RISC-V PMP を用いた 静的パーティショニング型 VM の隔離

伊藤 大輝[1]　光来 健一[1]

概要：近年，組み込みシステムにおいて仮想化技術の利用が広がっており，静的パーティショニング型ハイパーバイザを用いて複数の仮想マシン（VM）を単一のハードウェア上で動作させている．しかし，ハイパーバイザが侵害された場合，VM のメモリが攻撃者に読み取られ，機密情報が漏洩する危険性がある．そこで，RISC-V プロセッサでも機密 VM を用いて VM のメモリ分離を行うことができるようになっているが，VM への排他的なメモリ割り当てを保証しつつ，メモリ共有を行えるようにするのは難しい．本稿では，RISC-V の PMP を用いて静的パーティショニング型 VM をハイパーバイザから保護する PRICEE を提案する．PRICEE は静的パーティショニング型 VM のメモリ領域が連続であることを利用して，メモリアクセス権限を動的に設定する．PMP はシンプルな機構であるため，VM 間の排他的なメモリ割り当てを保証することができ，メモリ共有も可能である．さらに，VM のレジスタも保護するために，PRICEE は必要最小限のレジスタのみをハイパーバイザに公開する．PRICEE を OpenSBI と Bao ハイパーバイザに実装し，QEMU 上で動作確認と性能測定を行った．

1. はじめに

近年，自動車や産業制御などの組み込みシステムにおいて仮想化技術の利用が広がっている．仮想化技術を用いることで，安全性や信頼性の異なる複数のシステムを単一のハードウェア上で統合する Mixed Criticality System（MCS）を実現できる．MCS では静的パーティショニング型ハイパーバイザが用いられることが多く，各仮想マシン（VM）に固定の CPU コアやメモリ領域を割り当てることで，システム間の干渉を防いでいる．しかし，ハイパーバイザが脆弱性などにより侵害された場合，VM のメモリやレジスタが攻撃者に読み取られ，VM 内で動作する AI モデルや制御ロジックなどの機密情報が漏洩する危険性がある．

このようなハイパーバイザからの攻撃を防ぐために，VM のメモリやレジスタを保護することができる機密 VM が用いられている．クラウドでは AMD SEV や Intel TDX などがよく用いられているが，組み込み向けに用いられる RISC-V プロセッサにおいても CoVE [9] と呼ばれる機密 VM 拡張が策定されている．CoVE では，信頼できるソフトウェアが G ステージページテーブルを用いて機密 VM に物理メモリを排他的に割り当てるが，設定ミスにより VM 間で意図しないメモリ共有が行われてしまう可能性がある．一方，排他的なメモリ割り当てを保証しようとすると，VM 間でメモリ共有を行うのが難しくなる．

本稿では，RISC-V の Physical Memory Protection（PMP）を用いて静的パーティショニング型 VM をハイパーバイザから保護する PRICEE を提案する．PRICEE は VM とハイパーバイザ間のコンテキスト切り替え時に PMP の設定を動的に変更し，VM のメモリ領域に対するアクセス制御を行う．PMP は物理メモリ領域ごとにアクセス権限を設定するシンプルな機構であるため，VM 間の排他的なメモリ割り当てを保証することができ，メモリ共有も可能である．PMP に設定できるメモリ領域数には上限があるが，連続した物理メモリ領域が割り当てられる静的パーティショニング型 VM とは相性がよい．

PRICEE を OpenSBI と静的パーティショニング型ハイパーバイザである Bao [6] に実装した．最も高い権限である M モードでセキュリティモニタを動作させ，VM Exit と VM Entry をトラップして PMP の設定の切り替えを行う．さらに，VM のレジスタを保護するために，VM Exit 時にレジスタを退避し，VM Exit の処理に必要な最小限のレジスタのみをハイパーバイザに公開する．PRICEE を QEMU 上で動作させ，ハイパーバイザから VM のメモリへの不正アクセスを防止できることを確認した．また，PRICEE を用いることによる VM の性能への影響を

[1] 九州工業大学
Kyushu Institute of Technology

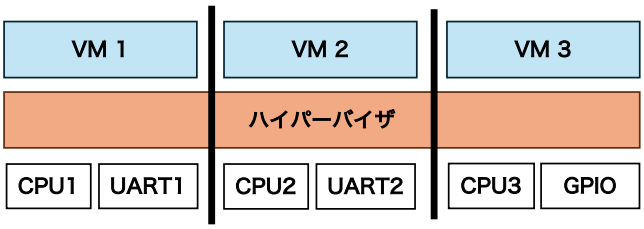


図 1: 静的パーティショニング型 VM の構成

測定した結果，ネットワーク性能が 32%低下することが分かった．

以下，2 章では RISC-V における機密 VM の問題点について述べる．3 章では PMP を用いて VM のメモリ分離を行う PRICEE を提案する．4 章では PRICEE の実装について説明し，5 章では動作確認と性能測定のために行った実験について述べる．6 章で関連研究に触れ，7 章で本稿をまとめる．

2. RISC-V の機密 VM

MCS においては静的パーティショニング型ハイパーバイザ [6], [8] が用いられることが多い．このタイプのハイパーバイザは**図 1** に示すように，リソースを静的に分割し，各 VM に排他的に割り当てる．CPU コア，メモリ領域，I/O デバイスなどのハードウェアリソースは，VM 起動時に固定的に割り当てられ，実行中に変更されることはない．仮想 CPU は単一の物理 CPU に固定的にマッピングされるため，CPU スケジューラが不要となる．また，リソースの動的な共有を行わないため，VM 間の干渉が最小限に抑えられ，リアルタイム性と決定論的な動作が保証しやすくなる．

このような仮想化を用いた MCS をインターネットに接続すると，すべての VM が外部からの攻撃にさらされる．攻撃者はまず，最も脆弱な VM に侵入し，次に，ハイパーバイザの脆弱性を利用してハイパーバイザを侵害することで，他のすべての VM にアクセスできるようになる．ハイパーバイザはすべての VM のメモリとレジスタにアクセスする権限を持っているため，ハイパーバイザが侵害されると，VM の機密情報が漏洩する危険性がある．また，VM のメモリやレジスタを書き換えることで，VM の動作を不正に変更することも可能である．

こうした脅威に対処する手段として，ハードウェアの機能を用いてメモリやレジスタを保護する機密 VM が用いられている．機密 VM は CPU が提供する VM 向けの高信頼実行環境（TEE）を用いて作成される．サーバ向けの TEE としては，AMD SEV [1] や Intel TDX [3]，Arm CCA [2] などがある．機密 VM を用いることにより，ハイパーバイザが侵害された場合であっても VM のデータやコードの機密性と完全性が維持される．組み込み向けに用いられる RISC-V プロセッサにおいても，CoVE [9] と

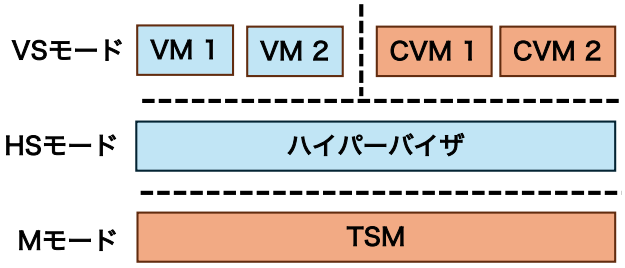


図 2: CoVE（モデル 3）のシステム構成

呼ばれる機密 VM の仕様が策定されている．CoVE では**図 2** のように，TEE セキュリティマネージャ（TSM）と呼ばれる信頼できるソフトウェアが機密 VM のメモリ分離を強制する．

CoVE では，TSM をどの特権レベルで動作させ，どのハードウェア機能を用いてメモリ分離を行うかに応じて，3 つのデプロイメントモデルが定義されている．モデル 1 では，TSM とハイパーバイザの両方を HS モードと呼ばれる CPU の特権レベルに配置する．そして，メモリトラッキングテーブル（MTT）を用いて物理メモリを機密メモリと非機密メモリに分離する．さらに，TSM がそれぞれの機密 VM のメモリを G ステージページテーブルを用いて分離する．ハイパーバイザが機密 VM に割り当てられた機密メモリにアクセスしようとした場合には，MTT によってアクセスが拒否される．ただし，MTT を利用するには RISC-V の Smmtt 拡張が必要であり，この拡張を備えた RISC-V プロセッサはまだ存在しない．

モデル 2 では，TSM のみを HS モードに配置し，ハイパーバイザはその上で動作する非機密 VM 内で動作させる．モデル 1 と同様に，TSM は G ステージページテーブルを用いて機密 VM のメモリを分離する．ハイパーバイザは VM 内に隔離されるため，機密 VM のメモリに直接アクセスする手段を持たない．このモデルはネストした仮想化を用いて実現されるため，追加のハードウェア拡張を必要とせず，ハイパーバイザ拡張に対応した既存の RISC-V プロセッサで実現できる．ただし，ネストした仮想化のオーバヘッドが大きい．

モデル 3 では，図 2 のように TSM を最も高い権限である M モードに配置し，ハイパーバイザを HS モードで動作させる．そして，物理メモリを機密メモリと非機密メモリに静的に分割し，PMP と呼ばれる標準ハードウェアを用いて機密メモリをアクセス制御により保護する．他のモデルと同様に，機密 VM のメモリは G ステージページテーブルを用いて分離する．ハイパーバイザの実行中は PMP のアクセス制限により，機密 VM のメモリにアクセスすることはできない．メモリを静的に分割する必要があるため，機密 VM へのメモリ割り当ての柔軟性は低下するが，静的パーティショニング型ハイパーバイザとは相性がよい．

CoVE において TSM は信頼できるソフトウェアである

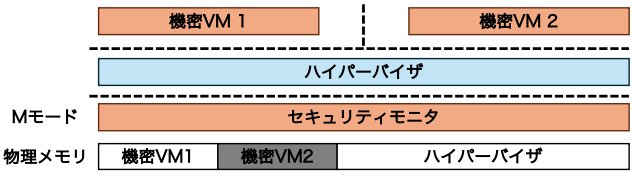


図 3: PRICEE のシステム構成

が，TSM が G ステージページテーブルを用いて機密 VM に物理メモリを排他的に割り当てることを保証するのは容易ではない．G ステージページテーブルは多段の階層構造と多数の制御ビットを持つため，設定ミスにより機密 VM 間で意図しないメモリ共有が行われてしまう可能性がある．この問題を解決するために，モデル 3 の一実装である ACE [7] では，Rust 言語の所有権を用いて物理ページを排他的にしか G ステージページテーブルに登録できないようにしている．しかし，この機構を用いると，物理ページを機密 VM 間で共有することはできなくなる．

3. PRICEE

本稿では，RISC-V の PMP のみを用いて静的パーティショニング型 VM を保護する PRICEE [*1]を提案する．PRICEE は PMP を用いる点で CoVE のモデル 3 に似ているが，機密メモリ全体ではなく，それぞれの VM のメモリ領域に個別に PMP の設定を行う．PMP を用いて設定可能な物理メモリ領域の数は限られるため，この手法を一般的な VM に適用するのは難しいが，連続した物理メモリ領域が割り当てられる静的パーティショニング型 VM には適用可能である．これにより，G ステージページテーブルによって VM のメモリ分離を保証する必要はなくなる．PMP は物理メモリ領域ごとにアクセス権限を設定するだけであるため，ページ単位で設定を行う G ステージページテーブルよりはるかに単純である．そのため，VM に対する排他的なメモリ割り当てやメモリ共有の設定ミスは生じにくい．

図 3 に PRICEE のシステム構成を示す．CoVE のモデル 3 と同様に，M モードでセキュリティモニタを動作させ，PMP の設定を行う．M モード以外の特権レベルでは PMP の設定を変更できないため，HS モードで動作するハイパーバイザが PMP の設定を書き換えて VM のメモリにアクセスすることはできない．CoVE とは異なり，G ステージページテーブルはハイパーバイザによって管理されるが，PMP による VM 単位の保護により，VM に対して意図しないメモリ共有を行うことはできない．

PRICEE の脅威モデルは以下の通りである．ハードウェアおよびセキュリティモニタは信頼できるものとし，脆弱性はないものとする．攻撃として，外部の攻撃者によって

[*1] PMP-based RISC-V Isolated Confidential Execution Environment

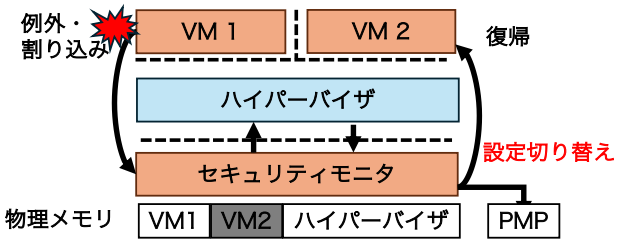


図 4: コンテキストに応じた PMP 切り替え

侵害された VM を経由したハイパーバイザの侵害を想定する．ハイパーバイザは VM のメモリやレジスタにアクセスすることで盗聴や改ざんを行ったり，G ステージページテーブルを書き換えることで VM 間の隔離を無効化したりしようとするものとする．

PRICEE は実行中のコンテキストに応じて PMP の設定を動的に切り替える．ハイパーバイザの実行中には，ハイパーバイザのメモリ領域へのアクセスのみを許可し，すべての VM のメモリ領域へのアクセスを禁止する．これにより，侵害されたハイパーバイザによる VM のメモリに対する攻撃を防ぐことができる．VM の実行中には，その VM のメモリ領域へのアクセスのみを許可し，他の VM のメモリ領域へのアクセスを禁止する．これにより，侵害された VM による他の VM のメモリに対する攻撃を防ぐことができる．一方，VM の実行中に，他の VM のメモリ領域の一部へのアクセスを許可することで，VM 間でのみ安全にメモリを共有することができる．これにより，ハイパーバイザによる VM 間の共有メモリに対する攻撃を防ぐことができる．

このような PMP の切り替えは**図 4** に示すように，VM Exit と VM Entry の際にセキュリティモニタが行う．VM の実行中に例外や割り込みによって VM Exit が発生すると，ハイパーバイザに遷移する前にセキュリティモニタが VM Exit をトラップする．セキュリティモニタは PMP の設定を変更した後で，ハイパーバイザに VM Exit をリダイレクトする．また，ハイパーバイザが VM Entry を実行した時にも，VM に遷移する前にセキュリティモニタが VM Entry をトラップする．セキュリティモニタは PMP の設定を変更した後で，VM Entry をエミュレートして VM に制御を移す．ただし，VM の起動時にはハイパーバイザが OS 等の初期イメージを VM のメモリに書き込めるようにするために，最初の VM Entry が実行されるまでは VM のメモリへのアクセス制限は行わない．

VM のレジスタを保護するために，VM Exit 発生時にはセキュリティモニタが VM のレジスタをコンテキスト領域に退避する．コンテキスト領域はセキュリティモニタのメモリ上に置かれ，ハイパーバイザからはアクセスできない．ハイパーバイザが VM Exit の処理のために読み込む必要があるレジスタについては，VM Exit の要因に応じ

て最小限のレジスタのみをハイパーバイザに公開する．これにより，VM の内部状態の漏洩を抑えることができる．一方，ハイパーバイザが書き換えたレジスタについては，VM Entry 時にセキュリティモニタがコンテキスト領域に反映する．その際に，VM Exit の要因に応じて書き換えが必要なレジスタの値のみを書き戻す．その後で，コンテキスト領域のレジスタの値を用いて VM のレジスタを復元する．これにより，VM の制御フローやスタック領域の変更などを防ぐことができる．

4. 実装

PRICEE のセキュリティモニタを OpenSBI を拡張して実装した．静的パーティショニング型ハイパーバイザには Bao [6] を用いた．以下の実装において参照する RISC-V の特権命令および制御レジスタは，RISC-V 特権アーキテクチャ仕様 [10] に従う．

4.1 VM のメモリ分離

PMP を用いて VM のメモリを保護するために，M モードでのみアクセス可能な PMP 設定レジスタ（pmpcfg）と PMP アドレスレジスタ（pmpaddr）を用いて物理メモリ領域ごとにアクセス権限を設定する．PMP のエントリ数は最大で 64 であるが，実際にはもっと少ないことが多い．しかし，静的パーティショニング型 VM の場合，1 つの VM につき 1 つのエントリで済み，VM 数もそれほど多くはないのが一般的である．PMP のアドレスマッチング方式として TOR を使用し，一つ前のエントリに設定した物理アドレスから対象のエントリに設定した物理アドレスまでのメモリ領域を保護対象とする．そのため，設定する物理メモリ領域に重複が生じることはなく，VM 間や VM とハイパーバイザ間で意図しないメモリ共有が行われないことが保証される．

PRICEE は Bao ハイパーバイザの設定ファイルの情報を基に，PMP の設定を行うセキュリティモニタのコードを自動生成する．Bao ハイパーバイザでは，各 VM に割り当てられる物理メモリ領域のアドレスとサイズは設定ファイルで静的に定義される．設定ファイルからコードを自動生成することにより，定義と同じ物理メモリ領域に PMP の設定が行われることが保証できる．ハイパーバイザが実行時に設定ファイルと異なる物理メモリ領域を VM に割り当てたとしても，VM を正常に動作させることはできないことが多く，VM のメモリに対して盗聴や改ざんを行うこともできない．

ハイパーバイザ実行中は，ハイパーバイザのメモリ領域にだけ読み書き実行（RWX）権限が付与され，どの VM のメモリ領域にもアクセス権限が付与されないように PMP の設定を行う．VM 実行中は，その VM のメモリ領域に RWX 権限が付与され，他の VM のメモリ領域にはアクセス権限が付与されないように PMP の設定を行う．なお，VM 実行中であっても，CPU が 2 段階アドレス変換を行う際に，ハイパーバイザのメモリ領域にアクセスして G ステージページテーブルをたどる．そのため，ハイパーバイザのメモリ領域にも読み取り権限のみが付与されるように PMP の設定を行う．

4.2 VM 間の安全なメモリ共有

VM 間の共有メモリ領域も PMP を用いて保護することにより，安全かつ柔軟なメモリ共有を行うことができる．共有メモリ領域は VM の物理メモリ領域と同様に，Bao ハイパーバイザの設定ファイルで静的に定義される．PRICEE では，この設定ファイルにおいて共有メモリ領域ごとにアクセスを許可するコンテキストを VM やハイパーバイザを列挙することで記述することができる．さらに，コンテキストごとに読み書き実行権限を記述することができる．例えば，ある共有メモリ領域を VM 1 には読み取り専用，VM 2 には読み書き可能，ハイパーバイザにはアクセス不可になるように設定できる．ただし，それぞれの共有メモリ領域に 1 つの PMP エントリを割り当てるため，PMP エントリ数の制約により共有メモリ領域の数には限りがある．

VM 実行中は，その VM にアクセスが許可されている共有メモリ領域に対してのみ，設定ファイルで定義されたアクセス権限が付与されるように PMP の設定を行う．その VM にアクセスが許可されていない共有メモリ領域に対しては，アクセス権限を付与しない．ハイパーバイザ実行中は，ハイパーバイザにアクセスが許可されている共有メモリ領域に対してのみ，設定ファイルで定義されたアクセス権限が付与されるように PMP の設定を行う．これにより，ハイパーバイザからアクセスできない VM 間専用の共有メモリも実現でき，ハイパーバイザが侵害された場合でも VM 間通信の内容が保護される．

4.3 VM Exit のトラップ

セキュリティモニタが VM Exit をトラップできるようにするために，VS モードで発生するすべての例外と割り込みを M モードのセキュリティモニタで受け取る．通常は，マシン例外委譲レジスタ（medeleg）とマシン割り込み委譲レジスタ（mideleg）の対応するビットを 1 にすることより，すべての例外と一部を除くすべての割り込みを HS モードのハイパーバイザに委譲している．そのため，例外と割り込みは直接ハイパーバイザがトラップする．PRICEE では，これらのレジスタの対応するビットをクリアすることにより，ハイパーバイザへの委譲を行わないようにする．発生する例外としては，VM がハイパーバイザを呼び出すために用いる SBI コール，ページフォールト，不正命令例外などがある．割り込みとしては，タイマ割り

込み，外部割り込み，ソフトウェア割り込みがある．

セキュリティモニタは VM Exit の処理に必要な情報をハイパーバイザとの間の共有メモリに格納する．例えば，VM が MMIO 領域にアクセスすることによってページフォールトが発生した際には，セキュリティモニタはマシントラップ命令レジスタ（mtinst）から対象となる命令の情報を取得し，共有メモリに格納する．例外を委譲している場合，ハイパーバイザはハイパーバイザトラップ命令レジスタ（htinst）から命令情報を取得することができるが，委譲しない場合，このレジスタには情報が格納されない．セキュリティモニタであっても htinst レジスタには書き込むことができないため，共有メモリ経由でハイパーバイザに情報を受け渡す．

その後，PMP の設定をハイパーバイザ用に切り替え，VM Exit をハイパーバイザにリダイレクトする．まず，マシン例外プログラムカウンタレジスタ（mepc）にハイパーバイザの例外ハンドラのアドレスを設定し，マシンステータスレジスタ（mstatus）に遷移先の動作モードとして HS モードを設定する．次に，VM Exit の要因や関連情報をハイパーバイザが参照できる HS モード用のレジスタに設定する．最後に，M モードから下位の特権レベルに戻る MRET 命令を実行すると，HS モードに遷移してハイパーバイザの例外ハンドラが実行される．

割り込みについては，ハイパーバイザ実行中には従来通り，委譲を行うようにする．ハイパーバイザ実行中に割り込みをトラップし，VM Exit をハイパーバイザにリダイレクトすると，割り込みが正常に処理できず，M モードへのトラップが無限に発生してしまうためである．そこで，VM Exit の際に割り込みの委譲を有効化し，VM Entry の際に再び委譲を無効化する．ハイパーバイザ実行中に発生した割り込みはハイパーバイザが処理するため，PMP の設定を切り替える必要はない．そのため，割り込みの委譲を行っても問題はない．

4.4 VM Entry のトラップ

セキュリティモニタが VM Entry をトラップできるようにするために，M モード用の mstatus レジスタに設定を行う．これにより，ハイパーバイザまたは OS が下位の特権レベルに戻るために SRET 命令を実行した際に，M モードに不正命令例外が発生する．この例外をトラップしたセキュリティモニタは，ハイパーバイザが VM に制御を戻すために実行した SRET 命令かどうかを調べる．まず，mstatus レジスタを参照して，ハイパーバイザが動作する HS モードから M モードへの遷移であることを確認する．さらに，HS モード用のハイパーバイザステータスレジスタ（hstatus）を参照して，ハイパーバイザへの遷移が VM Exit によって VM から行われたことを確認する．

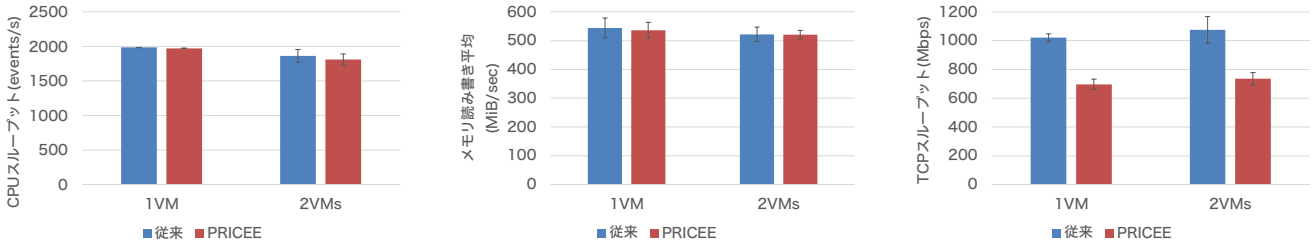
M モードから VM Entry を行うために，セキュリティモニタが SRET 命令をエミュレートする．そのために，hstatus レジスタと VS モード用のスーパーバイザステータスレジスタ（sstatus）の状態を mstatus レジスタに反映させ，hstatus レジスタと sstatus レジスタの状態を更新する．また，mepc レジスタに VM の実行を再開するアドレスを設定する．VM Entry を行う VM はハイパーバイザゲストアドレス変換および保護レジスタ（hgatp）に格納されている VM 識別番号（VMID）から特定する．最後に，PMP の設定をその VM 用に切り替え，MRET 命令を実行して VM に直接，制御を移す．

4.5 VM のレジスタ保護

VM のレジスタをハイパーバイザから保護するために，セキュリティモニタは VM Exit 時に VM のすべての汎用レジスタを自身のメモリ上に確保したコンテキスト領域に退避する．そして，VM Entry 時にコンテキスト領域から VM のレジスタを復元する．それに伴い，ハイパーバイザが従来，VM Exit や VM Entry の際に行っていたレジスタの退避と復元は行わないようにする．レジスタの退避後，セキュリティモニタはレジスタをクリアするため，ハイパーバイザは VM のレジスタを参照することができなくなる．VM Exit の処理に必要なレジスタの値はセキュリティモニタが共有メモリに格納し，ハイパーバイザは共有メモリ経由で参照する．逆に，ハイパーバイザが書き換えたレジスタの値は共有メモリに格納し，セキュリティモニタが共有メモリ経由で取得する．

VM Exit 時に，セキュリティモニタはホワイトリストに基づいて，レジスタを共有メモリ経由でハイパーバイザに公開する．その際に用いるホワイトリストは VM Exit の要因によって決定する．例えば，SBI コールの際にはハイパーバイザが呼び出し番号と引数を取得する必要があるため，これらを格納するレジスタ群のみを公開する．MMIO 領域への書き込みに伴うページフォールトでは，mtinst レジスタに格納されている命令情報から特定した書き込み元のレジスタのみを公開する．一方，MMIO 領域の読み取り時のページフォールトや割り込みなど，ハイパーバイザがレジスタを参照せずに処理を行える場合にはレジスタの公開は行わない．

VM Entry 時に，セキュリティモニタはホワイトリストに基づいて，ハイパーバイザが書き換えたレジスタの値をコンテキスト領域に反映する．このホワイトリストは直近の VM Exit の要因によって決定する．例えば，SBI コールの後には，返り値が格納されたレジスタの値のみを反映する．MMIO 領域の読み取りをエミュレートした後は，読み込み先のレジスタの値のみを反映する．ただし，VM の起動時に初めて発生する VM Entry については，ハイパーバイザがすべてのレジスタに初期値を設定する必要があるため，ホワイトリストによる制限は行わない．



(a) CPU 性能　(b) メモリ性能　(c) ネットワーク性能

図 5: PRICEE の VM の性能への影響

5. 実験

PRICEE がハイパーバイザから VM のメモリへの不正アクセスを防止できることを確認する実験を行った．また，PRICEE がシステム性能に与える影響を測定した．比較として，機密 VM をサポートしない従来システムにおいても測定を行った．この実験は，QEMU 8.1.2 を用いて RISC-V プロセッサをエミュレーションして行った．ホストの CPU は Intel Core i7-14700，メモリは 32GB である．エミュレーション環境には CPU を 4 個，メモリを 4GB 割り当て，ファームウェアとして OpenSBI 1.7，ハイパーバイザとして Bao 1.0.0 を動作させた．その上で動作する VM には仮想 CPU を 2 個，メモリを 2 GB 割り当て，ゲスト OS として Linux 6.1.0 を動作させた．

5.1 ハイパーバイザによる不正メモリアクセス

Bao ハイパーバイザを改変し，トラップハンドラにおいて VM のメモリにアクセスするようにした．従来システムではこのメモリアクセスは成功したが，PRICEE ではアクセス違反の例外が発生し，不正アクセスが防止されることを確認した．

5.2 VM 性能への影響

VM 内で sysbench を実行して CPU 性能の測定を行った．1 つまたは 2 つの VM からなるシステム構成で測定を行い，2 VM 構成の場合には同時にベンチマークを実行してそれぞれのベンチマーク結果の平均を用いた．10 回測定した時の平均と標準偏差を**図 5(a)** に示す．従来システムに対して，PRICEE のオーバヘッドは 1 VM 構成で 0.65%，2 VM 構成で 2.8% と小さかった．これは実行したのが整数演算を中心としたベンチマークであり，VM Exit や VM Entry がほとんど発生しなかったためである．

同様に，VM 内で sysbench を用いてメモリ性能を測定した．測定結果は**図 5(b)** のようになり，この場合も PRICEE のオーバヘッドは 1 VM 構成で 1.4%，2 VM 構成で 0.26% と小さかった．2 VM 構成では VM 間でメモリ帯域を共有するため，PRICEE と従来システムのいずれにおいても 1 VM 構成より若干低い値となった．

表 1: 測定対象の例外処理

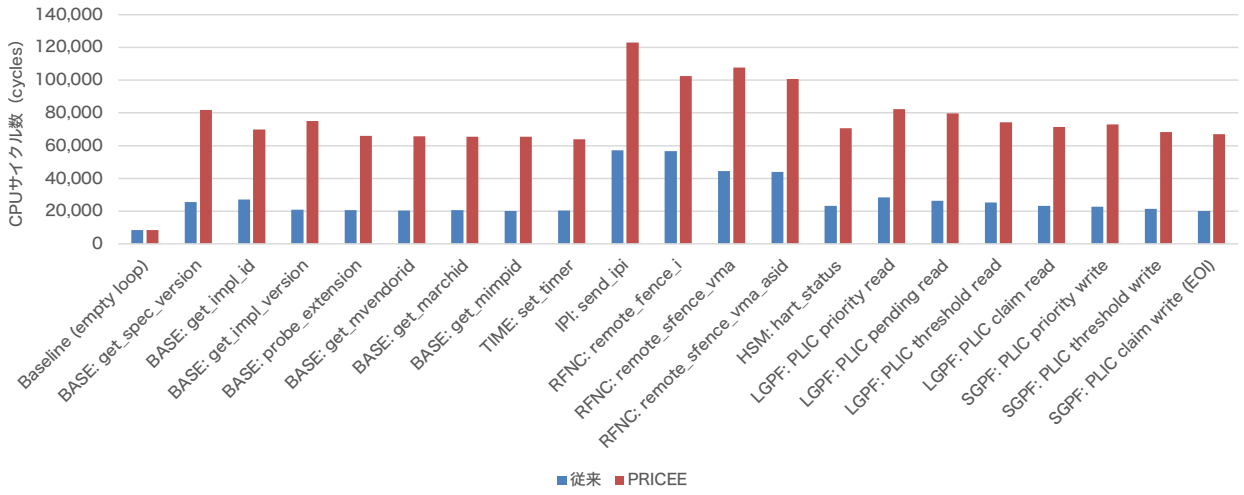
SBI コール	
BASE 拡張（7 種類）	SBI 仕様および実装情報の取得
TIME: set_timer	S モードタイマ割り込みの設定
IPI: send_ipi	CPU 間割り込みの送信
RFNC 拡張（3 種類）	同期命令・TLB フラッシュ
HSM: hart_status	CPU の状態取得
MMIO アクセス	
LGPF（4 種類）	PLIC レジスタの読み込み
SGPF（3 種類）	PLIC レジスタの書き込み

次に，VM 内で iperf3 のサーバを実行してネットワーク性能を測定した．iperf3 のクライアントはホスト上で実行した．測定結果は**図 5(c)** のようになり，1 VM 構成でも 2 VM 構成でも 32% と大きなオーバヘッドが見られた．ネットワーク I/O ではパケットの送受信のたびに MMIO アクセスが行われ，ページフォールトによる VM Exit が頻繁に発生する．PRICEE では，セキュリティモニタが VM Exit をトラップし，PMP の設定変更とレジスタの公開を行い，ハイパーバイザに VM Exit をリダイレクトする．さらに，VM Entry もトラップし，PMP の設定変更と更新されたレジスタの反映を行ってから VM Entry をエミュレートする．これらの追加処理が大きなオーバヘッドの原因である．

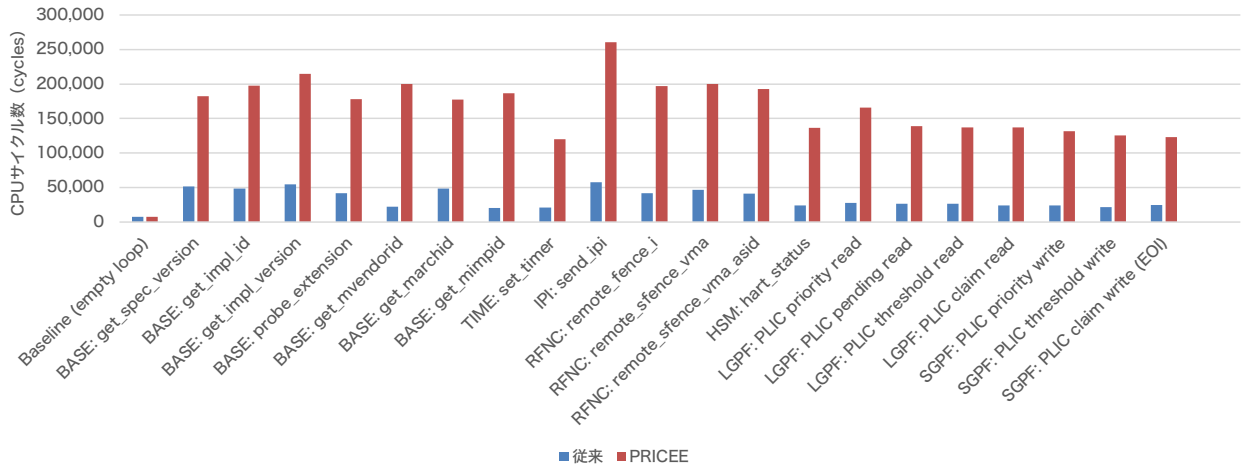
5.3 例外処理のオーバヘッド

例外処理における PRICEE のオーバヘッドを調べるために，VM 内で**表 1** に示す例外を発生させ，例外処理 1 回あたりの CPU サイクル数を測定した．測定には RISC-V プロセッサのサイクルカウンタを使用した．測定対象は SBI コールと，仮想 PLIC への MMIO アクセスに伴うページフォールトである．

測定結果を**図 6** に示す．1 VM 構成では従来システムの 1.8–3.6 倍の CPU サイクル数がかかることが分かった．SBI コールで約 1.8–3.6 倍，MMIO アクセスで約 2.9–3.3 倍となり，MMIO アクセスの方がオーバヘッドが大きい傾向にあった．これは PRICEE では例外発生時にセキュリ



(a) 1 VM 構成



(b) 2 VM 構成

図 6: 例外処理にかかる CPU サイクル数

ティモニタにおいて様々な処理を行うためである．2 VM 構成ではオーバヘッドがさらに増加し，3.5–9.1 倍の CPU サイクル数がかかった．SBI コールで約 3.5–9.1 倍，MMIO アクセスで約 5.0–6.0 倍となった．これは VM 間のコンテキストスイッチが発生し，PMP 設定の切り替えとコンテキスト領域の切り替えが追加で必要となるためである．

6. 関連研究

ACE [7] は M モードで動作する TSM を用いてハイパーバイザと機密 VM を分離し，CoVE 仕様に準拠しつつ形式検証を考慮した設計となっている．PMP によって機密 VM 全体を非機密側から分離した上で，機密 VM 同士の隔離は TSM が管理する G ステージページテーブルを用いて実現する．Rust の型システム上で各物理ページを単一の機密 VM にしか割り当てられないようにする Page Token 機構を用いることにより，G ステージページテーブルの排他割り当ての設定ミスを言語レベルで防ぐことができる．一方で，ACE は VM 間でメモリを共有することを想定しておらず，メモリ共有を実現するには Page Token 機構を回避する必要があるため，新たな脆弱性が入り込む余地が生じる．PRICEE は PMP のシンプルな設定のみで VM 間の排他的なメモリ割り当てとメモリ共有の双方を扱うことができる．

ACE は CoVE 仕様に従うため，ゲスト OS，ハイパーバイザ，TSM 間のやりとりの多くは SBI コールを介して行われる．このため，ゲスト OS に CoVE 用のドライバや初期化処理を組み込み，TSM に対して明示的に SBI コールを発行する必要がある．加えて，ハイパーバイザにも CoVE が定める SBI コールを実装しなければならず，ゲスト OS とハイパーバイザに対する変更が必要となる．これに対して，PRICEE は VM とハイパーバイザ間のやりとりをセキュリティモニタが透過的にトラップするため，ゲスト OS には一切の変更を加える必要がなく，ハイパーバイザに対しても共有メモリ経由でのレジスタ参照などの

最小限の改変だけで済む．

Keystone [4] は，RISC-V 向けの TEE フレームワークであり，M モードで動作するセキュリティモニタが PMP を用いてエンクレイヴのメモリを保護する．エンクレイヴ内ではランタイムと呼ばれる軽量な OS が動作するため，エンクレイヴは軽量な VM とみなすこともできるが，I/O はサポートされておらず，既存のアプリケーションを実行することもできない．Elasticlave [11] は PMP を用いてエンクレイヴ間で柔軟にメモリを共有することを可能にしている．この手法は PRICEE におけるメモリ共有に似ている．

AMD SEV-SNP [1] は，VM ごとに固有の鍵を用いてメモリを暗号化し，Reverse Map Table（RMP）を用いてメモリの整合性を保護する．RMP には物理ページごとに所有者と対応するゲスト物理アドレスを記録し，VM がその対応関係を認可する仕組みを用いることで，悪意あるハイパーバイザによる再マッピングや偽装を防ぐ．ネストしたページテーブル（NPT）はハイパーバイザが管理するが，メモリアクセス時に毎回 RMP チェックが行われるため，ハイパーバイザによる NPT の改ざんを検知できる．一方，RMP のために VM 間で暗号化されたメモリを安全に共有することはできない．これに対し PRICEE は，メモリにはその所有者のみがアクセスできることを保証し，VM 間の安全なメモリ共有も可能にしている．

Intel TDX [3] は Trust Domain（TD）と呼ばれる機密 VM を，Secure Arbitration Mode（SEAM）と呼ばれる新たな CPU モードで動作する信頼できる TDX モジュールを介して管理する．メモリは TD ごとに異なる鍵で暗号化され，Physical-Address-Metadata Table（PAMT）によって各物理ページが 1 つの TD にしか割り当てられないことを保証する．VM のメモリを管理する EPT は機密メモリ用の Secure EPT と共有メモリ用の Shared EPT に分離され，Secure EPT の編集は TDX モジュールが仲介することで，VMM による改ざんを防いでいる．PRICEE では EPT に対応する G ステージページテーブルにはハイパーバイザがアクセスでき，機密メモリの安全な共有も可能である．

TwinVisor [5] は Armv8.4 で導入された S-EL2 を利用して機密 VM を隔離する．セキュアワールドの S-EL2 に S-visor と呼ばれるハイパーバイザを配置し，ノーマルワールドのハイパーバイザと連携して機密 VM を実行する．セキュアワールドに割り当てるメモリ量はノーマルワールドと協調して動的に調整され，セキュアワールドにおいて機密 VM のメモリはセキュアステージ 2 ページテーブルを用いて分離される．これは RISC-V の G ステージページテーブルと同様であり，柔軟である反面，排他的な割り当てを保証するのが難しい．

Arm CCA [2] は Armv9-A で導入された Realm Management Extension （RME）を用いて，Realm ワールドと呼ばれる物理アドレス空間を新設している．物理メモリの所属ワールドは Granule Protection Table（GPT）によって強制される．Realm VM のメモリを管理する Realm Translation Table（RTT）は Realm Management Monitor（RMM）が管理し，ハイパーバイザからの要求を検証してから操作される．これにより，ハイパーバイザが侵害されても Realm VM のメモリにアクセスできないことを保証する．RTT は RISC-V の G ステージページテーブルに対応し，同様の問題を抱えている．

7. まとめ

本稿では，RISC-V の PMP を用いて静的パーティショニング型 VM を保護する PRICEE を提案した．PRICEE は M モードで動作するセキュリティモニタを用いて，VM Exit と VM Entry の際に VM のメモリ領域のアクセス権限を動的に切り替える．また，VM Exit の要因に応じて必要最小限の VM のレジスタのみをハイパーバイザに公開する．PRICEE を OpenSBI と Bao ハイパーバイザに実装し，ハイパーバイザから VM のメモリへの不正アクセスを防止できることを確認した．また，PRICEE によりネットワーク性能が 32%低下することを確認した．

今後の課題として，ハイパーバイザによる VM 内のメモリマッピングの不正な変更を防止できるようにする必要がある．そのために，G ステージページテーブルを保護できるようにする．また，汎用レジスタに加えて CSR や浮動小数点レジスタなども選択的にハイパーバイザに公開できるようにする．さらに，DMA による不正アクセスを防ぐために IOPMP や IOMMU の利用も検討する．並行して，PRICEE を実機で動作させ，オーバヘッドを測定する予定である．

謝辞　本研究は，JST 経済安全保障重要技術育成プログラム【JPMJKP24U4】の支援を受けたものである．

参考文献

[1] Advanced Micro Devices: AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More, White paper, AMD (2020).
[2] Arm Limited: Introducing Arm Confidential Compute Architecture, Guide DEN0125, Arm (2023).
[3] Intel Corporation: Intel Trust Domain Extensions, White paper, Intel (2023).
[4] Lee, D., Kohlbrenner, D., Shinde, S., Asanović, K. and Song, D.: Keystone: An Open Framework for Architecting Trusted Execution Environments, *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20)*, pp. 1–16 (2020).
[5] Li, D., Mi, Z., Xia, Y., Zang, B., Chen, H. and Guan, H.: TwinVisor: Hardware-isolated Confidential Virtual Machines for ARM, *Proceedings of the 28th ACM Symposium on Operating Systems Principles (SOSP '21)*, pp. 638–654 (2021).

[6] Martins, J., Tavares, A., Solieri, M., Bertogna, M. and Pinto, S.: Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems, *Workshop on Next Generation Real-Time Embedded Systems (NG-RES 2020)*, pp. 3:1–3:14 (2020).
[7] Ozga, W., Hunt, G. D. H., Le, M. V., Gäher, L., Shinnar, A., Palmer, E. R., Jamjoom, H. and Dragone, S.: ACE: Confidential Computing for Embedded RISC-V Systems, *arXiv preprint arXiv:2505.12995* (2025).
[8] Ramsauer, R., Kiszka, J., Lohmann, D. and Mauerer, W.: Look Mum, No VM Exits! (Almost), *arXiv preprint arXiv:1705.06932* (2017).
[9] RISC-V International: Confidential VM Extension (CoVE) for Confidential Computing (2024). Version 0.7.
[10] RISC-V International: The RISC-V Instruction Set Manual, Volume II: Privileged Architecture (2024). Document Version 20240411.
[11] Yu, J. Z., Shinde, S., Carlson, T. E. and Saxena, P.: Elasticlave: An Efficient Memory Model for Enclaves, *Proceedings of the 31st USENIX Security Symposium*, pp. 4111–4128 (2022).