IPA

2024 年度 未踏 IT 人材発掘・育成事業 採択案件評価書

1. 担当 PM

田中 邦裕（さくらインターネット株式会社 代表取締役社長）

2. クリエータ氏名

髙名 典雅（筑波大学 理工情報生命学術院）

3. 委託金支払額

2,880,000 円

4. テーマ名

RISC-V の拡張を仮想化できるハイパーバイザの開発

5. 関連 Web サイト

- ハイパーバイザ：https://github.com/Alignof/hikami
- デコーダ：https://crates.io/crates/raki
- アロケータ：https://github.com/Alignof/wild-screen-alloc
- 自動生成ツール：https://github.com/Alignof/ozora

6. テーマ概要

RISC-V はモジュール化された仕様である「拡張」が次々に策定されているものの、ハードウェアへの実装がほとんど進んでいない状況にあり、多くの拡張が活用されないまま放置されているという課題がある。本プロジェクトでは、これらの拡張機能を仮想化することで、ハードウェアに依存せずに拡張機能を導入・管理できる環境を整備した。

RISC-V の拡張モジュールは、命令セットの追加や CSRs（Control and Status Registers）の変更など多様な仕様を含んでいるが、ハイパーバイザ上で仮想化モジュールとして再現することで、ユーザは必要な拡張機能をオンデマンドで利用できるようになる。拡張モジュールは個別に設計されており、ハイパーバイザ上で機能単位に導入可能な形で提供される。ユーザは必要なモジュールを選択してインストールすることで、対象の拡張機能を仮想的に利用できる構成となっている。

本プロジェクトでは、仮想化レイヤーの最適化、拡張モジュールの動作検証、

モジュール管理インタフェースの構築、セキュリティ強化など、ソフトウェアの安定性と利便性を向上させるための多岐にわたる取り組みが行われた。

7. 採択理由

RISC-V は、オープンソースの命令セットアーキテクチャであり、基本命令セットにさまざまな拡張命令を追加できることから、多くの命令セットが作られているが、実際にはその大半が利用されていない状況にある。本提案においては、その使われていない拡張命令セットに対応するハイパーバイザ内のソフトウェアをモジュールとして提供し、それを組み合わせるだけで拡張命令セットを利活用できることを目指しており、ユーザがより簡単に拡張命令セットを利活用できることを意味している。

未踏事業においては、CPU を利用するだけでなく、CPU 自体を開発したり、その利活用を簡単に行うためのソフトウェアを開発しようとする提案はとても歓迎すべきものであり、特に本提案のように、未だ多くの取り組みがなされていない分野において、新たなエコシステムを構築するという提案は重要である。提案者は、低レイヤーの分野に対して造詣が深く、かつ興味関心が強く、未踏事業を通じて本プロジェクトを実現することで、国内外の RISC-V のコミュニティに対して、大きな刺激を与えるであろうと判断して採択した。

8. 開発目標

本プロジェクトの目標は、RISC-V の拡張機能をソフトウェア上で仮想化し、ハードウェア実装に依存せずにユーザが必要な拡張機能を柔軟に導入・利用できる環境を実現することである。

RISC-V では、ISA（命令セットアーキテクチャ）拡張や Non-ISA（ハードウェア機能、割り込み、ABI など）拡張など、多種多様な拡張仕様が策定されているが、これらの拡張はハードウェア実装がほとんど進んでいないため、利用が限定されている。そこで本プロジェクトでは、これらの拡張機能をハイパーバイザ上でソフトウェアモジュールとして再現し、モジュール単位での導入・管理を可能とすることで、ユーザが必要な拡張を選択的に利用できる柔軟な仮想化環境の提供を目指した。

さらに、モジュールの管理インタフェースの強化、仮想化レイヤーの最適化、モジュール間の相互干渉の防止、セキュリティ対策の強化など、拡張機能の利用を安全かつ効率的に行える環境を提供することも開発目標の一環である。最終的には、拡張機能の仮想化によって RISC-V エコシステムの活性化を促進し、より多くのユーザが拡張機能の恩恵を受けられる環境の実現を目指した。

9. 進捗概要

本プロジェクトで開発したハイパーバイザは Type-1 の軽量な設計であり、ゲスト OS の起動、メモリ管理、デバイスのエミュレーション、IOMMU（Input-Output Memory Management Unit）のサポート、命令や CSR のエミュレーションなど、仮想化環境で必要な基本機能を備えている。RISC-V はオープンソースの命令セットアーキテクチャ（ISA）であり、さまざまな拡張機能が次々に策定されているものの、ハードウェアレベルでの実装が進まず、多くの拡張機能が未活用の状態にある。この課題に対応するため、本プロジェクトでは拡張機能をハイパーバイザモジュールとして仮想化し、物理ハードウェアに依存せず、ユーザが必要に応じて拡張機能を導入・管理できる環境を構築した。

開発初期段階では、RISC-V の拡張仕様を ISA（命令セット）拡張と Non-ISA（割り込み管理、ハードウェア機能、ABI 関連機能）拡張の 2 つに分類し、それぞれ独立した仮想化モジュールとして設計する方針が決定された。ISA 拡張では、命令セットの追加や CSRs の変更、割り込み管理の拡張が仮想化対象となり、Non-ISA 拡張では、割り込み制御やメモリマネジメントの仮想化が重点的に扱われた。拡張機能の仮想化には、RISC-V H 拡張（Hypervisor Extension）が活用され、仮想マシンの起動、割り込み管理、メモリ管理などの処理効率が大幅に向上した。仮想環境内では、仮想 CPU やメモリマネージャーが拡張機能をエミュレーションし、ホストマシンのハードウェアに依存せず、最新の拡張機能を仮想マシン上で利用できることが確認された。また、複数の拡張機能を同時に仮想化して利用する際の競合管理や、仮想マシンのリソース管理も強化され、拡張機能の導入・運用がより柔軟になった。

拡張モジュールの導入・管理は CLI（Command Line Interface）を中心に行う設計となっており、拡張モジュールのビルド、インストール、アンインストール、依存関係の管理、モジュールのバージョン管理などが自動化された。CLI の自動化機能により、開発者はスクリプトを活用して効率的に拡張機能のテストと導入を行うことができる。将来的には GUI でのモジュール管理機能も構想されており、非エンジニア層への普及も視野に入れた開発が検討されている。モジュールの競合や依存関係の解決についても、今後は優先度制御や競合緩和機構の導入が期待されている。

中間フェーズでは、拡張モジュールの動作検証とパフォーマンスの最適化が進められた。複数の拡張機能を同時に利用する環境を想定し、各モジュールの動作安定性、競合回避の設計、エラー発生時のリカバリー処理の検証が行われた。特に競合設計においては、拡張機能の干渉を最小限に抑える構成が試行され、モジュール同士の干渉を最小限に抑えることが可能となった。また、VMM（Virtual Machine Monitor）の最適化により、モジュールのロード・アンロード時の負荷軽減、ページテーブルのキャッシュ効率向上、仮想マシン起動時の遅延抑制などが行われた。仮想環境でのパフォーマンスについては、拡張導入時のオーバーヘ

ッドを想定しつつ、今後の最適化に向けた検討が進められている。

この段階で導入された Sail 言語を用いた自動生成ツールは、プロジェクトの大きな成果の一つである。Sail は ISA の仕様記述言語であり、RISC-V の仕様策定にも使用されている。本プロジェクトでは、Sail を用いて ISA 仕様から拡張モジュールのデコーダ、ハイパーバイザモジュール、テストスクリプトを自動生成する仕組みが開発された。この自動生成ツールの導入により、開発者は手作業によるミスを防止しながら迅速に拡張機能を追加できるようになり、モジュールの開発工数が大幅に削減された。自動生成されたモジュールは、仮想環境での動作確認後にハイパーバイザに統合され、エラーの早期検出と安定性の向上が実現された。また、Sail 言語を活用したことで、RISC-V の拡張仕様の変更にも柔軟に対応できるようになり、新しい仕様への追従が迅速化された。

最終フェーズでは、ユーザテストとフィードバックをもとに拡張モジュールの安定性向上とセキュリティ強化が進められた。仮想化環境では、拡張モジュールのバージョン管理、依存関係の明示的な制御や競合の手動調整といった実用的な管理が試行され、モジュール同士の相互干渉を最小限に抑えた。また、将来的な展望として、動的・静的解析による異常検知やセキュリティチェックの導入が検討されている。モジュールのバージョン更新は手動で行われ、ユーザが必要に応じて最新のモジュールに切り替えることが可能となった。

仮想環境での拡張モジュールのパフォーマンス評価では、複数の拡張機能を同時に仮想化した際の応答時間、競合時の動作安定性、エラー発生時のリカバリー処理の検証が行われた。競合検出や依存関係の管理機能は設計上意識されており、干渉を抑えるための運用支援が実装された。また、ハイパーバイザのメモリ管理機能が強化され、ページテーブルのキャッシュ効率向上、拡張モジュールのロード・アンロード時の負荷低減、モジュール切り替え時の遅延抑制が行われ、仮想環境内でのモジュール動作の安定性が大幅に向上した。さらに、仮想化モジュールの管理には、モジュールの導入履歴、バージョン変更履歴、エラー発生時のログ情報を記録・出力する仕組みが強化され、モジュールの運用管理の透明性が向上した。CLI を中心としたインタフェース設計によりユーザビリティが意識され、ユーザはモジュールの更新や変更履歴、競合状況の確認を迅速に行うことができるようになった。

最終的に、RISC-V の拡張機能を仮想化できるハイパーバイザは、拡張機能の柔軟な導入、競合管理の柔軟な設計、セキュリティ強化、ユーザビリティの向上という多くの課題を克服し、ハードウェア実装に依存しない柔軟な拡張環境を提供するシステムとして完成した。本プロジェクトで開発されたハイパーバイザは、既存の拡張を手軽に導入できるだけでなく、新しい拡張仕様の策定やテスト、ユーザフィードバックの収集を通じて RISC-V エコシステム全体の発展に貢献できるシステムである。

10. プロジェクト評価

本プロジェクトは、RISC-V の拡張機能をソフトウェアモジュールとして仮想化し、ハードウェアの実装に依存せず、柔軟に拡張機能を導入・運用できる環境の提供を実現した点で非常に高く評価できる。

本プロジェクトでは、RISC-V の ISA 拡張および Non-ISA 拡張を分類・整理し、それぞれに対応する仮想化モジュールの設計・開発が行われた。モジュール設計の工夫により、複数の拡張を同時に導入する際の構成管理が簡便であり、将来的な競合管理機構の拡張にも備えた構造になっている。

また、Sail 言語を活用した自動生成ツールの導入により、拡張モジュールの迅速な開発と導入が可能になった点も大きな成果である。さらに、CLI ベースで拡張モジュールの管理が可能となっており、開発者だけでなく非エンジニアのユーザも直感的に拡張機能の導入・管理ができる環境が整えられ、将来的には GUI 対応の拡張も視野に入る。仮想環境での動作検証、競合時の自動切り替え、エラー発生時のリカバリー処理など、各フェーズで丁寧な検証と最適化が行われ、着実に開発目標を達成した点も評価できる。

11. 今後の課題

今後の課題として、まず拡張モジュールの多様化と新しい拡張仕様への迅速な対応が求められている。現時点では命令および CSR のエミュレーションを含む、基本的な拡張モジュールの仮想化は実現しているが、今後はさらに多くの拡張機能を柔軟に仮想環境で活用できるようにする必要がある。また、拡張モジュールのバージョン管理や更新の効率化も課題であり、自動更新機能の導入や依存関係の自動解決などの仕組みも、今後の拡張性を高めるうえで重要な検討課題である。

さらに、拡張モジュールのセキュリティ強化も重要であり、悪意のあるコードの混入防止や異常動作の早期検出の精度向上が必要である。特に、複数の拡張機能が同時に動作する際の安全性確保には、より高度な検証・隔離機能が求められる。また、拡張モジュール間の相互運用性の向上も課題であり、モジュールの競合解決やリソース共有の最適化によって、より複雑な仮想環境でも安定した運用を実現する必要がある。

最後に、拡張モジュールの開発者コミュニティの活性化と RISC-V エコシステムへの貢献も重要である。新しい拡張機能の開発、共有、検証、普及を促進することで、RISC-V の仮想化環境の利活用がさらに進み、エコシステム全体の発展に寄与することが期待されている。これらの課題に対応することで、RISC-V の拡張仮想化ハイパーバイザは、より幅広い用途での利用が可能となり、今後の技術革新の基盤となることが期待される。