HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Requirements and Impact Analysis of Applying a Service Mesh to a Distributed off-the Shelf Software System

Kolloquium zur Masterarbeit

Marc Herschel, 30.01.2024

Inhaltsverzeichnis

Bilder ohne Quellenangabe sind selber illustriert und aus der Arbeit entnommen!

Problem- und Zielstellung der Arbeit

- **HAFAS** ist ein komplexes verteiltes Softwaresystem, welches in der Domäne der Verkehrsauskunftssysteme zum Einsatz kommt.
- Evolution der Betriebsumgebung: *On-premise* -> *AWS EC2 (momentan)* -> *AWS EKS*

Gedankengang: Mit Kubernetes als Basis; welche Technologien können für die Zukunft relevant sein -> ***Service Mesh***

Problem: Was ist der praktische Nutzen der Einführung eines Service Mesh in ein bereits bestehendes verteiltes Softwaresystem?

Ziel: Durchführung einer Fallstudie zur Einführung eines Service Mesh in einer HAFAS-Umgebung um die Machbarkeit / Praktikabilität zu überprüfen.

HaCon Fahrplan-Auskunfts-System (HAFAS)

- Teil der Hacon MaaS (*Mobility as a Service*) Platform
- Verkehrsauskunftssystem mit Kunden wie *DB*, *SBB*, *BART*, …
- Verteiltes, modulares *“off-the Shelf Software System”* dem Baukastensystem je nach Kundenanforderungen anpassbar
- Service-basierte Architektur, aber keine Microservice-Architektur

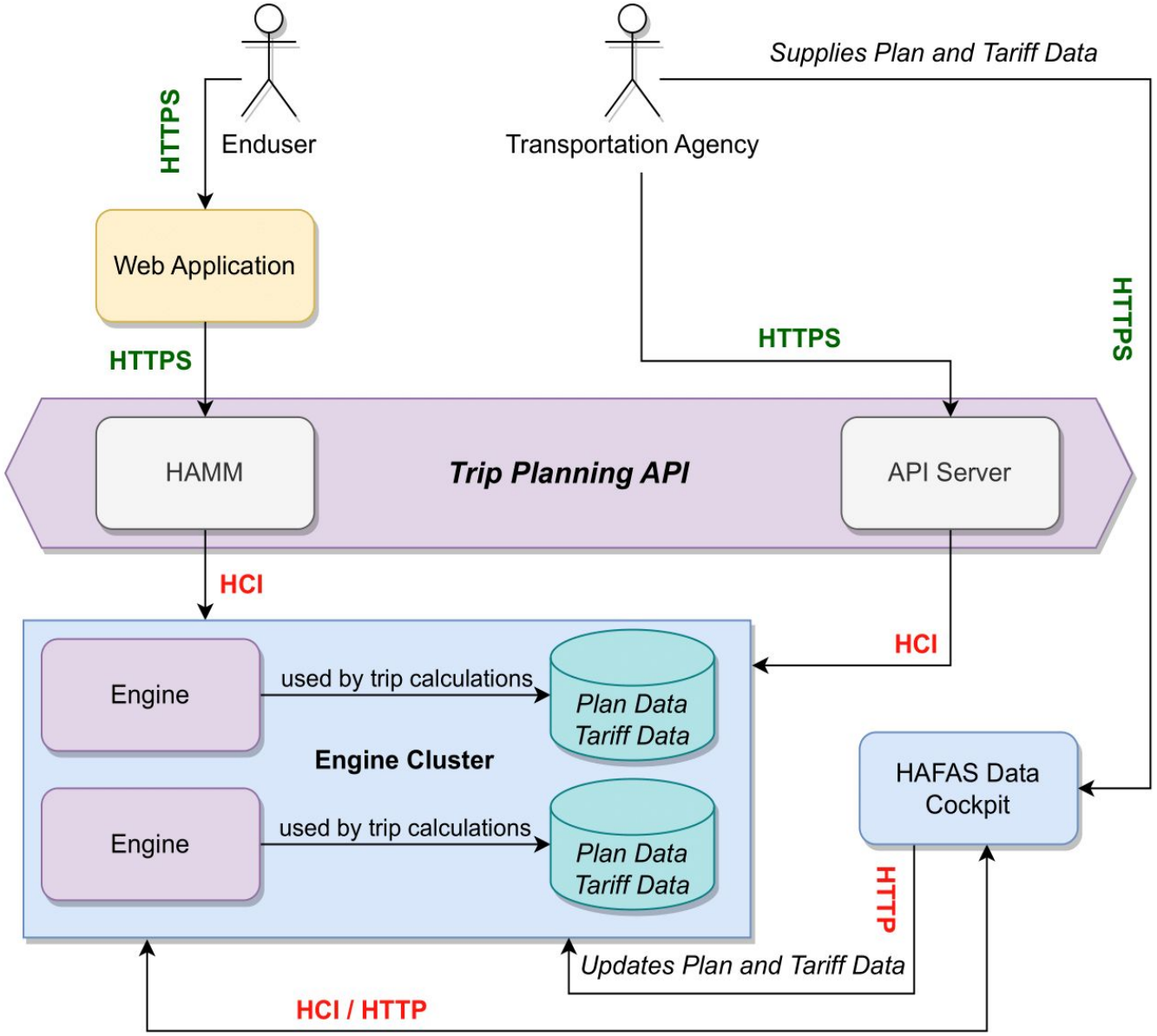
Quelle: hacon.de - SBB Reiseplaner-App

HAFAS wird derzeit hauptsächlich über AWS EC2-Instanzen für Kunden betrieben.

*Eine vollständige Migration aller Komponenten in containerisierte Umgebungen findet momentan statt -> **Kubernetes***

Quelle: hacon.de

HaCon Fahrplan-Auskunfts-System (HAFAS)

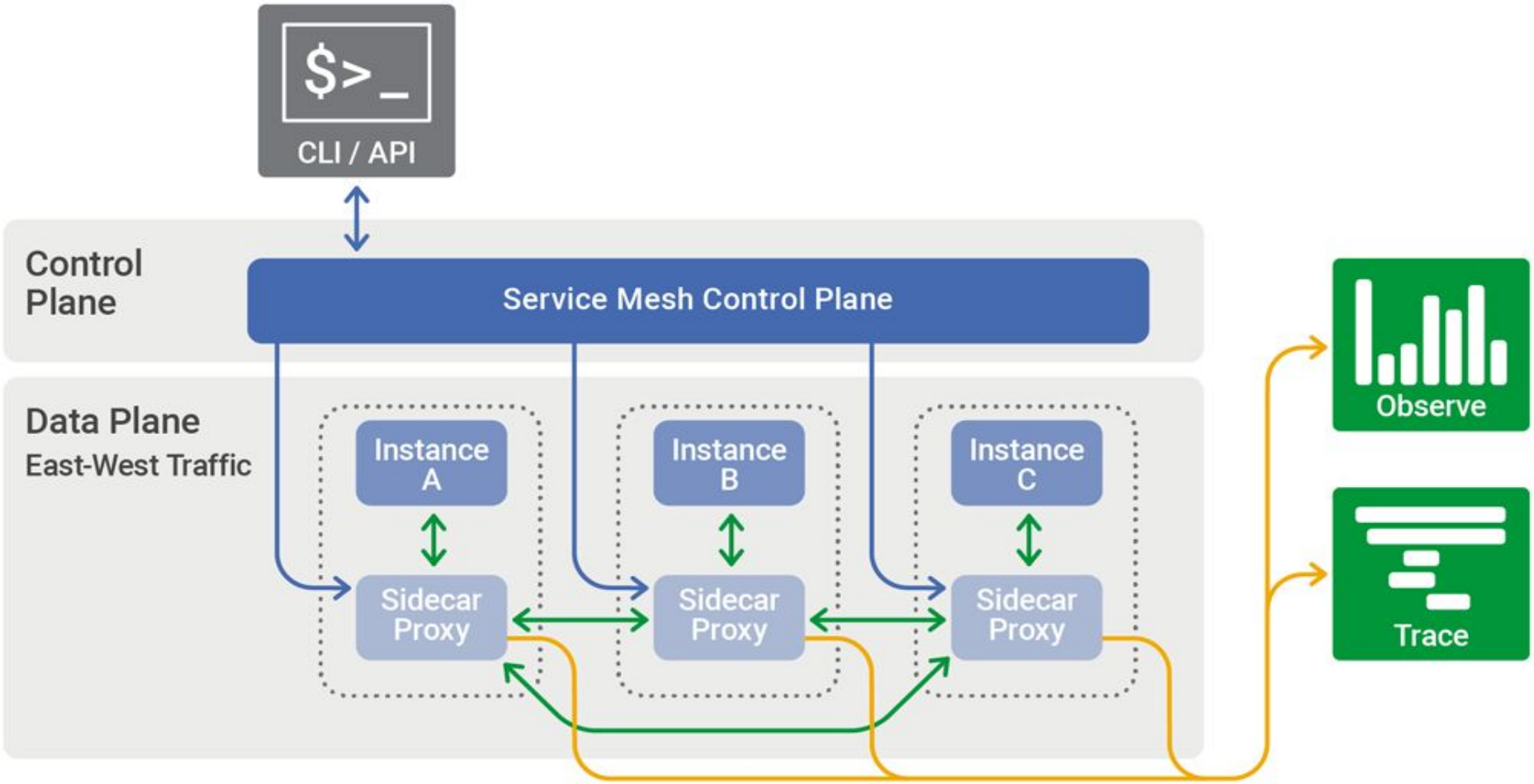


***Abgebildet**: Vereinfachtes Teilsystem welches im Verlauf der Arbeit betrachtet wird.*

HAFAS selber ist modular und weitaus komplexer mit Echtzeitdaten-Verwaltung, Business Intelligence Features, Ticketing, externen Verkehrs-Dienstleistern, etc…

Service Mesh

... ist eine dedizierte Infrastrukturebene zur Steuerung und Überwachung der Service-zu-Service-Kommunikation in einem verteilten System.



Quelle: https://www.nginx.com/blog/what-is-a-service-mesh/.

Service Mesh - Features

Netzwerkverkehr-Management

- Traffic Redirection / Splitting z. B. für Blue / Green / Canary-Deployments
- Circuit Breaking / Retry Logic zum Verbessern der Resilienz
- Fault Injection / Traffic Mirroring für Softwaretests
- … viele weitere Features

Sicherheit

- mTLS (mutual TLS)-Verschlüsselung zwischen Kommunikation von Services
- Automatisierung von Zertifikatsverwaltung
- Zugriffsverwaltung bei Service-zu-Service Kommunikation

Observerabilität

- Sammeln Metriken bzgl. Service Status / Traffic / Health etc.
- Distributed Tracing zum besseren nachvollziehen von Netzwerkverkehr innerhalb der Service-zu-Service Kommunikation

Service Mesh

Challenges:

Performance, Ressource Overhead, Verfügbarkeit, Adaptabilität, Komplexität

Opportunities:

Abstraktion / Standardisierung von Features, IaC-Konfigurationsverwaltung,

Verbesserung der Testbarkeit / Anwendungssicherheit

Quelle: Logos von respektiven Websites der Projekte

Service Mesh bei Hacon

Keine klassische Microservice-Architektur, bei der einzelne Dienste fachlich und feingranular getrennt voneinander entwickelt werden.

❖ eher als Service-basierte Architektur zu beschreiben

Grundsätzlich interessant aus folgenden Gründen:

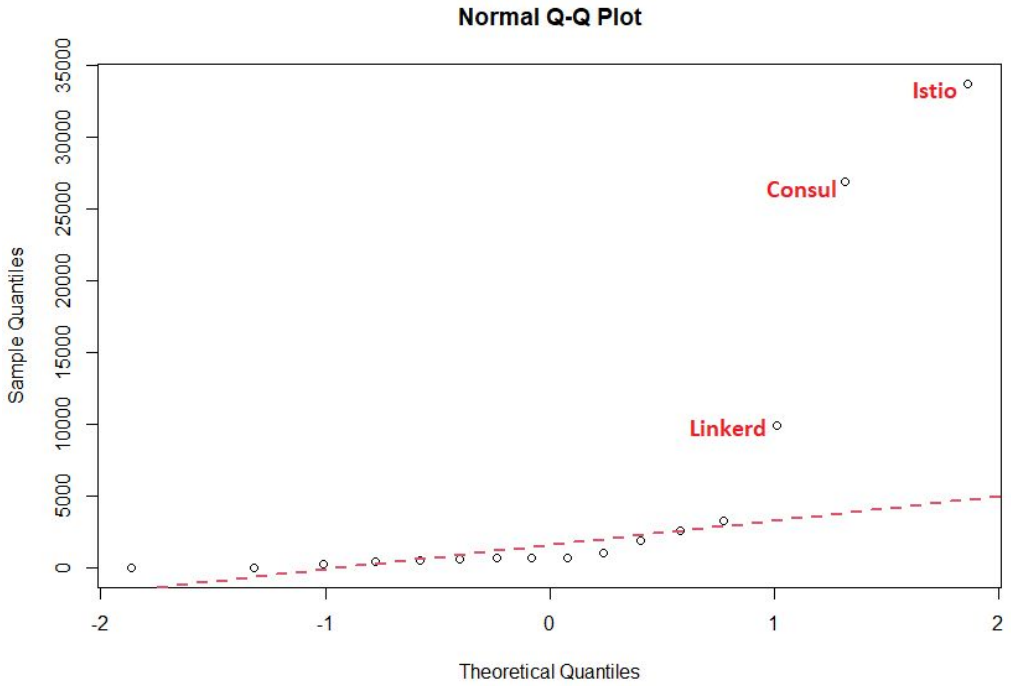
- Sicherheitsaspekte bezüglich mTLS
- Service-zu-Service Layer 7 Autorisierung bei Multi-Tenant-Kubernetes Clustern
- Nützlichkeit einiger der Netzwerk-Aspekte bzgl. Testing + Deployment
- Ergänzende Monitoring Eigenschaften evtl. nützlich in der Produktion

Anforderungen an Service Mesh-Projekte

Id	Szenario / Funktionale Anforderungen
S1 (FR1)	Data-in-Transit Verschlüsselung mit mTLS
S2 (FR2)	Zertifikatsverwaltung für Daten- und Kontrollebene
S3 (FR3)	Layer 7 Service-zu-Service Zugriffskontrolle
S4 (FR4)	Circuit Breaking zur Verbesserung der Resilienz
S5 (FR5)	Fault Injection zum vereinfachten Testen der Robustheit

Nichtfunktionale Anforderungen	
Technische Reife	Benutzbarkeit
Kompatibilität	Portabilität
Wartbarkeit	Nutzungskosten
Konfigurierbarkeit	Lizenzierbarkeit

Auswahlverfahren von Projekten für die Fallstudie



NFR	1	2	3	4	5	6	7	8
vs	$L > I$	$L = I$	$L < I$	$L = I$	$L > I$	$L = I$	$L = I$	$L = I$

Table 14: Comparison of Linkerd vs. Istio responses to non-functional requirements

Linkerd > Istio: Technische Reife
Linkerd > Istio: Benutzbarkeit
Istio > Linkerd: Wartbarkeit

FR	Linkerd	Istio	Consul	App Mesh
FR1	Ja	Ja	Ja	Ja
FR2	Ja	Ja	Ja	Ja
FR3	Ja	Ja	Ja	**Nein**
FR4	Ja	Ja	Ja	Ja
FR5	Ja	Ja	**Nein**	**Nein**

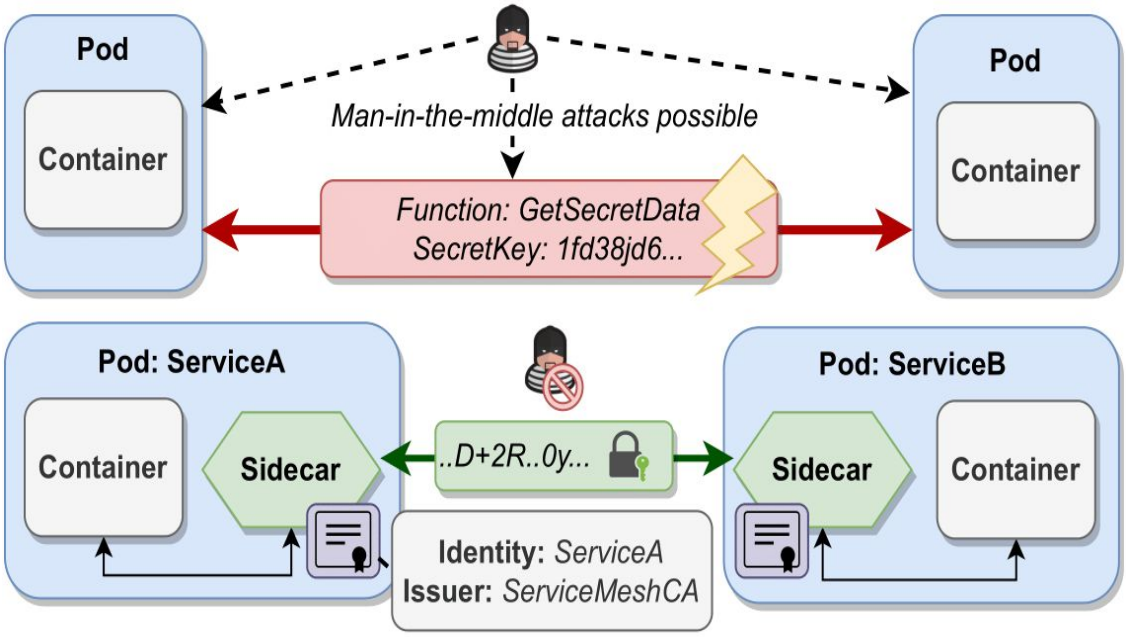
Entscheidung gefallen für Linkerd!

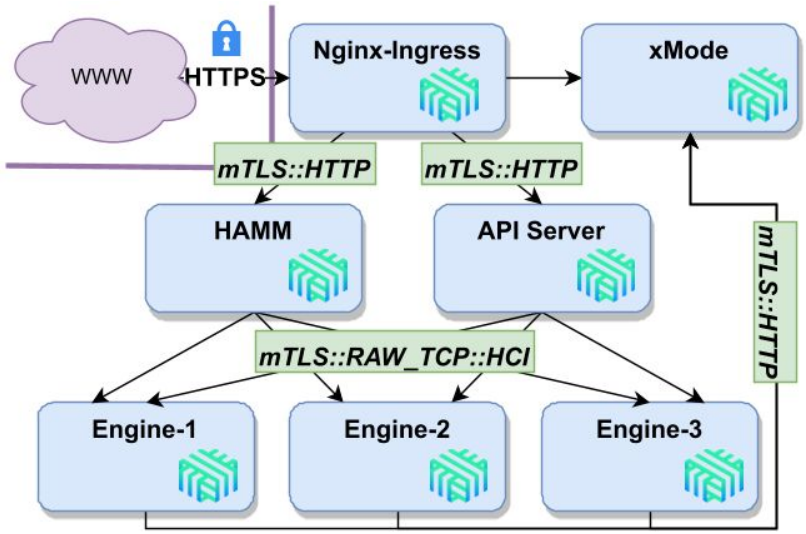
- Bessere Performance und Verwaltung von Ressourcen
- Mehr Audits durchgeführt
- Durch Rust-Sidecar ganze Klasse an Speicherverwaltungs-CVEs nichtig

Untersuchung von Anwendungsfällen für HAFAS

Data-in-Transit Verschlüsselung mit mTLS

- Verschlüsselung von HTTP und Raw-TCP Verkehr mit beidseitiger Authentisie der Services mittels ausgestellten Zertifikaten
- Für Server-Speak-First-Protokolle ist eine Sonderkonfiguration notwendig
- Jeder Dienst weist sich mit Zertifikat aus → *Identities* (Identitätsverfahren)





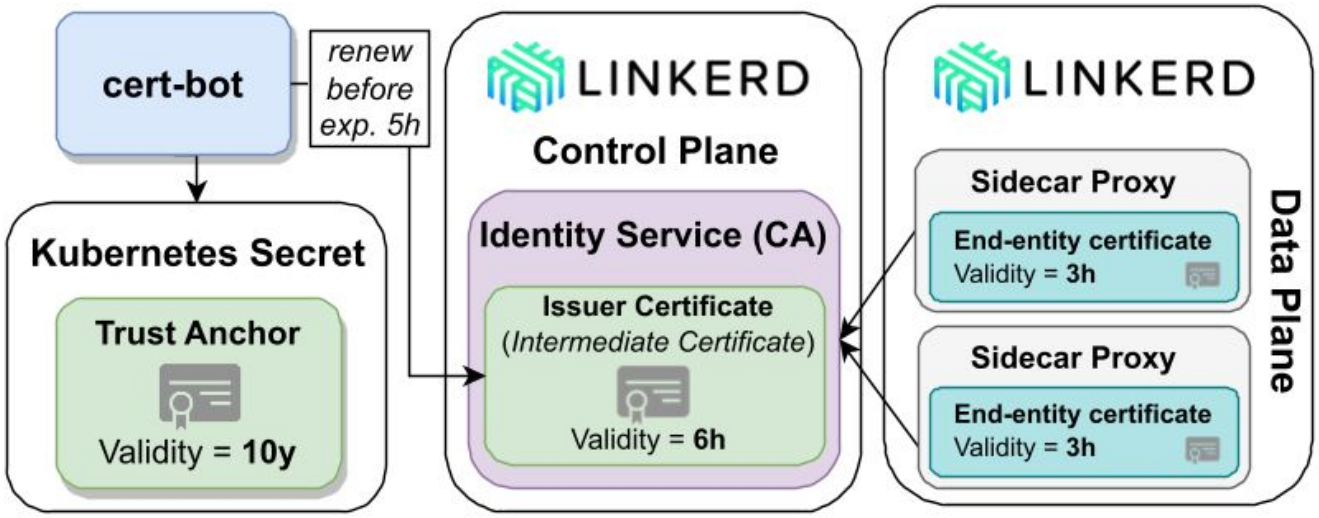
Untersuchung von Anwendungsfällen für HAFAS

Zertifikatsverwaltung für Daten- und Kontrollebene

Zertifikat-Hierarchie: Trust Anchor > Issuer Zertifikat (CA) > End-entity Zertifikat (Sidecar)

Trust Anchor und CA-Zertifikat standardmäßig 1 Jahr gültig ohne automatische Erneuerung
→ Betriebsrisiko bei Ablauf der Zertifikate; daher automatische Erneuerung (z.B. cert-bot)

Kurzlebige Zertifikate für CA und Sidecar Proxies empfehlenswert falls ein Zertifikat oder Schlüssel kompromittiert wird.



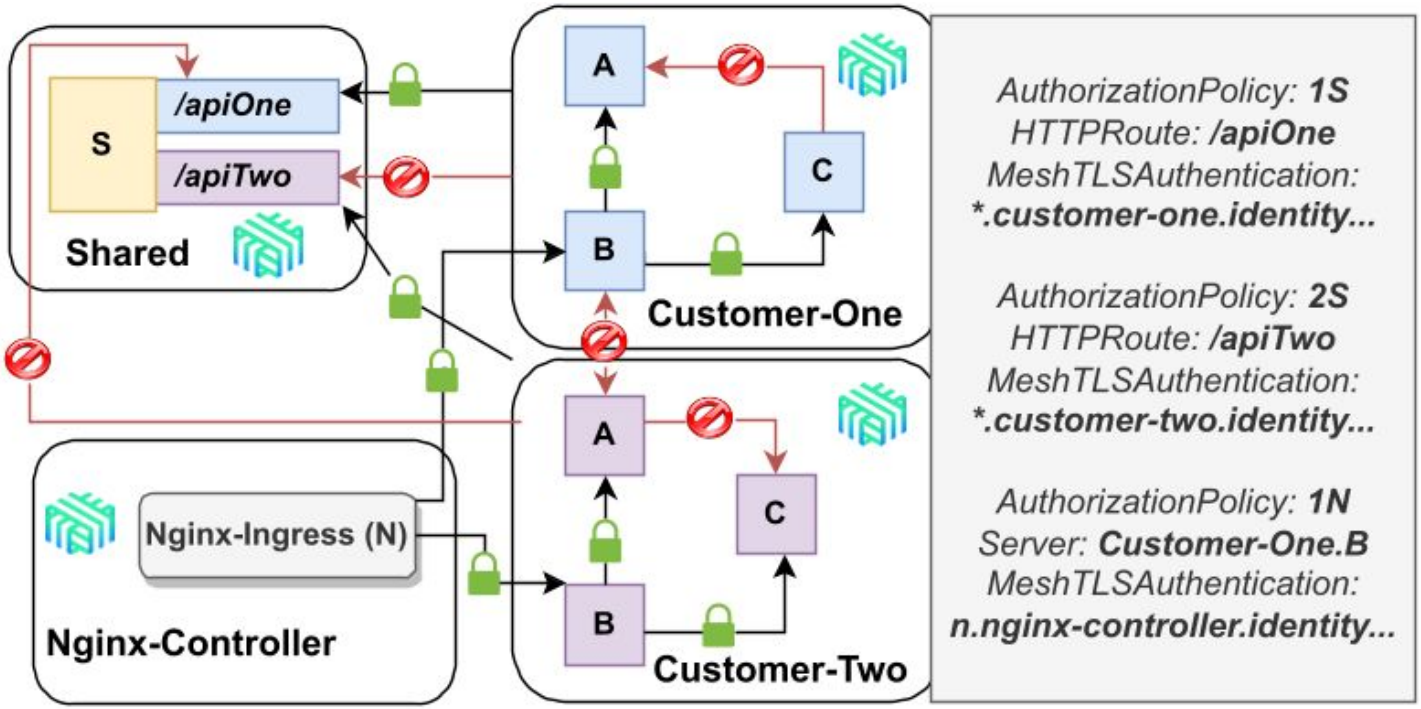
Trust Anchor muss händisch erneuert werden, hier ist eine lange Laufzeit empfehlenswert.

Austausch kann ohne Downtime durchgeführt werden.

Untersuchung von Anwendungsfällen für HAFAS

Service-zu-Service Zugriffskontrolle

- Zusätlicher Mechanismus zur feingranularen Zugriffskontrolle auf OSI-Layer 7 (k8s bietet z. B. nur Zugriffskontrollen auf Layer 4 an) aufbauend auf *Identities*
- Einschränkung auf Server und HTTP-Routen möglich (mittels *Identities* und CIDR)



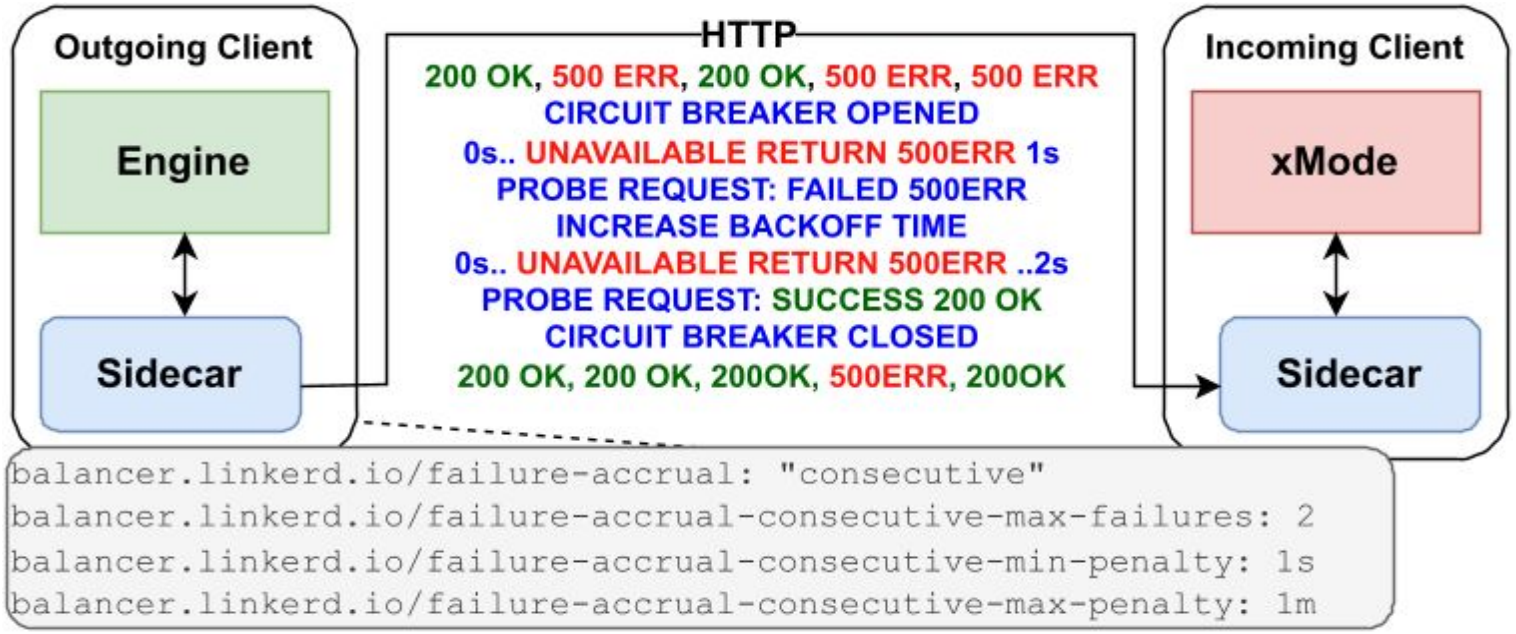
Konfiguration mittels k8s-CRDs.

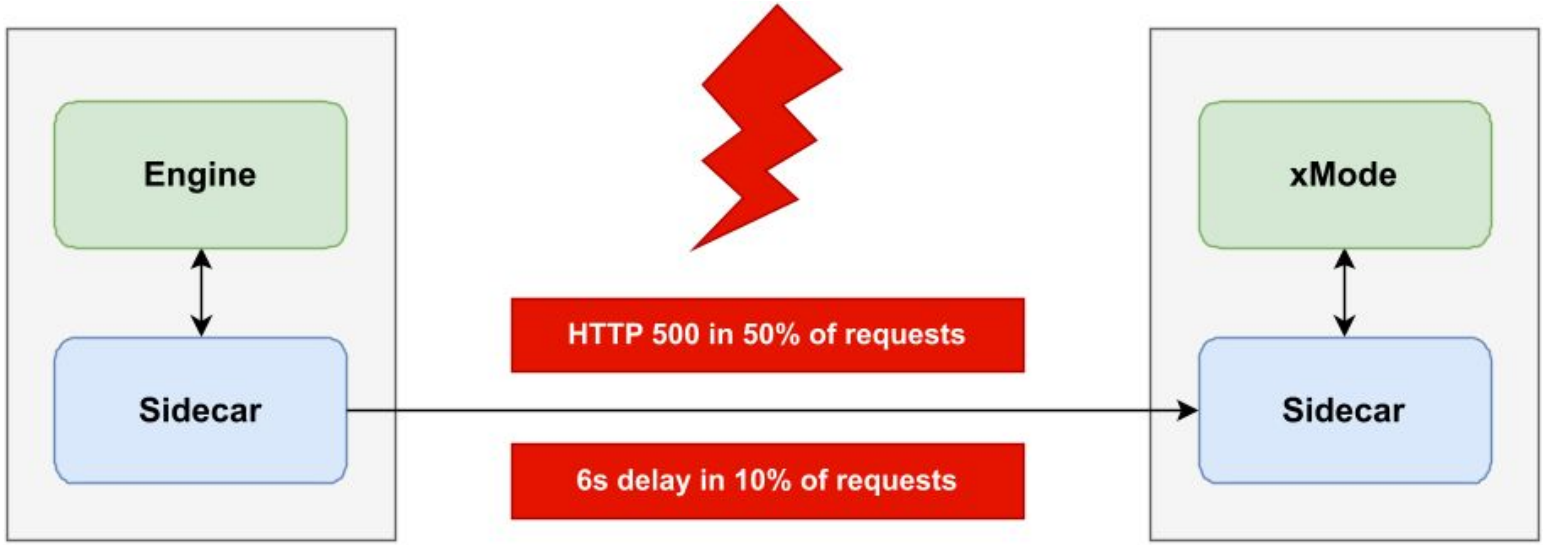
Use case: Isolation von Multi-Tenant-Umgebungen und teilen von Diensten.

Implementation von Zero-Trust Policies.

Circuit Breaker

Verbessert Resilienz durch das entfernen von fehlerhaften Diensten aus der Liste des Load Balancers bis sich eine Erholung ergeben hat.



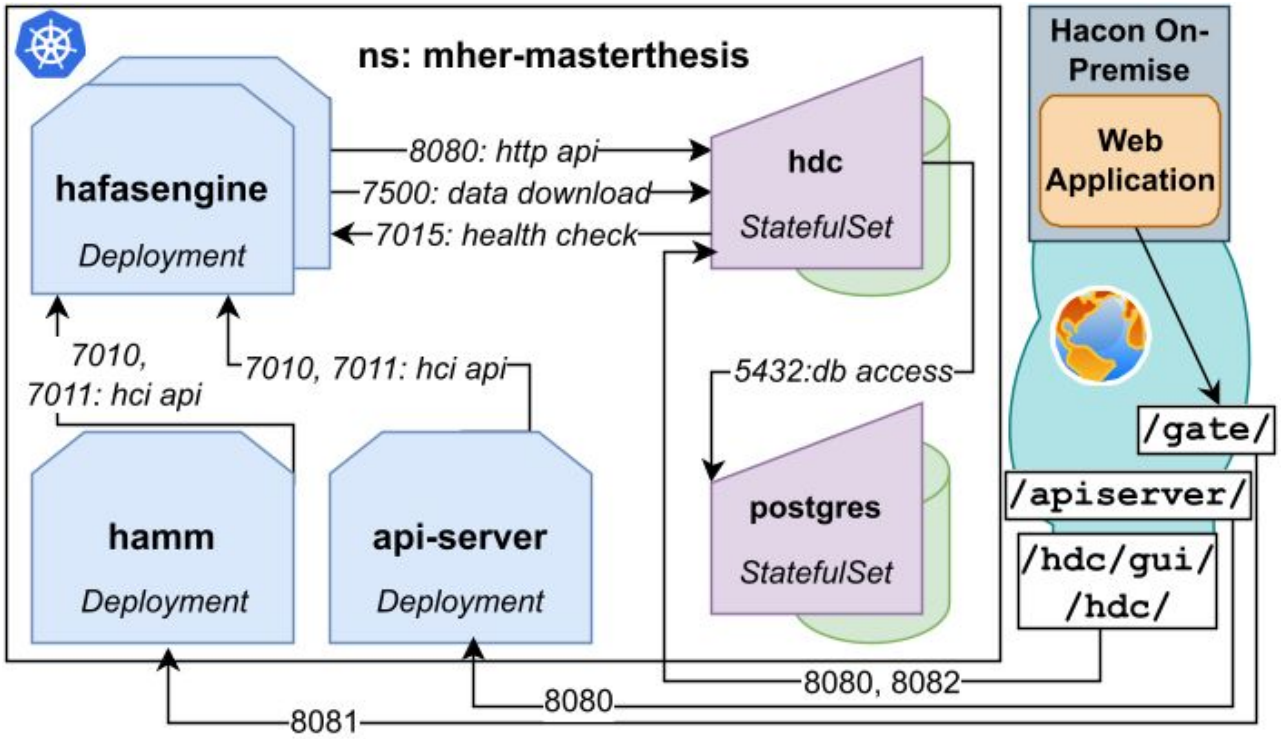


Fault Injection

Chaos Engineering Technik um Robustheit eines Systems zu testen. Erlaubt es fehlerhafte Requests in HTTP-Routen einzuschleusen.

Durchführung der Fallstudie

- Aufsetzen einer Testumgebung in einem Kubernetes Dev-Cluster
- Bereitstellung von Nodes nur für den Namespace für Lasttests
- Praktische Anwendung von Szenario S1 und S2 (mTLS + CA-Management)
- Deployment und Konfiguration von Linkerd und Linkerd-Viz
- Validierung von mTLS-Verschlüsselung und Durchführung von Lasttests



Lasttest-Szenario 1:

Location Search by Name @ 20RPS über fünf Minuten

Kurzlebig, wenig CPU-Nutzung

Lasttest-Szenario 2:

Trip Search @ 10 RPS über fünf Minuten

Langlebig, mehr CPU-Nutzung

Validierung von mTLS-Verschlüsselung

1. *Trust Anchor & CA-Zertifikat*
2. *Sidecar-Zertifikat*
3. *Package Dump analysiert in Wireshark*

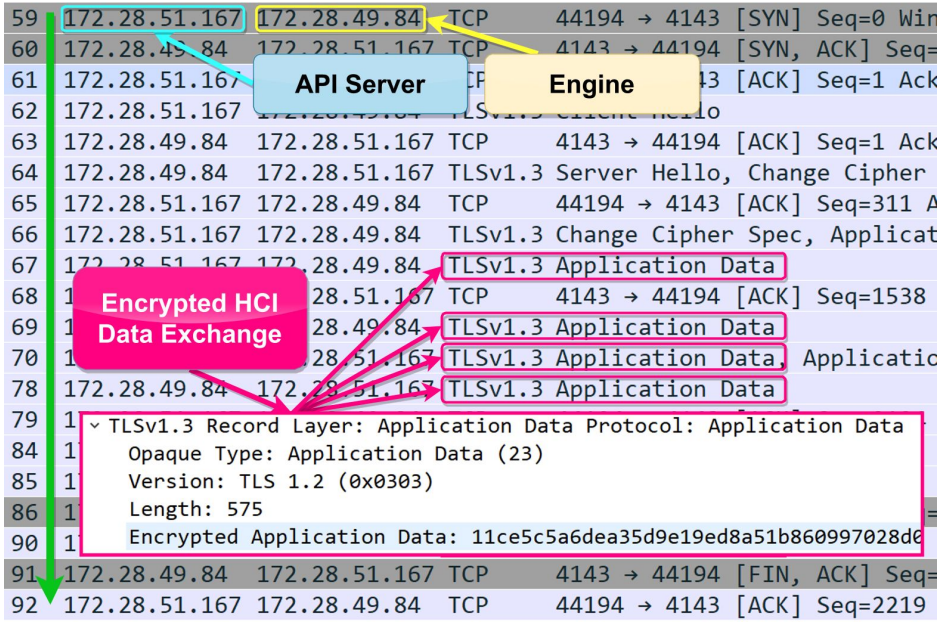
Long Living Trust Anchor (10y)

Property	Value
Issuer	CN = root.linkerd.cluster.local
Subject	CN = root.linkerd.cluster.local
Valid From	23 Nov 2023, 11:12 p.m.
Valid To	20 Nov 2033, 11:12 p.m.
Serial Number	FE:B4:71:37:ED:3F:7F:7B:B4
CA Cert	Yes
Key Size	256 bits
Fingerprint (SHA-1)	03:EB:D2:0D:83:C6:E5:A8:04
Fingerprint (MD5)	05:7A:AB:61:EC:37:42:40:BC

Short Living Intermediate Certificate (6h)

Property	Value
Issuer	CN = root.linkerd.cluster.local
Subject	CN = identity.linkerd.cluster.local
Valid From	25 Nov 2023, 5:03 a.m.
Valid To	25 Nov 2023, 11:03 a.m.
Serial Number	FB:30:A4:AA:9C:21:8C:2A:9B:0
CA Cert	Yes
Key Size	256 bits
Fingerprint (SHA-1)	6A:4C:F3:53:D1:DA:3F:7F:3A:7
Fingerprint (MD5)	52:B9:37:11:9F:5B:EF:FB:65:AF

```
Certificate: POD hdc-0 (1 of 1)
Data:
    Version: 3 (0x2)
    Serial Number: 2 (0x2)
Signature Algorithm: ECDSA-SHA256
    Issuer: CN=identity.linkerd.cluster.local
    Validity
        Not Before:  Nov 25 04:19:29 2023 UTC
        Not After :  Nov 25 07:20:09 2023 UTC
    Subject: CN=default.mher-masterthesis.serviceaccount.
       identity.linkerd.cluster.local
    Subject Public Key Info:
        Public Key Algorithm: ECDSA
            Public-Key: (256 bit)
            X:
```



Ergebnisse der Fallstudie

Auswirkungen auf Latenzzeiten

- Konstanter Faktor von *3-4 ms* Overhead bzgl. Median-Latenz
- Starke Ausreißer im 99. Perzentil bei Linkerd zu sehen (? jMeter)

→ *Der konstante Latenz-Overhead ist damit akzeptabel!*

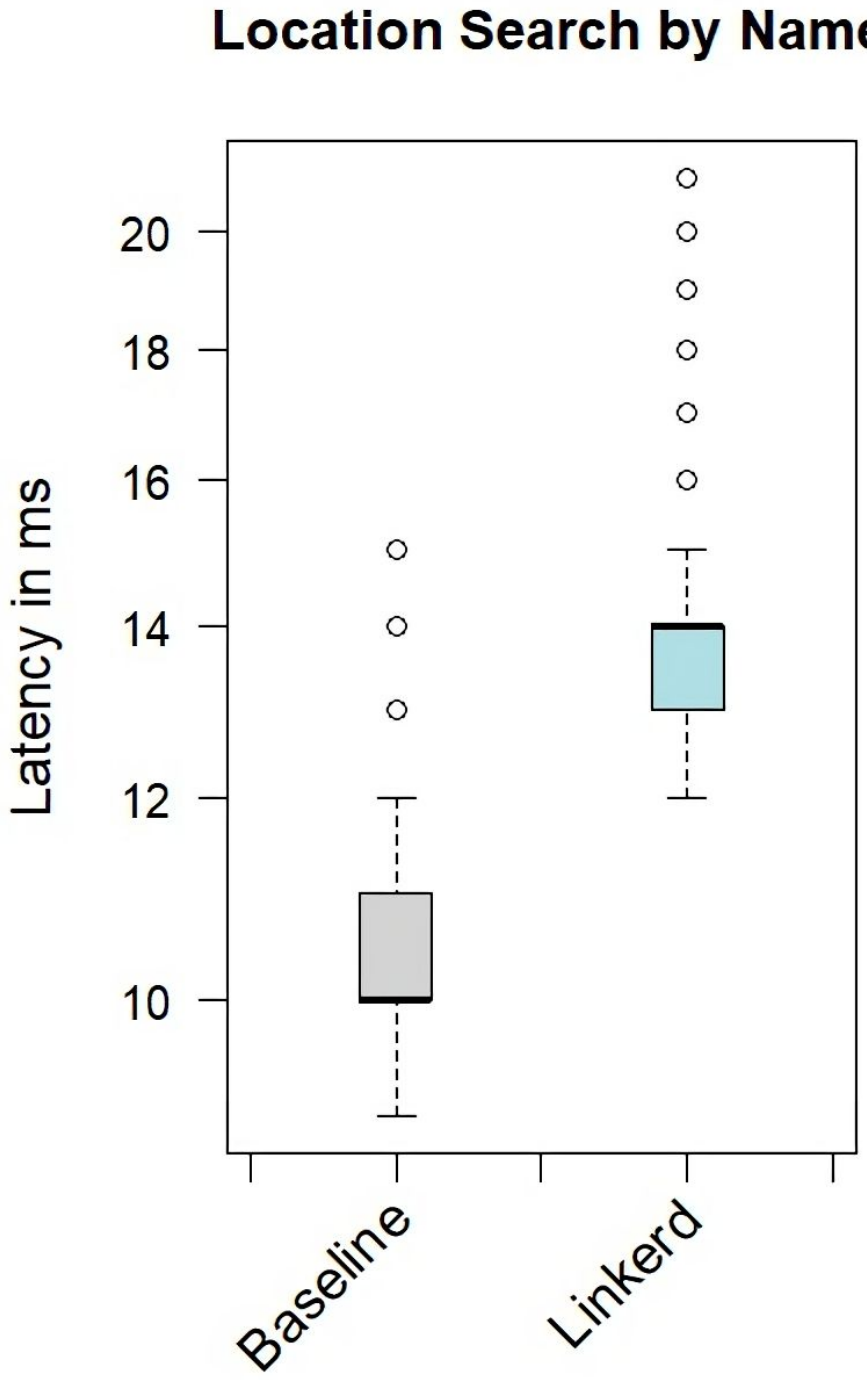
Auswirkungen auf Ressourcen-Overhead

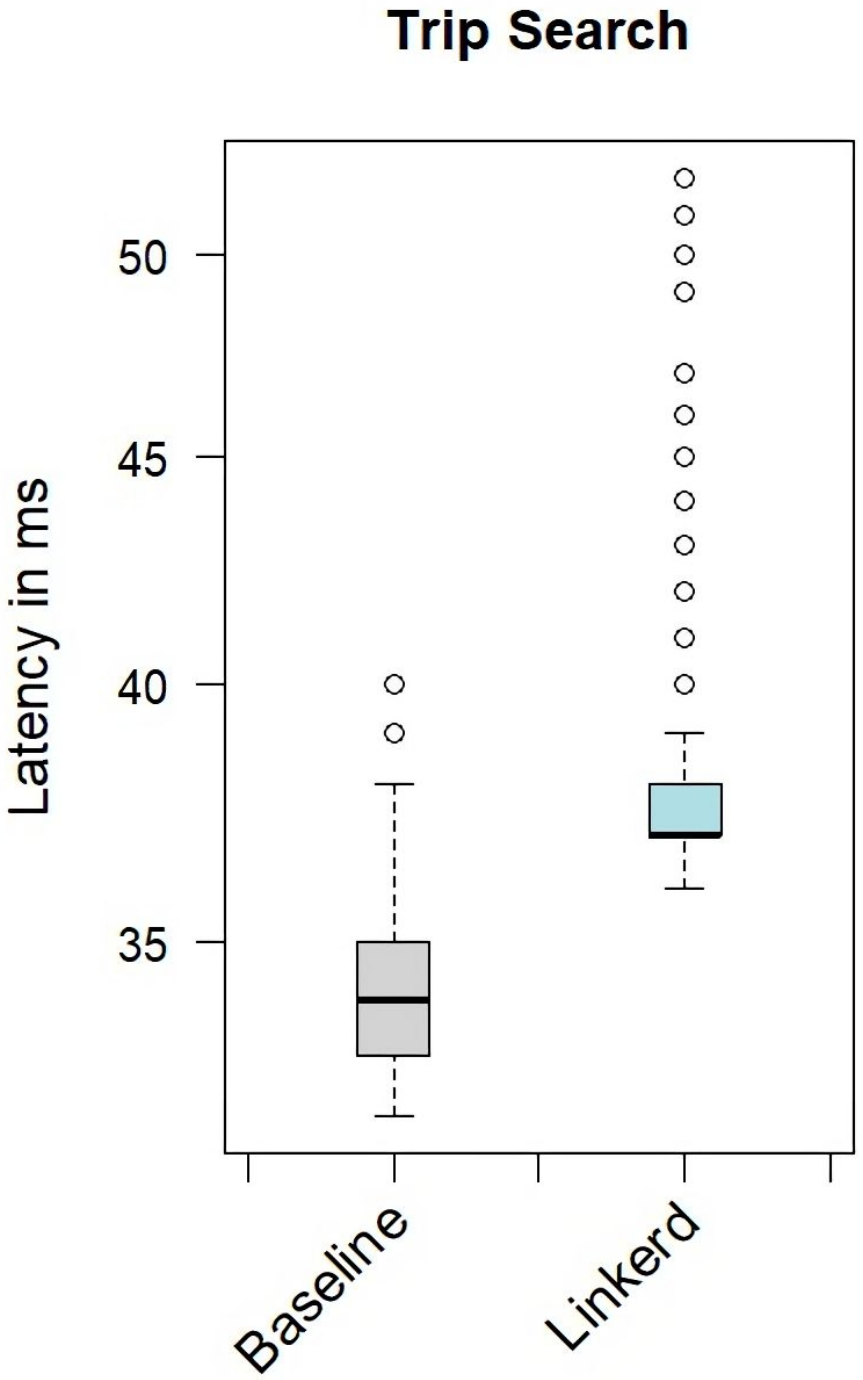
- Memory-Overhead von ca. *4,5 MB* in Ruhe- und Lastszenarien
- CPU-Overhead von ca. *2,1%* in Lastszenarien

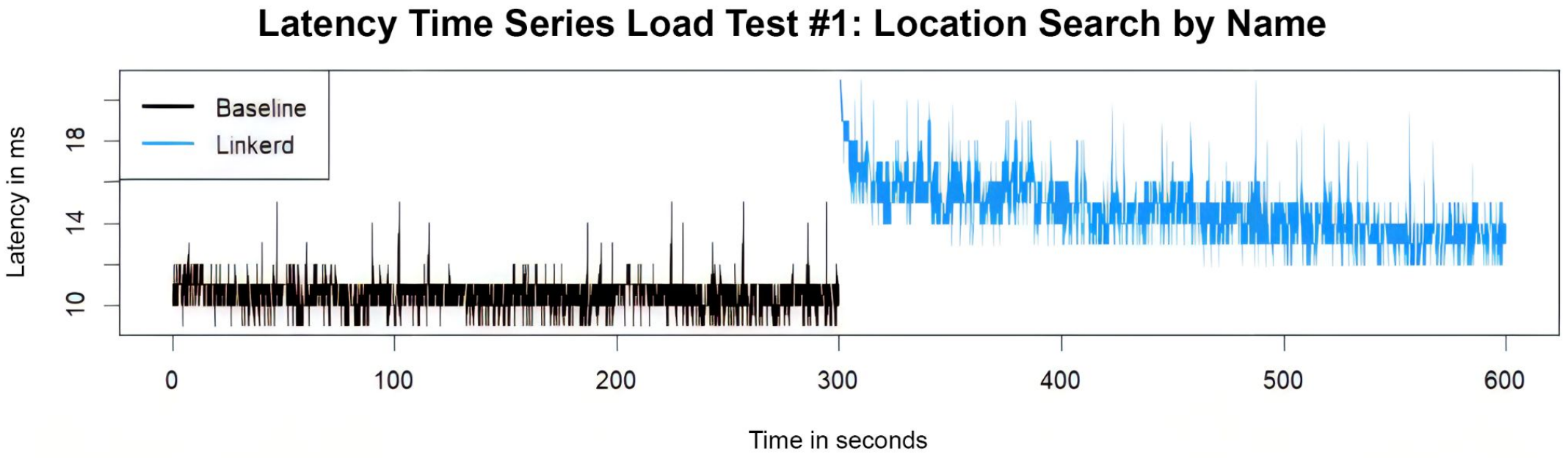
→ *Der Ressourcen-Overhead ist damit akzeptabel!*

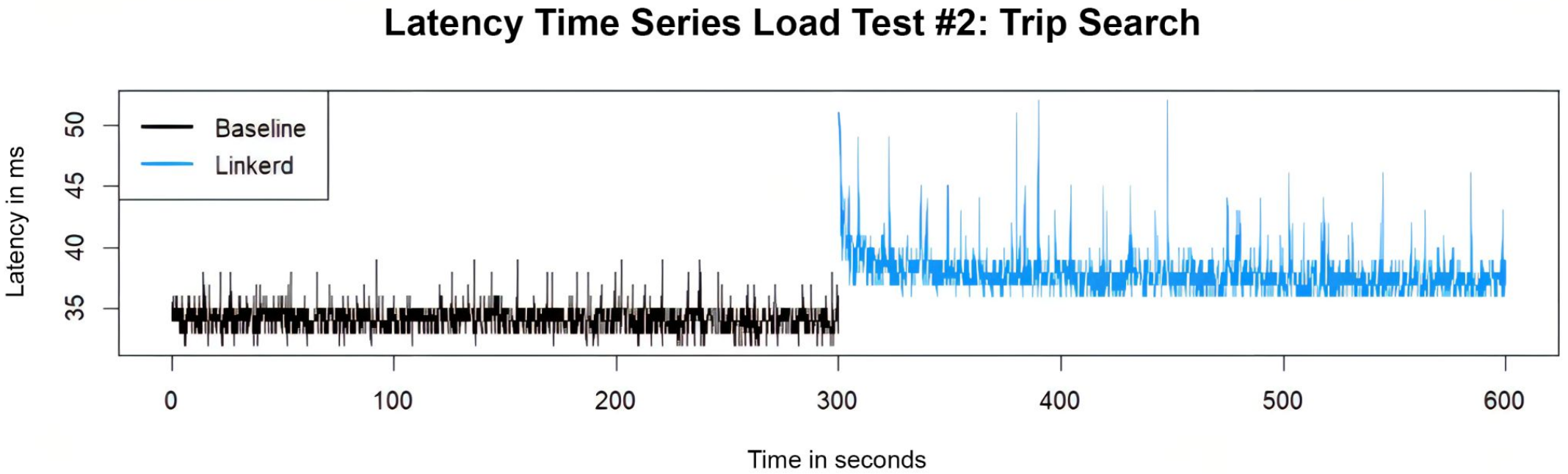
Feldbeobachtungen

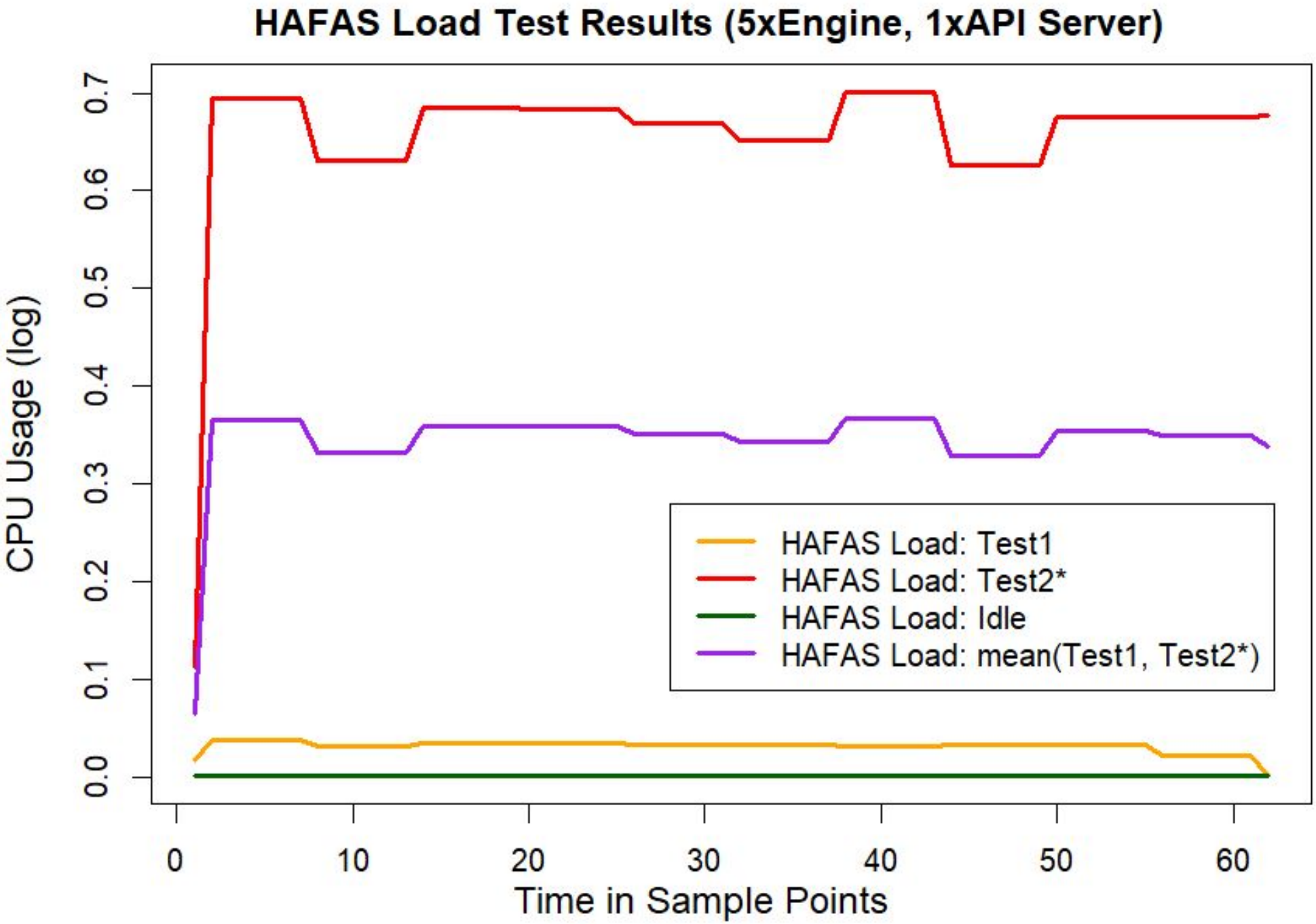
- Linkerd’s Monitoring behandelt HTTP bevorzugt / wenig Metriken für Raw-TCP
- Linkerd’s Upgrade-Prozess erlaubt keine Redundanz

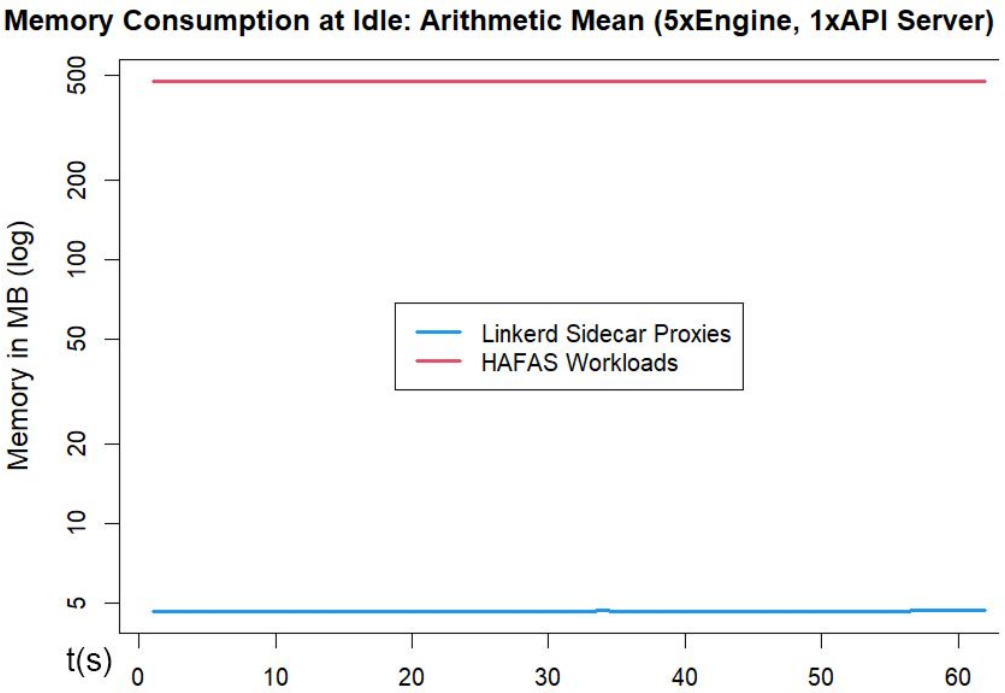

Location Search by Name
Latency in ms
20
18
16
14
12
10
Baseline
Linkerd

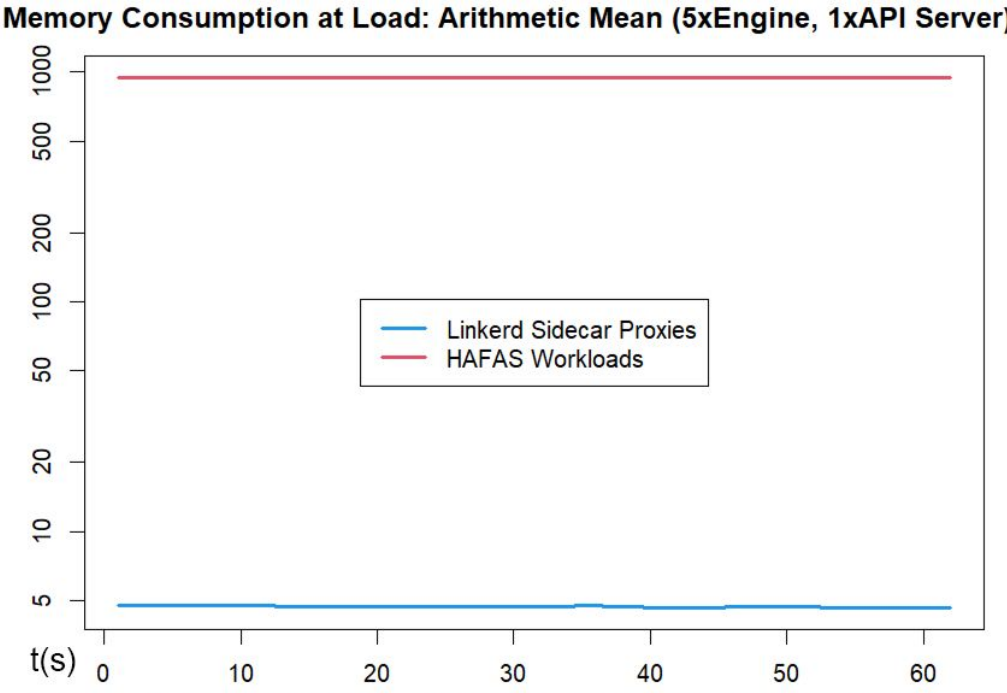

Trip Search
Latency in ms
50
45
40
35
Baseline
Linkerd

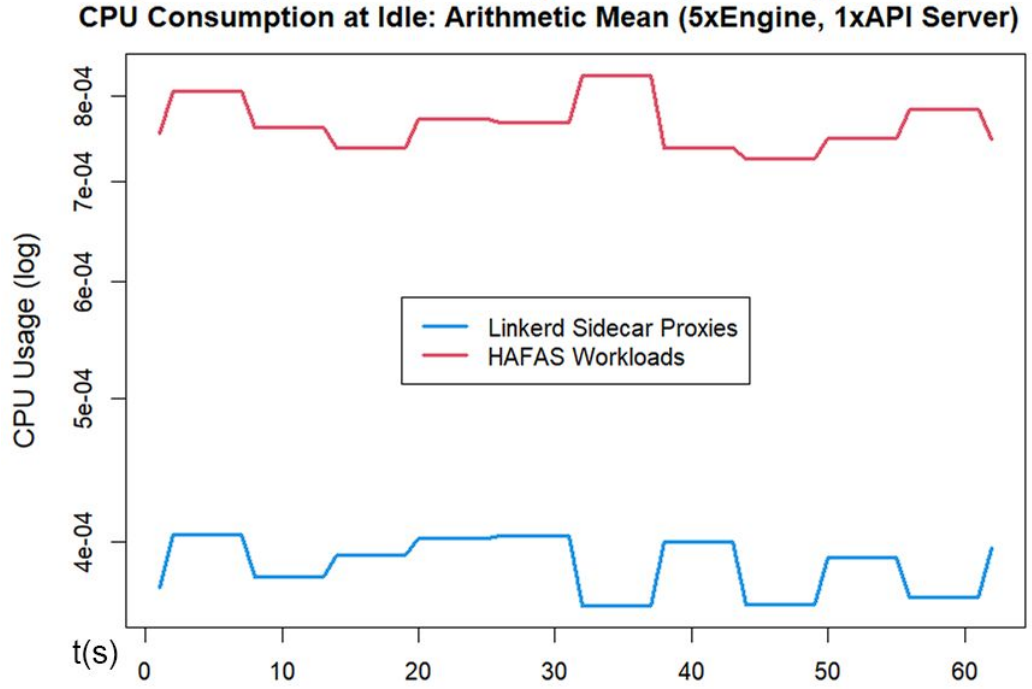

Latency Time Series Load Test #1: Location Search by Name
Baseline
Linkerd
Latency in ms
10
14
18
0
100
200
300
400
500
600
Time in seconds

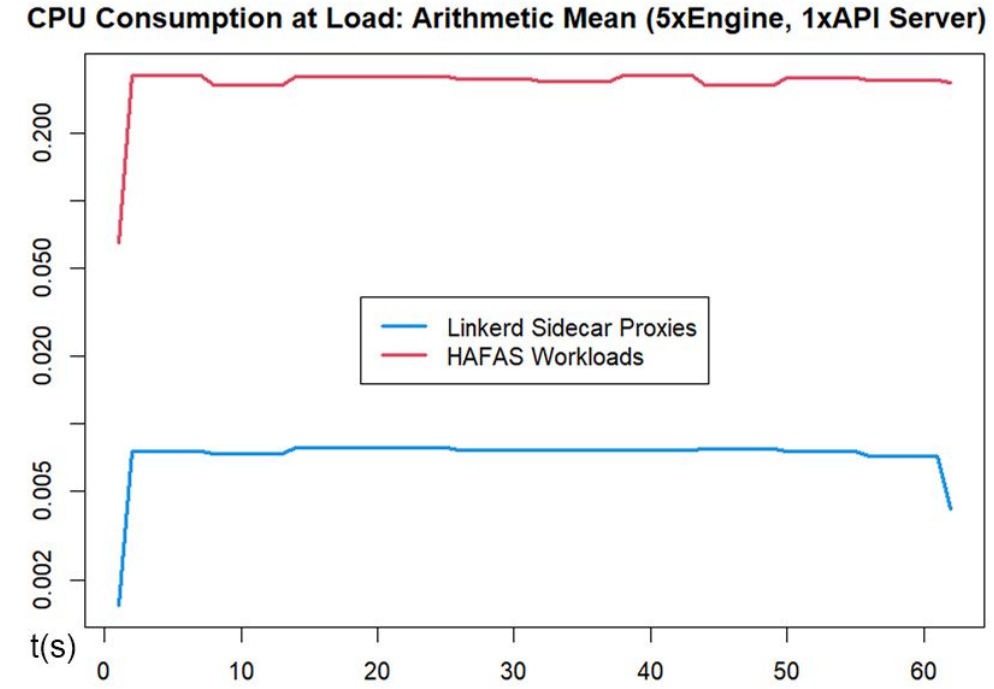

Latency Time Series Load Test #2: Trip Search
Baseline
Linkerd
Latency in ms
35
40
45
50
0
100
200
300
400
500
600
Time in seconds


HAFAS Load Test Results (5xEngine, 1xAPI Server)
CPU Usage (log)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50
60
Time in Sample Points
HAFAS Load: Test1
HAFAS Load: Test2*
HAFAS Load: Idle
HAFAS Load: mean(Test1, Test2*)


Memory Consumption at Idle: Arithmetic Mean (5xEngine, 1xAPI Server)
Memory in MB (log)
500
200
100
50
20
10
5
Linkerd Sidecar Proxies
HAFAS Workloads
t(s)
0
10
20
30
40
50
60


Memory Consumption at Load: Arithmetic Mean (5xEngine, 1xAPI Server)
1000
500
200
100
50
20
10
5
Linkerd Sidecar Proxies
HAFAS Workloads
t(s)
0
10
20
30
40
50
60


CPU Consumption at Idle: Arithmetic Mean (5xEngine, 1xAPI Server)
CPU Usage (log)
8e-04
7e-04
6e-04
5e-04
4e-04
Linkerd Sidecar Proxies
HAFAS Workloads
t(s)
0
10
20
30
40
50
60


CPU Consumption at Load: Arithmetic Mean (5xEngine, 1xAPI Server)
0.200
0.050
0.020
0.005
0.002
Linkerd Sidecar Proxies
HAFAS Workloads
t(s)
0
10
20
30
40
50
60

Fazit

Vorteile

- Verbessert die Sicherheit, Zuverlässigkeit, Belastbarkeit und Beobachtbarkeit in verteilten Softwaresystemen
- Umsetzung von Infrastruktur as Code durch Auslagerung der Konfiguration an CRDs

mTLS-Verschlüsselung und Sicherheit

- mTLS-Verschlüsselung stärkt die Sicherheitseigenschaften *Vertraulichkeit*, *Integrität* und *Authentizität*
- Ist ***keine*** allumfassende Sicherheitslösung; Überlegungen z. B. zum Schutz von ruhenden Daten und zum Schutz vor Ransomware sind unerlässlich

Überlegungen zur Implementierung

- Latenzzeiten und der Ressourcen-Overhead, die durch ein Service Mesh (z. B. Linkerd) entstehen, sind überschaubar und die Vorteile wert
- Einführung sollte durch ***nachvollziehbare Gründe*** vorangetrieben werden; ein Einsatz ohne klare Motivation ist nicht empfehlenswert

Fazit

Chancen und Risiken

- Funktionen in einem großen verteilten System einzuführen, ohne sie für jeden Dienst einzeln zu implementieren und konfigurieren
- Zusätzliche Komplexität, potenzielle Fehlerquellen und eine Lernkurve für Teams, die ein Service Mesh nutzen
- Projekt muss bedacht gewählt werden, kaum Kompatibilität in der Konfiguration zwischen Projekten

Technologielandschaft und Auswirkungen

- Mit Linkerd und Istio qualitativ hochwertige Projekte verfügbar
- Betriebskosten, Komplexitätsanalyse und eine sorgfältige Planung der Migration ist notwendig

Konzeptnachweis und Integration

- Anhand einer Fallstudie wurde gezeigt, dass ein Service Mesh in ein verteiltes System integrierbar ist und definierten Anforderungen erfüllt

Diskussion

Vielen Dank für Ihre Aufmerksamkeit!

Fragen können nun gestellt und beantwortet werden.