IPA

2023 年度 未踏 IT 人材発掘・育成事業 採択案件評価書

1. 担当 PM

曾川 景介（株式会社メルコイン取締役 CINO）

2. クリエータ氏名

上田蒼一朗（京都大学 工学部 情報学科）
野崎愛（東京大学 情報理工学系研究科 システム情報学専攻）

3. 委託金支払額

2,736,000 円

4. テーマ名

Wasm を実行する unikernel と Wasm コンパイラ

5. 関連 Web サイト

- Mewz のソースコード ：https://github.com/mewz-project/mewz
- Wasker のソースコード：https://github.com/mewz-project/wasker

6. テーマ概要

Wasm（WebAssembly）とは Web ブラウザ上で実行可能な仮想命令セットである。従来ブラウザ上で実行されるプログラムには JavaScript に代わってより高速な実行形態を求めて Wasm が開発された。近年では Wasm の実行環境はブラウザにとどまらず、各種 OS 上で直接実行するランタイムが登場している。

unikernel とは軽量なカーネルの構成方法の一種である。unikernel では単一のアプリケーションがビルド時にカーネルと静的にリンクされ、1 つのカーネルイメージにまとめられる。よって unikernel 上では 1 つのアプリケーションのみ実行し、カーネルもアプリケーションも単一のアドレス空間で、カーネルモードで動作する。汎用性を重視した Linux や Windows などの一般的な OS と比較して、unikernel はコードサイズやメモリフットプリントの小ささ、常にカーネルモードで動作することによる実行速度の向上、起動時間が短いといった利点を持つ。

本プロジェクトでは Wasm 及び unikernel の利点を活かすことで、既存のコンテナ技術を置き換えるような、Wasm の実行に特化した unikernel と Wasm

をオブジェクトファイルに変換する AoT コンパイラ、そしてその unikernel をコンテナとして扱うためのソフトウェアを開発した。本プロジェクトの成果では、仮想マシン上で Wasm を、類似の目的を持つ Wasm Runtime よりも高パフォーマンスで実行することが可能となった。

7. 採択理由

本提案は Wasm（WebAssembly）を実行できる unikernel とコンパイラの開発を行うものであり、期待される具体的な効果として以下が挙げられる。

- 本提案をベースとした PaaS プラットフォーム
- サーバレスコンピューティングでの活用
- Functional as a Service などのクラウドコンピューティングに新しい実装形態を提案する
- コンテナレイヤーにおける Linux カーネル依存からの脱却（あるいは、Linux カーネルへの依存度の低いプログラムをわざわざ Linux カーネル上で実行することのオーバヘッドの低減）

dotCloud, Inc.（現 Docker, Inc.）の創設者で docker の作者である Solomon Hykes 氏も、2008 年に Wasm/WASI があれば docker を作っていなかっただろうという主旨の発言を Twitter でしており、まさに本提案が docker の作者が描いたミッシングピースを埋めるものになると考えられる。

また、本提案についてさらに付け加えるべきことは、具体的な成果物以外にももう一つの重要な成果が生まれてくると考えられることである。本提案の実装を通じてシステムインターフェースの標準化への貢献が期待できる。Wasm を実行できる環境を広げることで Wasm をさらにどこでも実行できるようにし、コンピューティングの未来を切り開いてほしい。

8. 開発目標

上記で Wasm の軽量性について述べたが、パブリッククラウド上ではこの軽量性について課題がある。それは、パブリッククラウド上では隔離性のため、Wasm を仮想マシン上で実行しなければならないことである。パブリッククラウド上では多くのユーザがクラウドベンダの用意したデータセンター上でワークロードを実行するが、ユーザ間の干渉を防止するためにそれぞれのワークロードを十分に隔離しなければならない。多くのクラウドベンダでは、ハイパーバイザを用いて仮想マシンとして計算リソースを分離することでこれを実現している。そのため、Wasm をパブリッククラウド上で実行する際には、仮想マシン上で実行する必要がある。しかしこのような状況では、前項で述べた軽量性とは裏腹に、仮想マシンのレイヤまで含めた実行環境全体は軽量ではなくなる。その

ため実行環境の大きさゆえに性能のオーバーヘッドが発生するという問題がある。本プロジェクトは unikernel の特長を活かして、パブリッククラウド上で仮想マシンによる隔離性を維持したまま、実行環境のオーバーヘッドを削減して Wasm を実行する環境を実現することを目的とする。

9. 進捗概要

本プロジェクトでは大きく 2 つの成果物を開発した。Wasm の実行環境としての WASI を備えた unikernel である「Mewz」とその Mewz と Wasm バイナリを静的にリンクするためのコンパイラ「Wasker」である。

- Mewz

  Mewz は先述の通り、WASI の実装を備えた unikernel である。Mewz とリンクされた Wasm アプリケーションが WASI の関数を呼ぶと、Mewz に実装された WASI の関数へ処理が移る。これによって Mewz は Wasm に対して WASI API の機能を提供し、Wasm を実行する。Mewz がサポートする CPU アーキテクチャは現在 x86 64 のみである。

  Mewz は Wasm の実行に特化した unikernel なので、その機能は WASI を通してのみ提供される。よって、Mewz は WASI の実装に必要な機能のみを実装している。これによって Mewz が持つ機能を最低限に抑えている。例えば、現在策定されている WASI ではスレッドの API がないため Mewz にもスレッドの機能は実装されていない。そのため Mewz はスレッドのスケジューリング機能も持たない。これによりスレッドの生成や切り替え、スケジューリングによるオーバーヘッドを削減している。Mewz は Zig 言語で開発を行った。Zig は C 言語のライブラリと組み合わせることが容易であるという特長がある。この性質を利用して、組み込み向けの TCP/IP プロトコル・スタック実装である lwIP や組み込み向けの標準 C ライブラリ実装である Newlib などのライブラリを利用している。

- Wasker

  Mewz と Wasm バイナリを静的にリンクするため、Wasm バイナリをターゲット CPU アーキテクチャ上のネイティブコードに変換する AoT コンパイラが Wasker である。Wasker は Wasm バイナリをオブジェクトファイルにコンパイルする。その際に WASI の関数を未解決シンボルとして残す。そのため、Wasm から変換されたネイティブバイナリには WASI の実装が含まれない。Wasker は LLVM を用いてネイティブコードを生成している。これにより、Wasker は LLVM がバックエンドとして対応している多様な CPU アーキテクチャに対応できる。Wasker の実装は Rust で行った。Wasm バイナリを扱う Rust のライブラリが豊富に存在し、これらを利用するため

Rust を採用した。

Mewz と Wasker の 2 つの成果物を期間中に完成することができた。加えて、Kubernetes などのコンテナ管理ソフトウェアで Mewz を管理することを可能にするソフトウェア「containerd-shim-mewz」も期間中に開発することができた。

10. プロジェクト評価

unikernel に Wasm を組み合わせる大胆な構想の提案を、外部公開できるレベルでアウトプットを達成した。加えてグローバルなコミュニティに対してもリーチアウトを行い、認知を得ることに成功した。作るものが非常に多かったが 2 人のチームワークでぎりぎり間に合わせることができたことも良かった。

11. 今後の課題

Mewz については一部の未対応の WASI 関数への対応が挙げられる。Wasker についてはサイズの大きい Wasm バイナリをネイティブコードに変換する際に時間がかかるため、並列化による高速化の余地がある。今後は各ソフトウェアの改善を進めるとともに、実際のクラウドコンピューティングにおける利用や Wasm／WASI のコミュニティ活動を通じて最新の動向を追いかけてキャッチアップを進める必要がある。今回の成果を基盤に Wasm や unikernel の更なる応用領域の探索などが挙げられる。