

Service Mesh [前編]

これからのクラウドネイティブアーキテクチャのあり方を探る
Exploring the Future of Cloud Native Architecture

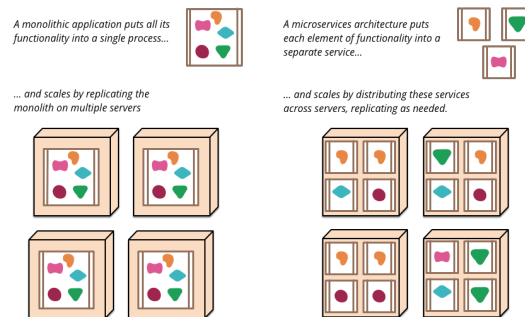
28 October 2022

Miya Kohno, Cisco Systems (mkohno@cisco.com)

Abstract

サービスマッシュは、マイクロサービス間の接続、制御、監視・観測性を可能にし、マイクロサービスの一貫した開発、展開、セキュリティ、およびスケーラビリティを提供します。しかし、マイクロサービスベースのアーキテクチャは、攻撃対象領域を拡大し、アプリケーションを新たな脆弱性や脅威にさらすため、アプリケーション・セキュリティに対するまったく新しいアプローチが必要になります。

本講演では、サービスマッシュの実装における課題、CNIとの関係を概観し、クラウドネイティブアーキテクチャのあり方に迫ります。



Self Introduction

シスコシステムズ合同会社 業務執行役員
ディステイングイッシュド システムズ エンジニア

ソフトウェア開発者としてキャリアをスタートさせる。現在シスコシステムズにおいて、主にサービスプロバイダ向けの技術支援、アーキテクチャ検討、コンサルティングを行っている。

分散コンピューティング、ネットワーク・システム・アーキテクチャ、モバイルシステムを専門とする。
2021年11月 MPLS Japan Award 受賞。2022年4月日本経済新聞「テクノロジストの時代」で紹介される。

Team SRv6 !!

- システム理論
<https://qiita.com/mkohno/items/ba8e207c225484814aff>
- Blog: ネットワークアーキテクチャ考
<https://gblogs.cisco.com/jp/author/miyakohno/>
- 🎵 Cellist



Agenda

- Service Mesh ざっくりおさらい
- K8s Networking model のおさらい
- Service Mesh と データプレーンの問題
 1. マルチクラスタ接続
 2. Service Mesh におけるデータプレーン
 3. Sidecar Proxy は必要か
 4. Kernel vs. User space 議論 – 再び！
- 前編のまとめ

セキュリティなども含めた Service Mesh の機能全般、そしてCisco Solution 概要については [後編] で！

Service Mesh ざっくりおさらい - 定義

A service mesh is a dedicated infrastructure layer that controls service-to-service communication over a network. This method enables separate parts of an application to communicate with each other. Service meshes appear commonly in concert with cloud-based applications, containers and microservices.

サービスメッシュとは、ネットワーク上のサービス間通信を制御する専用のインフラストラクチャレイヤである。この方法によって、分散アプリケーションの部分同士が互いに通信できるようになる。サービスメッシュは、クラウドベースのアプリケーション、コンテナ、マイクロサービスとの連携でよく登場する。

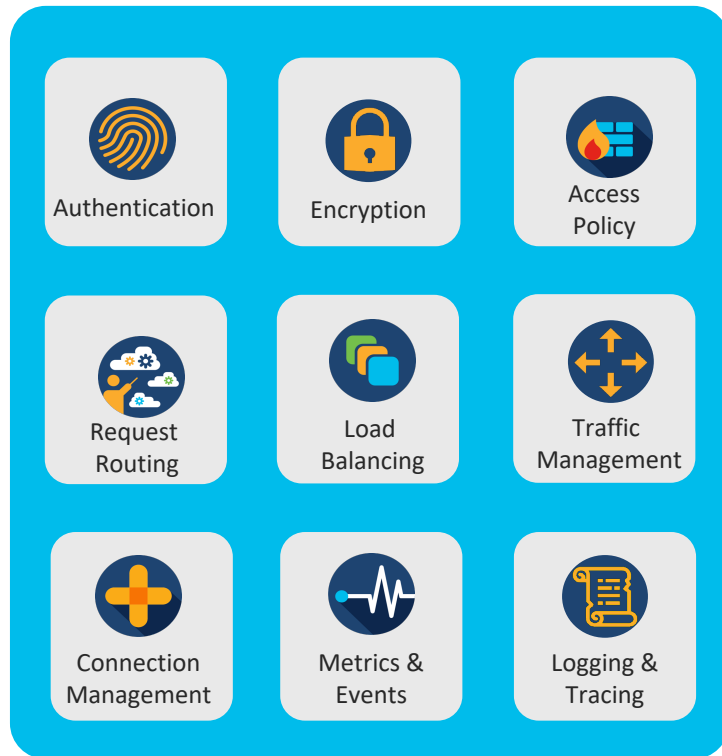
マイクロサービス間の接続性、可観測性、セキュリティを提供するインフラストラクチャレイヤ

Service Mesh ざっくりおさらい – 課題

Service Mesh 運用上の課題

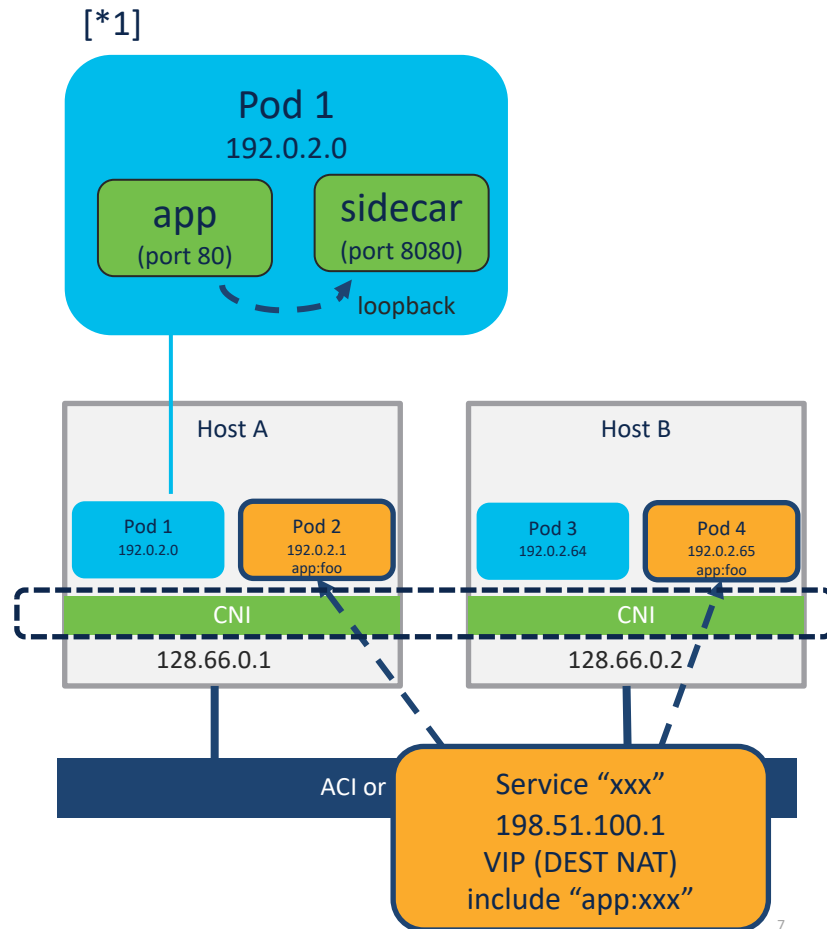
- ・ ライフサイクルマネジメント
- ・ 観測性のばらつきと断片化
- ・ マルチクラスターの課題
 - Availability (可用性)
 - クラスタ横断的なサービス発見
 - クラスタ間トラフィック管理ポリシー
 - マルチテナンシー
- ・ 非同期メッセージングの処理

Service Mesh



Kubernetes Networking Model

- 各Pod が、ユニークなIPアドレスを持つ
- Pod間は、NATなしに、相互に通信できる
- Pod内のすべてのコンテナは同じIPアドレスを共有し、ローカルのループバックを介して互いに通信する [*1]
- サービスによりPodを内部で負荷分散したり、クラスタ外のサービスから利用させるために外部IPアドレスを公開することができる [*2]
- コンテナ ネットワーク インターフェイス (CNI) は、接続とサービス・プロキシの実装を担当する
- IPAM (IP Address Management) により、Node IPレンジ、Pod IPレンジ、Service IPレンジ、外部IPレンジ、ロードバランサーIPレンジの管理を行う



[*2]

Service Mesh と データプレーンの問題

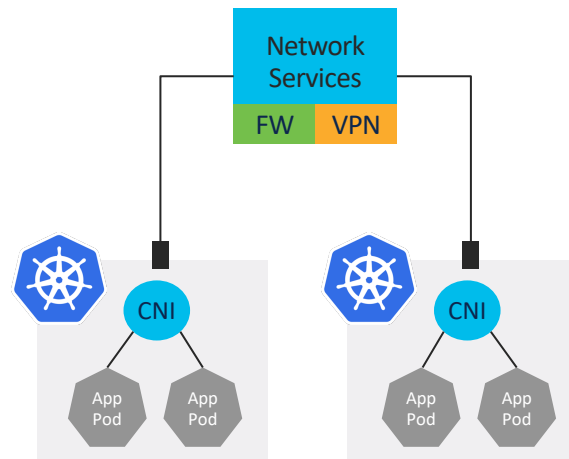
(1) マルチクラスタ接続

クラスタ間接続の必要性

- サービスのロードバランシング
- データレプリケーション
- サービスの依存関係
- パートナー提供モデルによるサービスのコネクティビティ





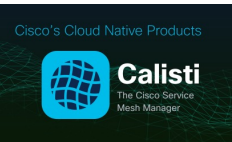
潜在的な問題

- 今日、これらクラスタ間を司るインフラの動作は、恣意的な仮定に基づいている
- イングレスロードバランサーを使用し、基本的なネットワーキングと名前解決で解決させる。
- VPC内／ネットワーク内：クラスタを同じVPC ネットワークに配置し、接続を容易にする。
- VPC間／ネットワーク間：ネットワーキングはすでに構築され管理されている(ハイブリッドクラウド、VPCピアリングなど)



Service Mesh と データプレーン の問題

(1) マルチクラスタ接続方式

カテゴリー	名称	特徴
Gateway/API Based	Submariner 	Gatewayベースで、Kubernetesサービスのマルチクラスタ接続性を提供。IstioなどService Meshのトランスポートして組み合わせて使用することも可能。 https://submariner.io/
CNI Based	Cilium Cluster Mesh 	eBPFベースのネットワーキング、可観測性、セキュリティを提供するCiliumによるCluster Mesh https://docs.cilium.io/en/stable/gettingstarted/#cluster-mesh
Service Mesh Based	Linkerd 	シンプル性を追求したService Mesh "Linkerd"による Multi Cluster Solution。データプレーンは、軽量のLinkerd-proxyを使用 https://linkerd.io/2.12/features/multicluster/
	Istio 	HTTP、gRPC、WebSocket、TCPの自動ロードバランシング機能、トラフィック制御機能を持つistioによるMulti ClusterデータプレーンにはEnvoyを採用 https://istio.io/latest/docs/setup/install/multicluster/
	Calisti (Cisco Service Mesh Manager) 	基本的にIstioに準じるが、トラフィック管理、可観測性、セキュリティ、シンプル化を追求 https://calisti.app/

Service Mesh と データプレーンの問題

(2) Service Meshにおけるデータプレーンとコントロールプレーン

サービスメッシュの主要機能

- サービスの発見
- ヘルスチェック
- ルーティング
- ロードバランシング
- 認証と承認
- 可観測性

これらは全てデータプレーンの機能と言える

サービスメッシュの機能を実現するためには、サービスインスタンスとの間を流れる全てのネットワークパケットを転送し、観察し、また条件付きで変換する。

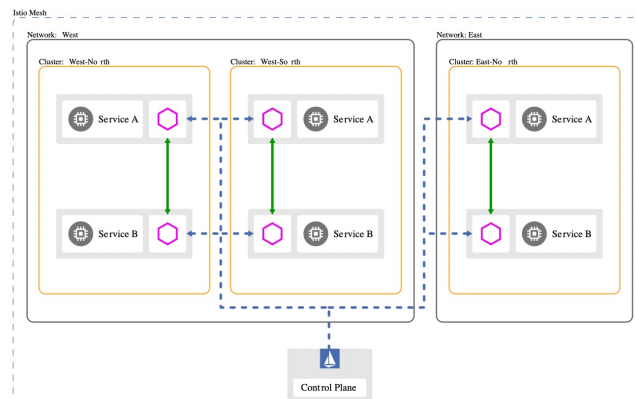
サービスメッシュのコントロールプレーンの役割

- メッシュ内のすべての実行中のデータプレーンに対する、ポリシーとコンフィギュレーションの提供
(つまり、何をやるかを決定)

[参考]

<https://cilium.io/blog/istio/>

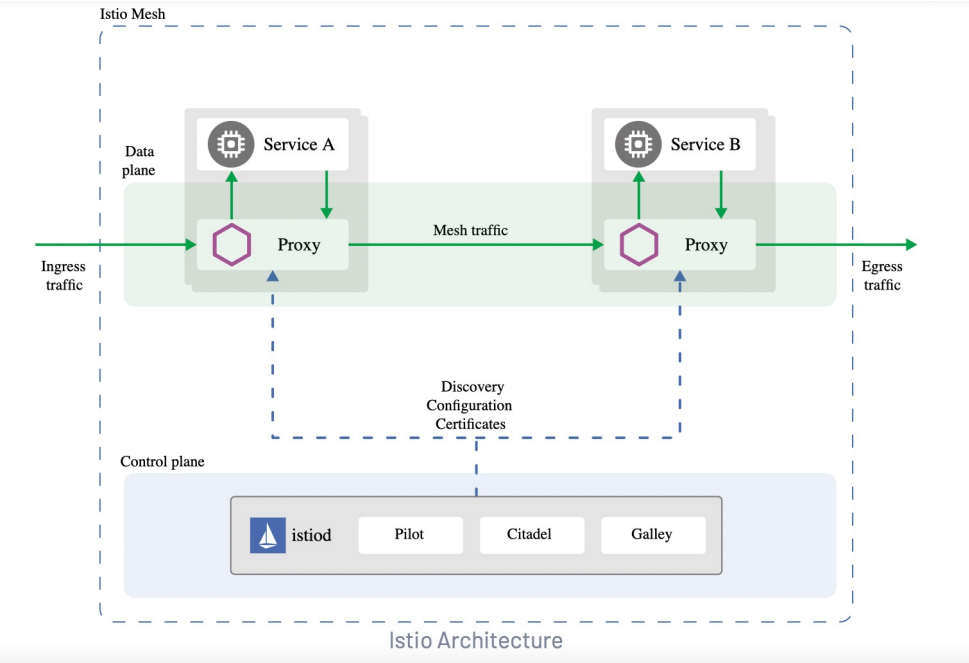
<https://blog.envoyproxy.io/service-mesh-data-plane-vs-control-plane-2774e720f7fc>



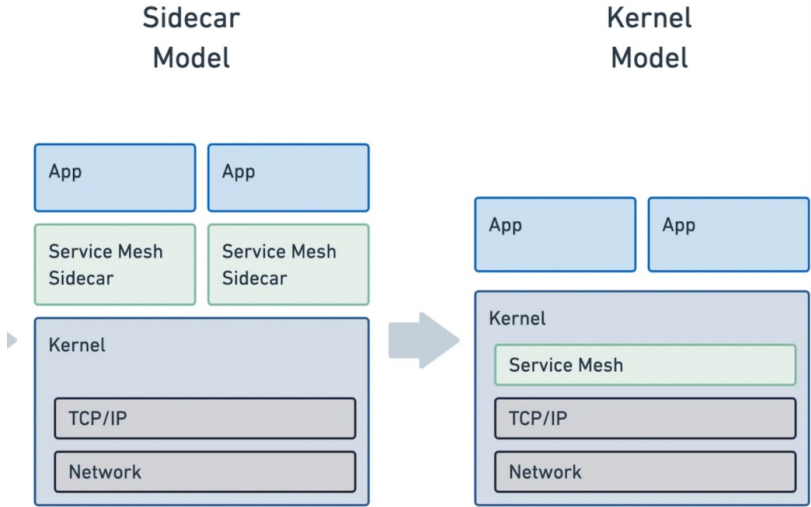
Service Mesh と データプレーンの問題

(3) Sidecar Proxy は必要か – Sidecar Free という考え方

Istio Architecture – Side Car Proxy / Pod

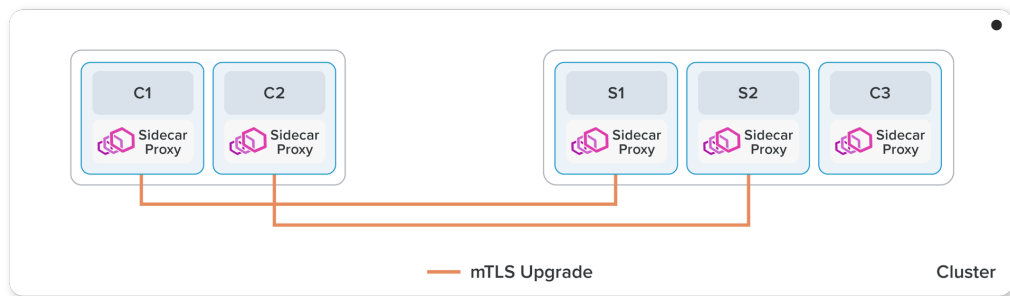


Cilium Mesh - Goodbye Side Cars !



Service Mesh と データプレーン の問題

(3) Sidecar Proxy は必要か – Pod毎ではなく node毎のproxy

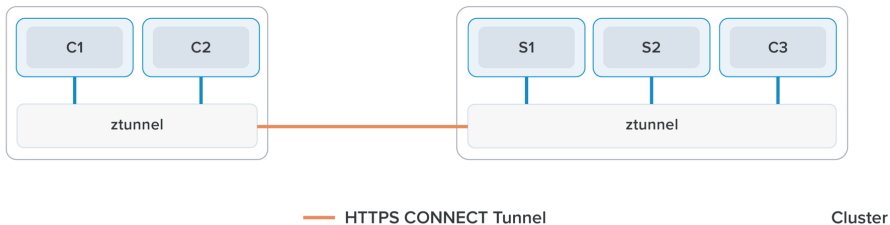


• 従来のIstio Model
(1 proxy / 1 pod)

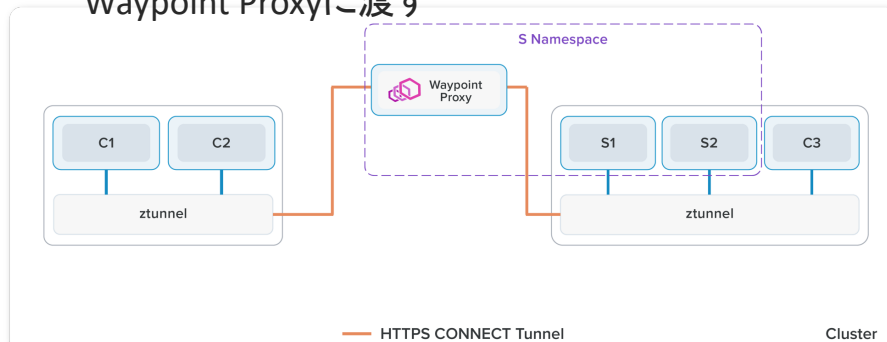


Istio Ambient Mesh

- Secure OverlayとL7機能を分離
- 共有エージェント(1 agent/1 pod) がSecure Overlay(Ztunnel)を生成

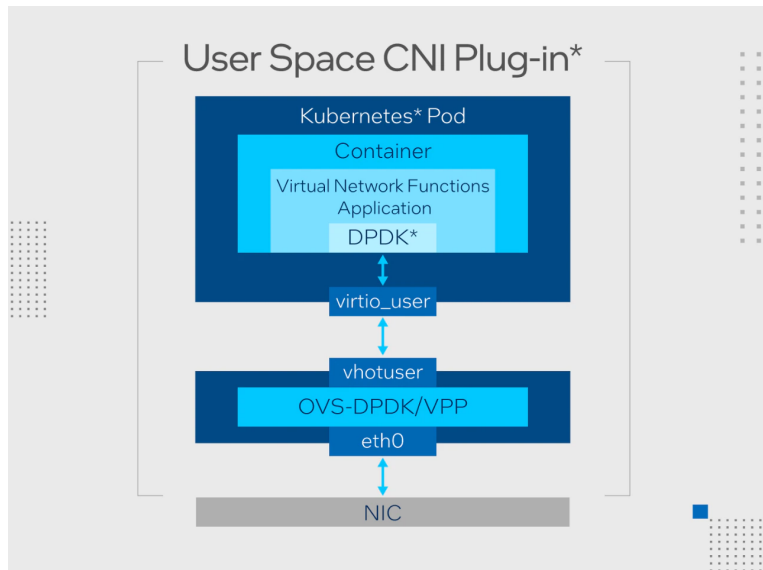


- L7処理が必要な場合は、Name space内のWaypoint Proxyに渡す



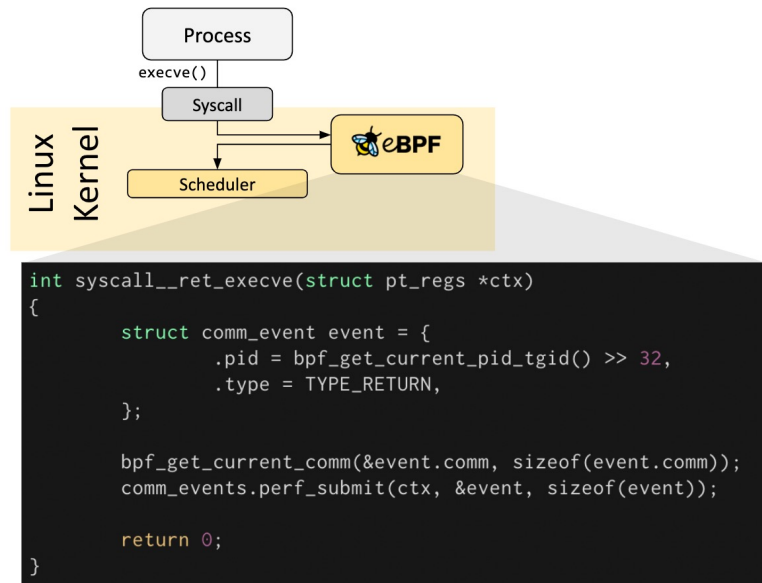
Service Mesh と データプレーンの問題

(4) Kernel vs. User space 議論 – 再び！



<https://www.intel.com/content/www/us/en/developer/articles/technical/user-space-cni.html>

- Kernelが既に持っている機能は使えないので再実装が必要



```
int syscall__ret_execve(struct pt_regs *ctx)
{
    struct comm_event event = {
        .pid = bpf_get_current_pid_tgid() >> 32,
        .type = TYPE_RETURN,
    };

    bpf_get_current_comm(&event.comm, sizeof(event.comm));
    comm_events.perf_submit(ctx, &event, sizeof(event));

    return 0;
}
```

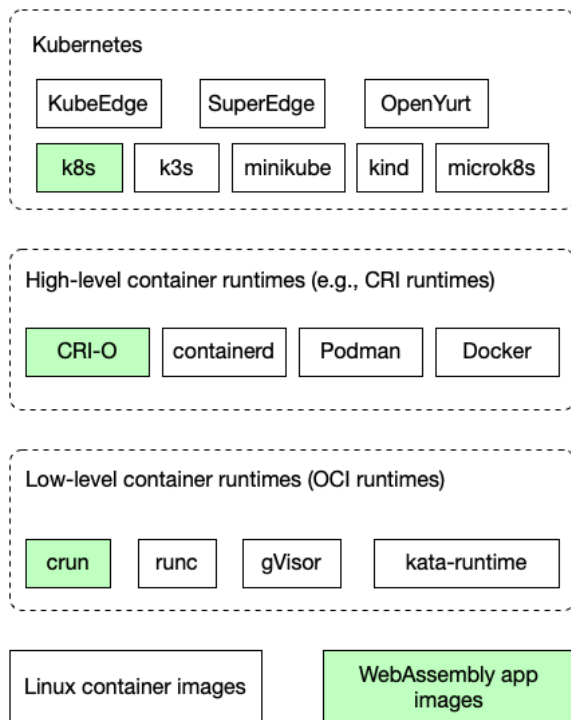
<https://ebpf.io/what-is-ebpf/>

- 条件によっては、性能が出にくい場合がある
- チューニング不完全 <https://www.secondstate.io/articles/ebpf-and-webassembly-whose-vm-reigns-supreme/>

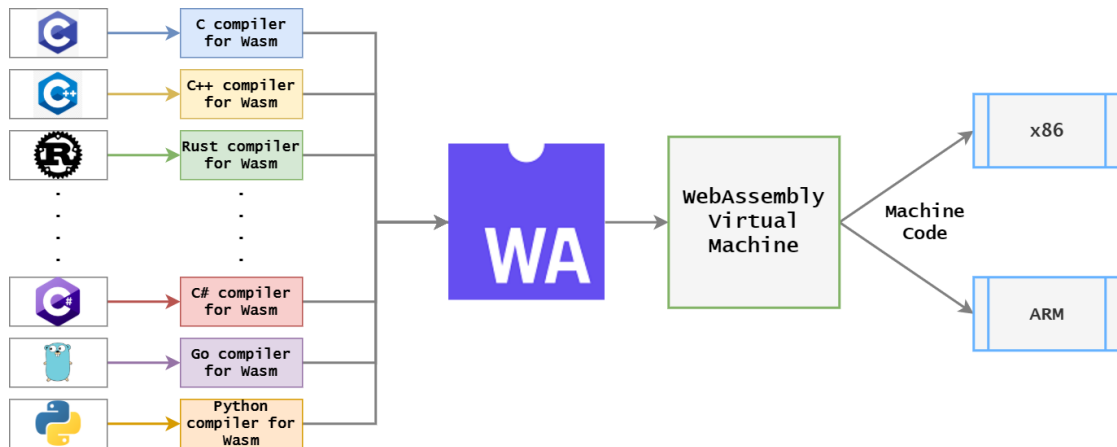
Service Mesh と データプレーンの問題

(4) Kernel vs. User space 議論 – 再び！ 新たな地平? WASM?

WASM – Web Assembly



The container ecosystem



- WASMは汎用のアプリケーションコンテナとして機能する
- 新たなサービスメッシュデータプレーンとしての応用

Sidecar Proxy

WASM runtime

<https://www.infoq.com/news/2022/01/ebpf-wasm-service-mesh/>

<https://istio.io/latest/docs/concepts/wasm/>

[前編] のまとめ

- Service Mesh を Data plane 中心に概観した
- 「Data plane がどうあるべきか」という問いに対する一定の答えはない
 - Sidecar proxy ?
 - Kernel vs User space ?
 - Data plane agnostic ?
 - Data plane optimization ?
- 要件・条件により、様々な組み合わせも考えられる

→ Control Plane / Data Plane に求めるものを整理しよう

→ まずは “What” に注目し、Control Plane から Service Mesh を検討しよう

(後編へ 😊)



The bridge to possible