2025 年度未踏アドバンスト事業

オンプレと連携可能な Wasm に特化したクラウドサービス
― Wasm による更新容易で軽量なエッジ AI 実行基盤 ―

1. 背景

近年 IoT デバイスの普及に伴い、データをクラウドではなくエッジで処理する必要性が高まっている。エッジで動作する AI モデルはエッジ AI と呼ばれ、エッジ AI の必要性が高まっている。

エッジ AI を運用する上で問題となるのが、リソース制約の厳しいエッジという環境でどう AI モデルやワークロードを更新し続けるかである。従来の IoT システムは「1 つのハードウェア＝1 つの用途」で設計されることが一般的であったが、ハードウェアリソースの有効活用や高度なサービス提供の観点から 1 台のハードウェアで異なるアプリケーションを動作させるニーズが増加している。また単一のタスクを継続的に実行するだけでなく、状況や時間帯、外部からのトリガーに応じて動的に役割（タスク）を切り替えることも必要になっている。

しかし既存の AI 実行基盤では、リソース（CPU、メモリ、ストレージ）が厳しく制限されたエッジデバイスにおいて、頻繁にアプリケーションを更新・管理することが困難である。一般的な AI モデルの実行において、環境構築の再現性を担保すべく、Python が提供するパッケージングマネージャーや Docker 等のコンテナ技術による配布が採用される。しかしそれらのバイナリやコンテナイメージサイズは大きく、帯域幅やストレージ、メモリ等を圧迫し、リソース制約のあるエッジではオーバーヘッドが大きい。

つまり既存の実行環境では、軽量で可搬性の高い AI モデルの実行方式が実現できない。今のエッジ AI にはこのようなモデルの配布形式とその実行環境が求められる。

2. 目的

本プロジェクトでは WebAssembly（Wasm）を採用することで軽量かつ可搬性の高いエッジ AI モデルの実行基盤を実現する。Wasm とは任意の OS・CPU アーキテクチャで実行可能なバイナリ形式である。Wasm はコンテナと同様にスタンドアロンな形でアプリケーションをカプセル化できる一方で、そのイメージサイズがコンテナよりも小さい。特にコンテナでは AI を実行するためのモデルに加えてその推論ランタイム等の依存関係を同梱している一方で、Wasm では実行するモデルと前処理・後処理のみがイメージ内に含まれる。その結果バイナリサイズは数十倍ほど小さくなる。

本プロジェクトでは、Wasm を用いたエッジ AI システムの開発・運用の基盤として、IoT デバイスやエッジサーバー上での AI モデルの実行ランタイムとそれらのオーケストレーションツールを開発した。

3. 製品・サービスの内容

本プロジェクトでは”Pipit”という Wasm を用いたエッジ AI プラットフォームを開発した。図 1 のように 3 つのコンポーネントから構成され、マイコンでの AI の実行、エッジサーバーでの AI の実行、マイコンからエッジサーバーまでのオーケストレーションをそれぞれサポートする。

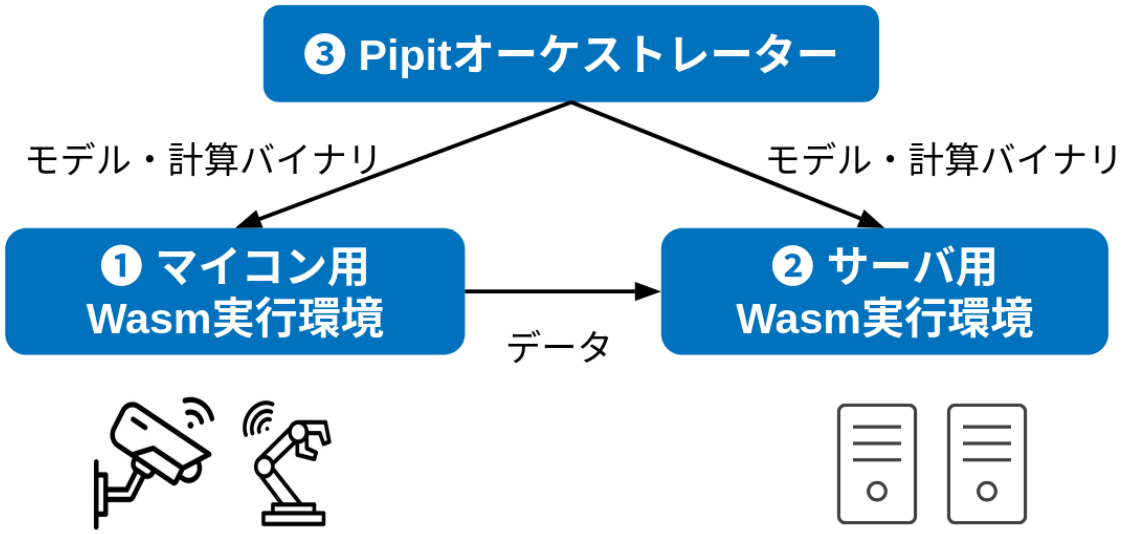

図 1 Pipit の構成

・waiot: マイコン用 Wasm 実行環境

リソース制約が厳しいマイコン（ESP32 等）向けの実行・デプロイ環境である。従来の組込み開発ではファームウェアとアプリケーションが一体化していたが、waiot ではこれらを分離する。図 2 のように waiot がファームウェアとして動作し、その上で Wasm 化されたアプリケーションが載る構造をとる。これにより、ファームウェア全体を書き換えることなく、上位の Wasm アプリケーション部分のみをネットワーク経由で安全に書き換えることができる。

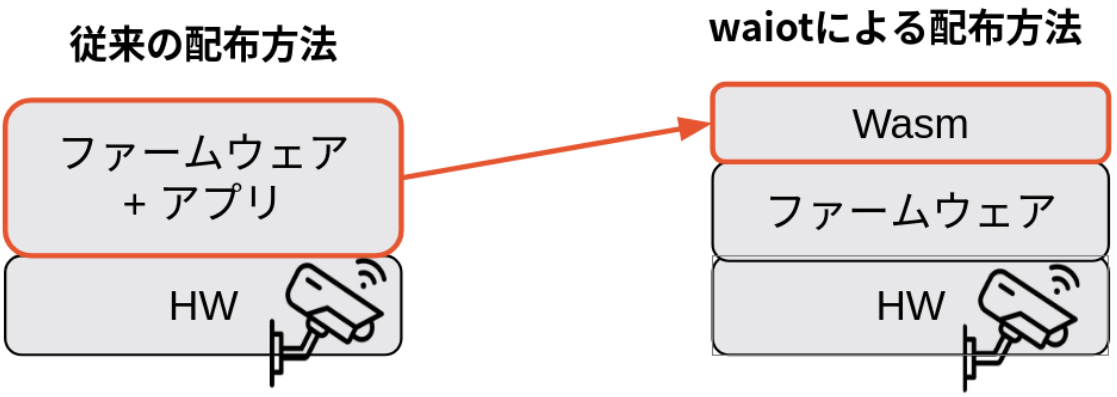

図 2 waiot によるアプリケーションの更新

・Mewz: サーバー用 Wasm 実行環境

Linux サーバーやエッジ GPU サーバー（Jetson 等）向けの実行環境である。Mewz は Wasm を実行するのに特化した OS として動作し、仮想マシン上で直接動作する。そのため例えばマルチテナントなエッジコンピューティング基盤でも十分な隔離性を持つ。また図 3 のように Wasm の実行に必要な機能のみを持ち Wasm をカーネル空間内で実行するため、軽量であり起動が高速である。

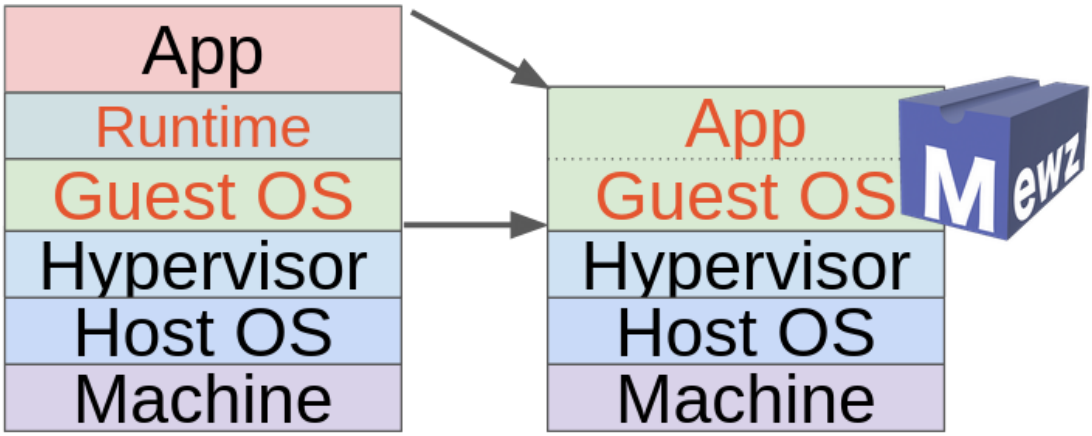

図 3 Mewz が Wasm をカーネル空間内で実行するイメージ

・Pipit オーケストレーター: エッジ AI のオーケストレーションツール

Kubernetes をベースに拡張された、システム全体の統合管理ツールである。Linux を搭載したサーバーだけでなく、通常は Kubernetes の管理下に置くことができない、Linux が動作しないマイコンデバイスまでも管理対象とする。図 4 のように統一的なインターフェースで Mewz が動作するサーバーや waiot が動作するマイコンに対し透過的に Wasm をデプロイすることが可能である。

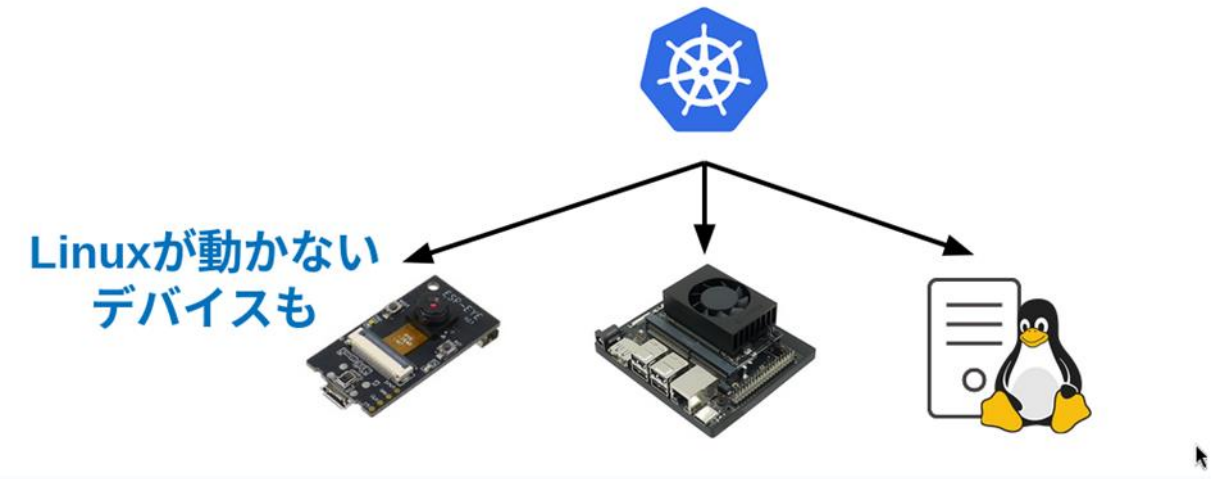

図 4 Pipit オーケストレーターによってマイコンから Linux サーバーまで統一的に管理する

4. 新規性・優位性

Pipit は WebAssembly をエッジ AI 領域に適用することで、以下のような利点を確立している。

・軽量性

既存の実行方式と比較して配布するバイナリのサイズを削減している。例えば ResNet を実行する Python のコードをコンテナ化すると Pytorch 等の推論ランタイムも含みイメージサイズは 25GB ほどになる。一方、Pipit で配布する Wasm バイナリは 100MB 程度で同じ機能を実現できる。またマイコンにおいては、カメラの画像を取得して推論するアプリケーションがファームウェア全体で 1.9MB であった。一方で Pipit では 452KB の Wasm バイナリで同じ機能を実現した。

・再起動不要なゼロダウンタイム運用

従来、マイコンのアプリ更新にはデバイスの再起動が必須であり分単位の時間を要した。waiot を用いることで再起動なしにアプリの入れ替えが可能である。これにより、秒単位で複数のタスクを切り替えることができるようになり、限られたハードウェアリソースを極限まで有効活用できるようになった。

・統合管理の範囲拡大

従来リソース制約の問題で不可能であったマイコンを Kubernetes のクラスタの一部として管理することを可能にした。これにより、エッジからクラウドまで一気通貫した DevOps が実現される。

5. 事業普及（または活用）の見通し

waiot と Pipit オーケストレーターについては OSS として公開し、さらなる開発を続行する予定である。

また Mewz については、Pipit で得た Wasm によるエッジ AI システムの知見と合わせて、通信事業者や CDN 事業者等に売り込む予定である。基地局や CDN 等のエッジで AI 推論を行う基盤を提供する際に、その実行環境として Mewz が使用されることを目指す。そのためにこういった事業者での研究開発等の形で実用化に向けて Mewz のさらなる開発を行う。

6. 期待される波及効果

エッジ AI の軽量な実行環境を提供することで、従来では通信帯域やデバイス性能の制約により AI 導入が困難であった環境での高度な AI 活用が可能となる。また、こういったエッジの環境からクラウドの環境までを Kubernetes により統一的に管理することが可能になる。

さらにエッジコンピューティングにおける AI 実行の最適化にも寄与する。Wasm により AI 推論の凝集性が向上し、基地局や CDN といったエッジコンピューティング基盤での AI ワークロードの集約率が向上することが期待される。

7. イノベータ名（所属）

・野崎 愛（東京大学大学院情報理工学系研究科システム情報学専攻）

・上田 蒼一朗（京都大学大学院情報学研究科）