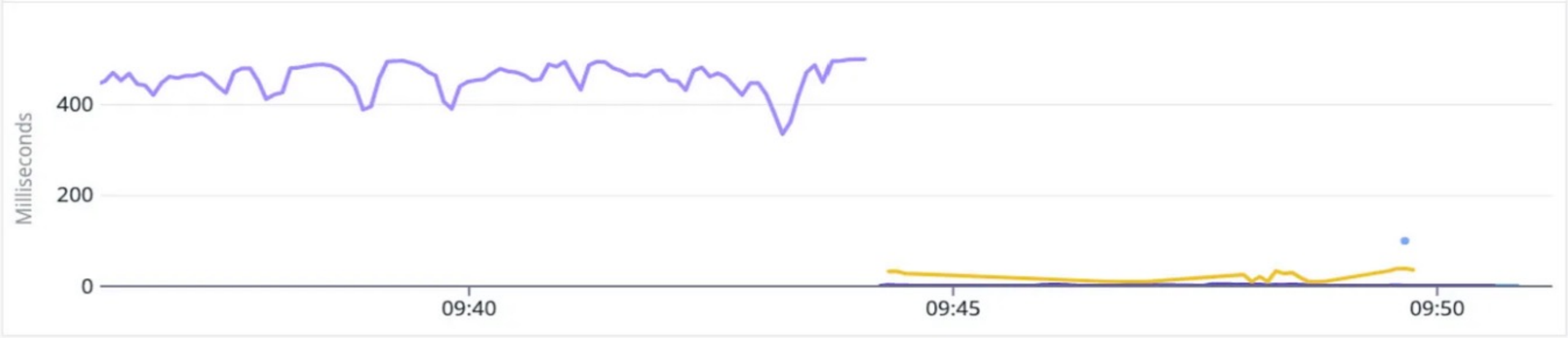
Fix from the maintainer

Thanks to the maintainers who swiftly fixed the issue by removing heavyweight processing outside of the locks, also turned one side of the lock to read lock.

Meanwhile, workaround time!

The fix was available in 1.19.0 which came out in September, but till that was released upstream, Expedia's Reliability engineering team found an interim workaround which solved majority of our problems. Let's see!

expedia group™



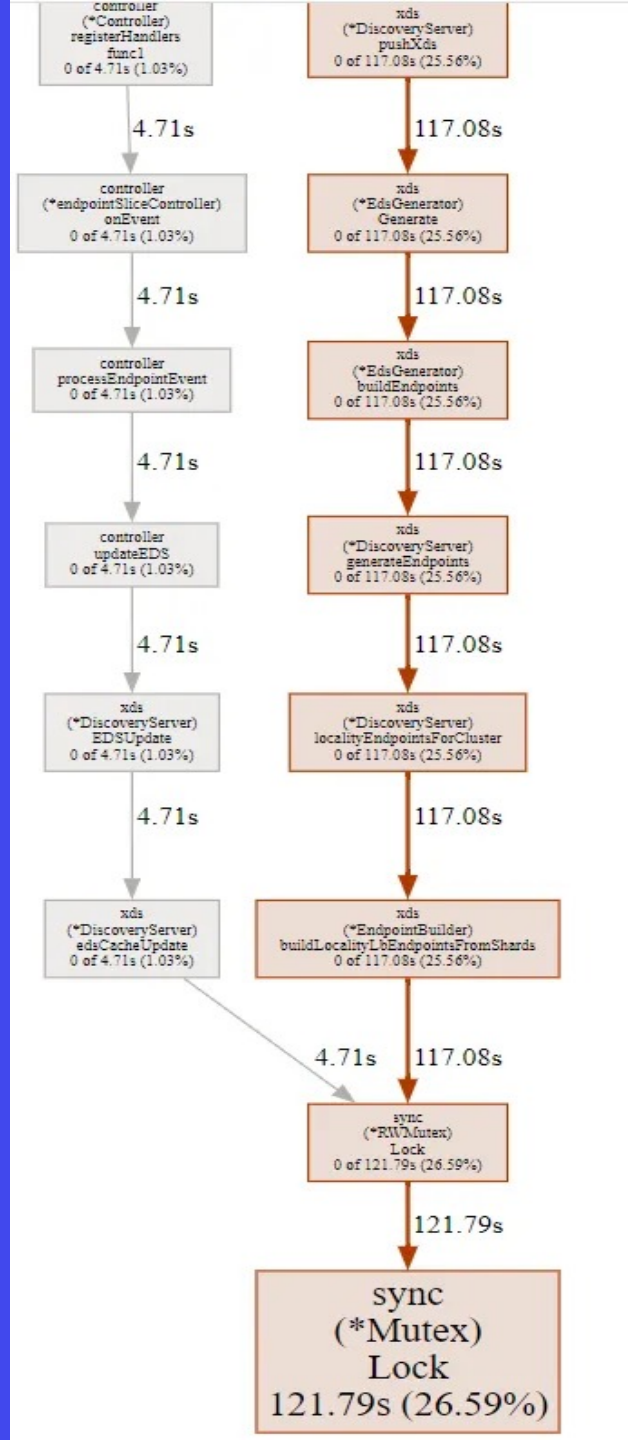
The processing time observed from one of our custom metrics dropped off the cliff!

Pilot push throttle to match vCPUs

This flag controlled the number of go-routines fighting for the lock which was contented. Reducing it from 100 to 16, worked for us.

expedia group™

Starved side ☹



More goroutines on this side, creating an imbalance!

Improved Telemetry

PR Link

- Expedia's Compute platform team added metrics to controller queue around latency and processing times.
- Can be enabled by setting env flag ISTIO_ENABLE_CONTROLLER_QUEUE_METRICS
- Available from v1.19.0

Huge Shoutout to Istio maintainers

John Howard, one of the members of Istio TOC swiftly addressed the root cause
in no time!
Kudos to them.

expedia group™

Take aways

Contribute back - beyond code!

Open-source tools should not be treated like a black box

expedia group

Thank you!

Medium blog

Istio GH Issue