Comparing eBPF and Istio/Envoy for Monitoring Microservice Interactions

Jonathan Perry

Roko Kruze

FLOWMILL

Join us for KubeCon + CloudNativeCon EU Virtual

Event dates: August 17-20, 2020
Schedule: Now available!
Cost: $75

Full Event Pass and **Complimentary Pass** are now available! Unsure what pass is the best for you? See the chart!

Register now!

Which pass is the best option *for me?*

	Full Event Pass	Complimentary Pass
All Keynote Sessions	✓	✓
All Breakout Sessions	✓	
All Lightning Talks	✓	
All Tutorials + 101 Track	✓	
Live Q+A with Speakers	✓	
Sponsor Showcase	✓	✓
Sponsor Demo Theater	✓	✓
Engage with Project Maintainers + Leads	✓	✓
Networking including Chat + Job Board	✓	
Experiences including Yoga, Meditation, Games, + Musical Performance	✓	
Ability to Register for Co-Located Events	✓	
50% off Certified Kubernetes Administrator or Application Developer Training + Exam Bundle	✓	

Hi! I'm Jonathan Perry

jperry@flowmill.com

www.flowmill.com

- Government: large-scale deployments
- MIT: extreme monitoring systems
  - prod at Facebook
- Flowmill: Founder

Hi! I’m Roko Kruze

rkruze@flowmill.com

www.flowmill.com

- Solutions Engineer
- Experience with large scale distributed systems

Two Approaches to Observability

Service Mesh (Istio + Envoy)

- Benefits
- Metrics available
- Considerations

eBPF (Berkeley Packet Filter)

- Approaches
- Metrics available

Benefits of a Service Mesh

Traffic Management

- Circuit Breakers
- Timeouts/Retries
- A/B testing

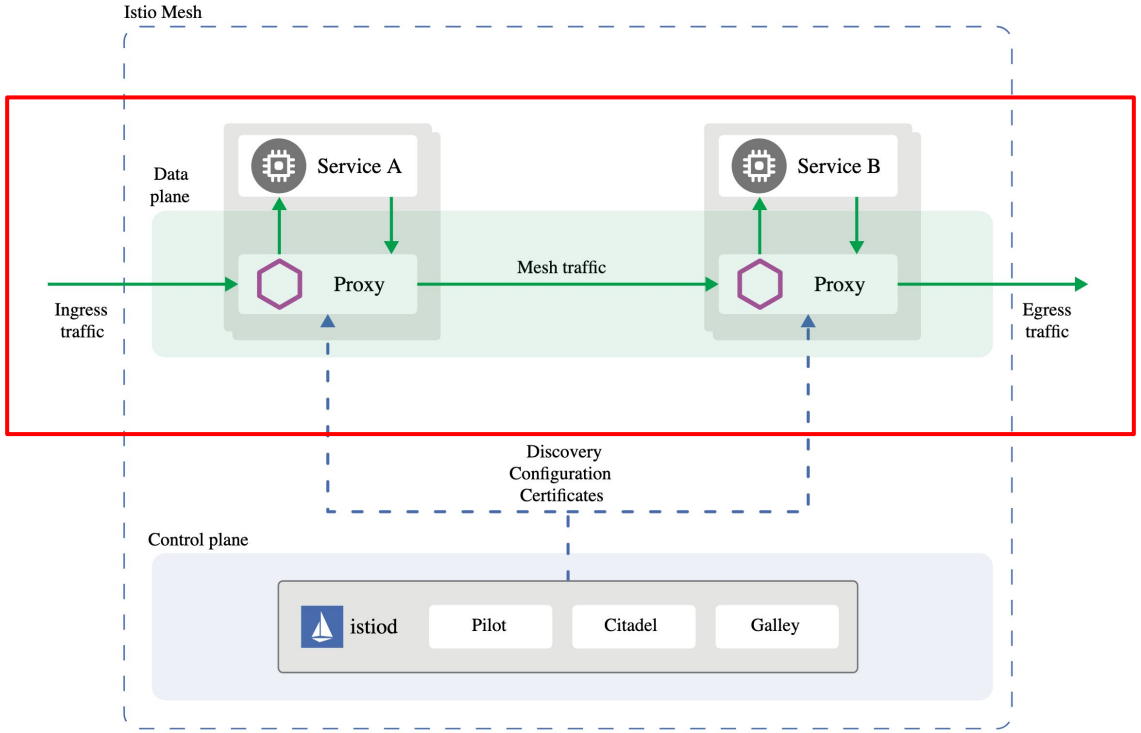
Security

- Auth[n,z]
- Encryption

Observability

- Tracing
- Monitoring
- Logging

Istio and Envoy Architecture



Istio Architecture

Metrics from Istio

HTTP, HTT2/2, GRPC:

Request Count	istio_request_total	COUNTER
Request Duration	istio_request_duration_milliseconds	DISTRIBUTION
Request Size	istio_request_bytes	DISTRIBUTION
Response Size	istio_response_bytes	DISTRIBUTION

For TCP traffic:

TCP Byte Sent	istio_tcp_sent_bytes_total	COUNTER
TCP Byte Received	istio_tcp_received_bytes_total	COUNTER
TCP Connections Opened	istio_tcp_connections_opened_total	COUNTER
TCP Connections Closed	istio_tcp_connections_closed_total	COUNTER

Network Layer vs Application Layer

Network layer metrics not available in a service mesh:

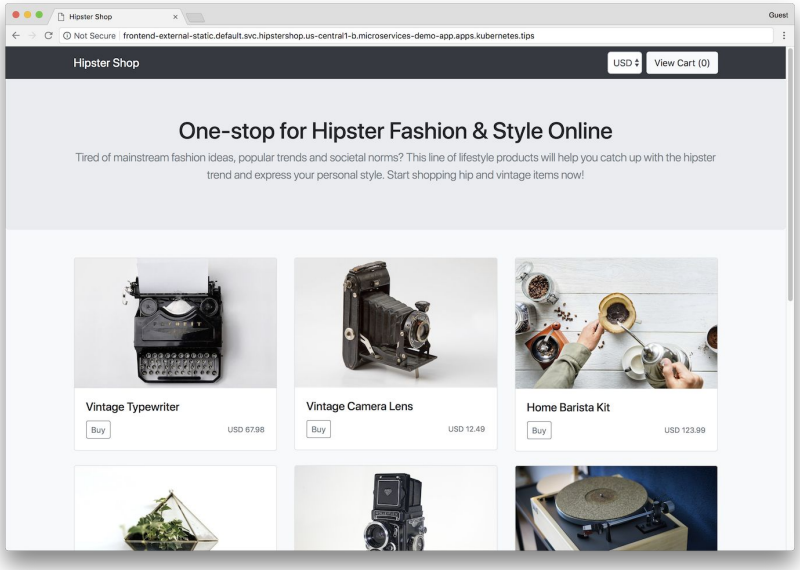
- Round trip time (RTT)
- Retransmissions / Packet loss
- UDP traffic
- DNS

- Istio and Envoy are primarily designed as an application layer service mesh

- You need another tool such as eBPF to get more detailed network data

Measuring Istio Overhead: Microservices-demo

GoogleCloudPlatform / **microservices-demo**



github.com/GoogleCloudPlatform/microservices-demo

- 3 node EKS cluster in AWS
- c5.xlarge instances
- Istio version 1.6.5 / default profile
- Locust simulating 500 users, running outside cluster

Benchmark Results

	Baseline application	Istio + Envoy	Percent Change
CPU Utilization (cluster)	13%	22%	+69%
P50 Response Time	16ms	25ms	+56%
P90 Response Time	33ms	48ms	+45%

eBPF

- Linux bpf() system call since 3.18
- Run code on kernel events
- Only changes, more data

- Safe: In-kernel verifier, read-only
- Fast: JIT-compiled

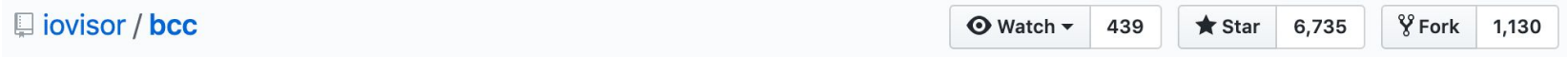
Unofficial BPF mascot by Deirdré Straughan

Metrics that can be provided by eBPF

From https://github.com/iovisor/bcc

- tools/tcpaccept.bt: Trace TCP passive connections (accept()). Examples.
- tools/tcpconnect.bt: Trace TCP active connections (connect()). Examples.
- tools/tcpdrop.bt: Trace kernel-based TCP packet drops with details. Examples.
- tools/tcplife.bt: Trace TCP session lifespans with connection details. Examples.
- tools/tcpretrans.bt: Trace TCP retransmits. Examples.
- tools/tcpsynbl.bt: Show TCP SYN backlog as a histogram. Examples.

Using eBPF



Demo:

to run a bcc container:

```
docker run -it --rm \
  --privileged \
  -v /lib/modules:/lib/modules:ro \
  -v /usr/src:/usr/src:ro \
  -v /etc/localtime:/etc/localtime:ro \
  --workdir /usr/share/bcc/tools \
  --pid=host \
  zlim/bcc
```

https://github.com/iovisor/bcc/blob/master/QUICKSTART.md
+ host pid namespace

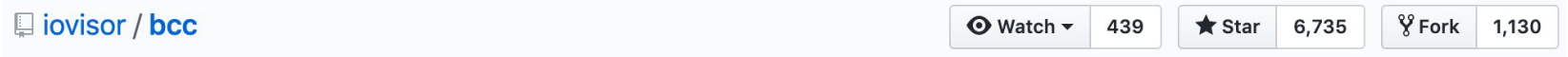
tcptop:

- instruments tcp_sendmsg and tcp_cleanup_rbuf

```
4782   aws-k8s-agen 127.0.0.1:50051       127.0.0.1:36184           0      0
20663  20663        127.0.0.1:36184       127.0.0.1:50051           0      0
4781   aws-k8s-agen 127.0.0.1:50051       127.0.0.1:36184           0      0

PID    COMM         LADDR6                          RADDR6                          RX_KB  TX_KB
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.12.162:52930          0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.12.162:52930          0      0
13132  metrics-serv ::ffff:172.31.44.33:4443        ::ffff:172.31.40.186:55870          0      1
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.12.162:52930          0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.12.162:52930          0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36946           0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36940           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36912           0      0
7612   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.12.162:52930          0      0
7612   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36918           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36878           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36896           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36964           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36890           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36958           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36872           0      0
13204  metrics-serv ::ffff:172.31.44.33:4443        ::ffff:172.31.40.186:55870          0      0
8055   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.12.162:52930          0      0
13205  metrics-serv ::ffff:172.31.44.33:4443        ::ffff:172.31.40.186:55870          0      0
13089  coredns      ::ffff:127.0.0.1:8080           ::ffff:127.0.0.1:52758              0      0
13086  metrics-serv ::ffff:172.31.44.33:4443        ::ffff:172.31.40.186:55870          0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36958           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36912           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36890           0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36964           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36872           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36940           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36896           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36878           0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36918           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36964           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36946           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36940           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36918           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36890           0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36878           0      0
7612   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36946           0      0
8055   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36896           0      0
8055   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36872           0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36958           0      0
7613   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36946           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36896           0      0
7612   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36940           0      0
7616   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36878           0      0
8872   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36918           0      0
7617   shippingserv ::ffff:172.31.42.75:50051       ::ffff:172.31.1.234:36872           0      0
```

Using eBPF



Demo:

to run a bcc container:

```
docker run -it --rm \
  --privileged \
  -v /lib/modules:/lib/modules:ro \
  -v /usr/src:/usr/src:ro \
  -v /etc/localtime:/etc/localtime:ro \
  --workdir /usr/share/bcc/tools \
  --pid=host \
  zlim/bcc
```

https://github.com/iovisor/bcc/blob/master/QUICKSTART.md
+ host pid namespace

tcptop:

- instruments tcp_sendmsg and tcp_cleanup_rbuf
- need to be careful of races:

```
# IPv4: build dict of all seen keys
ipv4_throughput = defaultdict(lambda: [0, 0])
for k, v in ipv4_send_bytes.items():
    key = get_ipv4_session_key(k)
    ipv4_throughput[key][0] = v.value
ipv4_send_bytes.clear()
```

as for loop is running, kernel continues with updates, clear() throws those out.

Getting application error codes

- eBPF supports user probes

```
$ go tool nm /root/hello | grep 'net/http\.'
  690a40 t net/http.Error
  64eee0 t net/http.Get
  6929e0 t net/http.HandleFunc
  6b6230 t net/http.Handler.ServeHTTP-fm
  6909e0 t net/http.HandlerFunc.ServeHTTP
  6805b0 t net/http.Header.Add
  680700 t net/http.Header.Del
  680690 t net/http.Header.Get
  680620 t net/http.Header.Set
  680750 t net/http.Header.Write
  681190 t net/http.Header.WriteSubset
  680840 t net/http.Header.clone
```

```
$ /funccount -p 31328 '/root/hello:net/http.*Header*'
Tracing 111 functions for
"/root/hello:net/http.*Header*"... Hit Ctrl-C to end.
^C
FUNC                                    COUNT
net/http.Header.Del                         3
net/http.Header.sortedKeyValues             3
net/http.Header.WriteSubset                 3
net/http.(*response).WriteHeader            3
net/http.extraHeader.Write                  3
net/http.(*chunkWriter).writeHeader         3
net/http.(*chunkWriter).writeHeader.func1       3
Detaching...
```

Benchmark Results

	Baseline application	tcptop from bcc	Percent Change
CPU Utilization (cluster)	13%	14%	+7%
P50 Response Time	16ms	16ms	0%
P90 Response Time	33ms	33ms	0%

Istio/Envoy Trade Offs

Strengths:

- Detailed Application Metrics
- Security
- Encryption
- Traffic Management

Weaknesses:

- Resource overhead
- Increased Latency
- Network Layer Metrics not available

eBPF Trade Offs

Strengths:

- Detailed Network Layer Metrics
- Can be optimized for minimal impact

Weaknesses:

- No full open source solution

Bringing it all together

Using a service mesh along with eBPF allows for deep observibility at both the application and network layer.

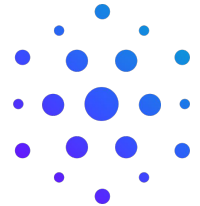
eBPF can help identify network issues that could affect the health of the service mesh.

Further questions on eBPF?

At Flowmill we are working with eBPF to bring network visibility to network applications.

<info@flowmill.com>

www.flowmill.com



FLOWMILL