Increasing Delivery Velocity with a Service Mesh at Indeed

Joshua Shanks
Senior Software Engineer, Indeed
Indeed is the #1 job site worldwide
We help people get jobs.
Software Engineer
Seattle, WA
60 countries
30 languages
200M unique visitors
20M jobs
About me
Agenda

- What is delivery velocity
- Our motivations
- Service mesh features
- How it helps
- Your options
Delivery Velocity
Delivery Velocity

- Automated Service Creation
Delivery Velocity

- Automated Service Creation
- Self Service VM provisioning
Delivery Velocity

- Automated Service Creation
- Self Service VM provisioning
- Self Service DB provisioning and migration
Delivery Velocity

- Automated Service Creation
- Self Service VM provisioning
- Self Service DB provisioning and migration
- Continuous Deployment & Integration
Where We Were
Where We Were

- Proprietary
Where We Were

- Proprietary
- Java
Where We Were

- Proprietary
- Java
- Data center Local
Where We Were

- Proprietary
- Java
- Data center Local
- Low Latency
Where We Were

- Proprietary
- Java
- Data center Local
- Low Latency
- 1 request = 1 connection
Where We Wanted To Be

- Open source
Where We Wanted To Be

- Open source
- Language agnostic
Where We Wanted To Be

- Open source
- Language agnostic
- gRPC, HTTP2, REST
Where We Wanted To Be

● Open source
● Language agnostic
● gRPC, HTTP2, REST
● Consul integration
Where We Wanted To Be

- Open source
- Language agnostic
- gRPC, HTTP2, REST
- Consul integration
- Cross data center
Where We Wanted To Be

- Open source
- Language agnostic
- gRPC, HTTP2, REST
- Consul integration
- Cross data center
Where We Are

Mesos

Host 1

A

B

Host 2

C

B

Host 3

D

C

Host 4

A

D

Local Registry
Where We Are

Mesos

Host 1
A
B

Host 2
C
B

Host 3
D
C

Host 4
A
D

Local Registry

Registration
Where We Are

Mesos

Host 1
A
B

Host 2
C
B

Host 3
D
C

Host 4
A
D

Local Registry

Connection
Where We Were

- Service to Client: Slot 22, Slot 71
- Service to Service: Slot 20, Slot 31
- Service to Service: Slot 72, Slot 82
Service Mesh

Linkerd + Consul
Big Picture

Host 1
- A
- Proxy
- Consul

Proxy
Agent
Mesos

Host 2
- Proxy
- Consul
- Agent
- B
- Proxy

Host 3
- Proxy
- Consul
- Agent
- B
- Proxy

Service Registration
C
C
Service Registration

ServiceA

Deploy Agent

ServiceB

Register A

(name, inbound port, healthcheck)

Register B

Consul
Big Picture

Host 1
A
B
Proxy
Consul
Agent

Host 2
Proxy
C
Proxy
B
Proxy

Host 3
Proxy
B
Proxy
C

Mesos
Service Discovery
Service Discovery

Host 1

Service A

Request
indeed-service=ServiceC

Host 2

Service C

Linkerd

Consul

Query C Pool

C Pool Results

Linkerd
Classifiers

- **Error**
  - HTTP 5XX
  - gRPC non-zero
- **Retryable**
  - HTTP GET, HEAD, OPTIONS, TRACE
  - gRPC UNAVAILABLE (14)
Load Balancing
Power of Two Choices: Least Loaded

Service B Pool

B1 R3
B2 R8
B3 R2
B4 R5
B5 R7
B6 R6

Pool Size == 6
Load Balancing
Power of Two Choices: Least Loaded

Service B Pool

- B1 R3
- B2 R8
- B3 R2
- B4 R5
- B5 R7
- B6 R6

Pool Size == 6
RNG(6) => 1
Load Balancing
Power of Two Choices: Least Loaded

Service B Pool

<table>
<thead>
<tr>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td>R8</td>
<td>R2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B4</th>
<th>B5</th>
<th>B6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5</td>
<td>R7</td>
<td>R6</td>
</tr>
</tbody>
</table>

Pool Size == 6
RNG(6) => 1
RNG(6) => 3
Load Balancing

Power of Two Choices: Least Loaded

Service B Pool

Pool Size == 6
RNG(6) => 1
RNG(6) => 3

B1 R3
B2 R8
B3 R2
B4 R5
B5 R7
B6 R6

B1 R3 > B3 R2
Circuit Breaking

Service B Pool

B1 F0  B2 F0  B3 F0
B4 F0  B5 F0  B6 F0
Circuit Breaking

Req 1

Service B Pool

Status Code 500

B1
B2
B3
B4
B5
B6

F1
F0
F0
F0
F0
F0
Circuit Breaking

Req 2 → Status Code 500 → B1
Req 3 → Status Code 500 → B1
Req 4 → Status Code 500 → B1

Service B Pool:
- B1 F4
- B2 F0
- B3 F0
- B4 F0
- B5 F0
- B6 F0
Circuit Breaking

Req 5 ➔ Status Code 500 ➔ B1

Service B Pool

- B1 (F5)
- B2 (F0)
- B3 (F0)
- B4 (F0)
- B5 (F0)
- B6 (F0)
Retries
1st Try

Request 1 → HTTP GET → Linkerd → HTTP 500 → B2
Delivery Velocity
Where we are now

- Open Source
  - okhttp h2c patch from Jaye Pitzeruse
- Language agnostic
  - Java & Python
- HTTP2
  - 95% 2ms added latency
- Consul Integration
- Cross Data center
Client Implementation
Client Implementation

1. Retrieve outbound port
Client Implementation

1. Retrieve outbound port

```java
port = Env.get("egressPort");
```
Client Implementation

1. Retrieve outbound port

```java
port = Env.get("egressPort");
cl = Client("http://localhost:" + port);
```
1. Retrieve outbound port
2. Inject service header

```java
port = Env.get("egressPort");
cl = Client("http://localhost:" + port);
```
Client Implementation

1. Retrieve outbound port
2. Inject service header

```java
    port = Env.get("egressPort");
    cl = Client("http://localhost:" + port);
    req = cl.makeFooRequest();
    req.setHeader("indeed-service", "ServiceB");
```
Service Implementation
Service Implementation
Future Plans
Future Plans

- Transparent TLS
Future Plans

- Transparent TLS
- Authentication
- Authorization
Future Plans

- Transparent TLS
- Authentication
- Authorization
- Rate Limiting
Future Plans

- Transparent TLS
- Authentication
- Authorization
- Rate Limiting
- Tracing
Future Plans

- Transparent TLS
- Authentication
- Authorization
- Rate Limiting
- Tracing
- Metrics
- Chaosify
Future Plans

- Transparent TLS
- Authentication
- Authorization
- Rate Limiting
- Tracing
- Metrics
- Chaosify
Linkerd

- Finagle
- Cloud Native Computing Foundation
- Scala
- HTTP
- Plugin Support
Conduit

- Kubernetes
- Alpha
- Rust & golang
- TCP
- Linkerd 2
Envoy

- Lyft
- CNCP
- C++
- TCP
- Extensions
Istio

- IBM & Google
- Envoy underneath
- golang
- TCP
- Security
Consul Connect

- HashiCorp
- Beta
- golang
- TCP
- ACLs
Thanks for coming

jshanks@indeed.com