

# サーバレス環境におけるエッジコンピューティングの高速化

TERM 中間発表

ONE B3 kino-ma

親: nyatsume

# 概要

サーバレス (FaaS, Function as a Service) 環境でのアプリケーション実行には, コンテナ (Docker 等) や VM が用いられている.

しかしそれらでは応答速度の面で課題が残る.

本研究では, **WebAssembly** を活用することで応答速度の改善を図る.

# 背景 - FaaS

サーバレスアーキテクチャ (Function as a Service, **FaaS**) の台頭

- AWS Lambda [1], Google Cloud Functions [2], IBM Cloud Functions [4] 等...

開発者が嬉しい [3]

- 環境構築の手間が省ける
  - 実行するプログラムを書くだけでよい
- 自動でデプロイ／スケールされる
- 通常のクラウドと違ってアイドル時間へのコストがない

# 背景 - FaaS + エッジコンピューティング

エッジコンピューティングに FaaS を採用することで、開発者・提供者ともにメリットがある

- 開発者:
  - 自分でエッジに配置(分散)する手間が省ける
  - 分散するためにかけていたコストが削減される
- 提供者:
  - データセンタでのリソース節約につながる
    - 仮想マシンを提供するより軽量
    - エッジのデータセンタでは資源が豊富でない場合もある [6]

# 問題

現在のサーバレスアーキテクチャでは、各アプリケーションのためにコンテナや VM を作成する

**その処理が遅い [5]**

→ 応答速度に厳しいユースケースに対応できない

- e.g.) 自動運転車

また、ユーザ体験に影響する [7]

# 目的

問題を解決するために、以下の要求を満たすこと

- **応答速度** がより短いこと
- **他のアプリケーションの動作に影響を及ぼさない** こと
  - FaaS では一つの物理マシンで複数のアプリケーションが動くと想定される
- **既存のアプリケーションからの 移行ハードルが低い** こと
  - 開発者の移行モチベーションが薄いと、提供者が FaaS のメリットを享受できない

# 要件

## 要求に対応する具体的な要件

- **応答速度:** 実行に必要な各処理ステップに要する時間が短いこと
  - アプリケーション起動処理
  - アプリケーション実行
  - 終了処理
- **他アプリケーションに影響しない:** 以下を制限／他と隔離できること
  - ファイルシステム
  - メモリ
  - CPU 資源
- **移行ハードル:** 既存のアプリケーションロジックを書き換える必要がないこと

# 関連研究




- コンテナの代わりに WebAssembly でサーバレスアーキテクチャを実現する [6]
  - 一度目の実行ではコンテナに勝る
  - 二度目以降ではコンテナが速い
- キャッシュを活用してコンテナの起動時間を大幅に削減する [7]
  - 条件によってはコンテナの起動が 1 ms を切る
  - 二度目以降のみ

課題: 初回とそれ以降の速度を両立できていない



# 提案手法

アプリケーション起動直後は WebAssembly インタープリタで実行し、一定時間経過後により軽量な方式に移行する

-  WebAssembly ランタイムでリソースを制限・隔離できる
-  メジャーな多くの言語がすでに WebAssembly をサポートしている
-  初回 & それ以降の速度を両立しうる
  
- 軽量な方式:
  - WebAssembly をより効率的なバイナリにコンパイルする
  - 開発者が事前に用意したコンテナ

# 実装(予定)

IBM Cloud Functions でも用いられている OSS のサーバレスシステム Apache OpenWhisk と近いアーキテクチャ

- Controller
  - HTTP リクエストを解釈し, 入力を Invoker に渡す
  - Invoker の出力をクライアントに返却する
- Invoker
  - 対象アプリケーションを実際に呼び出し, 結果を返す

# 評価

提案したシステムが要件を満たしていることを確かめるために、提案手法・コンテナのみ

- ・ WebAssembly のみで比較する

- 各ステップにかかる時間を、さまざまな規模や量のタスクで計測する
  - 起動処理
  - アプリケーション実行
  - 終了処理

# 今後の予定

- ~ 11 月末: 先行研究のより詳しい調査
- ~ 12 月上旬: Controller, Invoker の概形の実装 (設計) を行う
- ~ 12 月中旬: Controller の HTTP サーバ部分, Invoker 接続部分を実装する
- 並行: Invoker の呼び出し部分の実装を行う
- ~ 12 月下旬: 実装物の動作を検証し, 期待通りの動作を確認する
- ~ 1 月上旬: 評価内容をまとめ, 考察を行う
- ~ 1 月中旬: 考察結果をまとめ, 結論を導く

# まとめ

エッジコンピューティングにサーバレスアーキテクチャを採用することで、応答速度などの面で改善が見込まれる

しかし、既存のコンテナ式のシステムだと起動処理にオーバヘッドが存在する

本研究では、コンテナと WebAssembly を組み合わせることで、オーバヘッドを削減し応答速度の向上を図れるシステムを提案した

# 参考文献 (1)

[1] <https://aws.amazon.com/jp/lambda/>

[2] <https://cloud.google.com/functions>

[3] <https://www.hpe.com/us/en/insights/articles/serverless-computing-explained-1807.html>

[4] <https://www.ibm.com/cloud/architecture/architectures/open-cloud-platform/>

[5] Pelle, István, et al. "Towards latency sensitive cloud native applications: A performance study on aws." 2019 IEEE 12th International Conference on Cloud Computing (CLOUD). IEEE, 2019.

[6] Hall, Adam, and Umakishore Ramachandran. "An execution model for serverless functions at the edge." Proceedings of the International Conference on Internet of Things Design and Implementation. 2019.

## 参考文献 (2)

[7] Du, Dong, et al. "Catalyzer: Sub-millisecond startup for serverless computing with initialization-less booting." /Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems/. 2020.

# プロジェクト

(仕組みをよくわかっていないです. ごめんなさい)

- システム公開? : 湘南自治会投票システムの開発・運営
  - 進行中
- ハッカソン: Digital Hackday 2021, Acompany 賞