3J-1

SIVARMの設計と実装

鈴木　章浩†　　追川　修一†
†筑波大学コンピュータサイエンス専攻

1　はじめに

近年，ハードウェアの性能向上により組み込み機器において仮想マシンモニタ (VMM)[1] の利用が議論されるようになってきている．VMM を導入することで，組み込み機器の高可用性・耐故障性を容易に実現可能となる．そこで，本研究では組み込み機器用の CPU である ARM アーキテクチャのための VMM を設計し，実装する．この VMM を本研究では SIVARM (a Simple VMM for the ARM Architecture) と呼ぶ．

2　設計

VMM がゲスト OS を自身の提供した仮想マシン (VM) 上で制御するためには，ゲスト OS が実行するセンシティブ命令を検知し，適切にエミュレーションすることで VM の状態を更新しなければならない．そのため，本研究では VMM を特権モード，ゲスト OS を非特権モードで動作させることで，ゲスト OS の実行するセンシティブ命令を例外によって検知し，VMM で適切にエミュレーションする．また，VMM を特権モードに配置することによって，ゲスト OS からの不正なアクセスから VMM を保護することも同時に実現する．なお，VMM が提供する VM 環境は問題を単純化するために 1 つとする．

本研究では上記のようにセンシティブ命令を検知するためにゲスト OS を非特権モード上でのみ動作させる必要がある．そのため，本研究でゲスト OS として使用する Linux のカーネルモードでは，ARM アーキテクチャの特権モードでのみ使用することのできるバンクレジスタを使用することができない．また，カーネルプロセスはユーザモードからのアクセスをページングによる保護機構によって制御しているが，その制御は VMM の保護に使用されるため，カーネルモードをユーザモードから保護することができない．そこで本研究では非特権モード上にプロセッサモードを仮想的に構築し，仮想特権モードでカーネルプロセス，仮想非特権モードでユーザプロセスを動作させることでこの問題を解決する．

3　実装

3.1　センシティブ命令のエミュレーション

非特権モードで動作するゲスト OS は非センシティブ命令を通常通り実行するが，センシティブ命令を実行すると特権不足により未定義命令例外が発生し，プログラムカウンタが例外ベクタへ移行する．なお，センシティブ命令がデバイスへのアクセス命令の際はデータアボート例外が発生する．

例外ベクタには VMM の起動時に VMM の例外ハンドラへのブランチ命令を配置する．これによって VMM の例外ハンドラ内では高水準言語を用いてエミュレーション処理を柔軟に記述できる．但し，例外ハンドラ内で汎用スクラッチレジスタが破壊されるため [2]，例外ハンドラへの遷移前に低水準言語でそれらのレジスタを格納し，エミュレーション処理終了後に復帰する必要がある．

本研究では問題を単純化するために VMM の提供する VM 環境を 1 つと定義しているため，エミュレーション処理は VMM 内で特権を持った状態でセンシティブ命令を実行することで行う．その際，ゲスト OS がセンシティブ命令を実行したコンテキスト状態を復帰する必要がある．また，エミュレーション処理実行後は再びコンテキスト状態を退避させ，例外ハンドラから復帰する際にそれらを復元する．

非特権モードで実行すると例外を発生させる特権センシティブ命令は上記の手法によってエミュレーションが可能である．しかし，ARM アーキテクチャには非特権センシティブ命令が存在する．この命令はセンシティブ命令でありながら非特権モードで実行しても例外を発生させないため，VMM で検知することができず，VM 環境を適切に更新することができない．そのため，本研究ではこの命令を例外を発生させる別の特権命令に置換することで，その実行を検知する．本研究ではこの命令を代理特権命令と呼ぶ．VMM では代理特権命令と非特権センシティブ命令を関連付けておき，代理特権命令の実行を検知した際は非特権センシティブ命令のエミュレーション処理を実行する．エミュレーションの処理手順は特権センシティブ命令の場合と同様である．

3.2　仮想プロセッサモード

仮想プロセッサモードは仮想バンクレジスタとドメインを用いた仮想保護レベルから構成される．

3.2.1　仮想バンクレジスタ

仮想バンクレジスタは，ARM アーキテクチャにおいて特権モードでのみ利用することのできるバンクレジスタを非特権モード上の仮想特権モードにおいて仮想的に構築するものである．具体的には，VMM 内に仮想バンクレジスタとして利用する領域を用意し，ゲスト OS が自身の仮想プロセッサモードを遷移させた場合には現在のレジスタを仮想バンクレジスタの領域に格納し，遷移先の仮想バンクレジスタを現在のレジスタに配置する．また，仮想特権モードにおいて非特権モードのレジスタにアクセスする命令が実行された際は，仮想バンクレジスタ領域の仮想非特権モードレジスタの値を参照する．

Design and Implementation of SIVARM
†Akihiro Suzuki　†Shuichi Oikawa
†Department of Computer Science, University of Tsukuba

表 1: 仮想保護レベルのドメインアクセスタイプ

	D15 (VMM)	D1 (User)	D0 (Kernel)
VMM	Client	Manager	Manager
User	Client	Manager	No access
Kernel	Client	Manager	Manager

3.2.2 ドメインによる仮想保護レベル

本研究では仮想特権モード上で動作する Linux のカーネルプロセスを仮想非特権モード上で動作するユーザプロセスから保護するために，ドメインと呼ばれる ARM アーキテクチャ特有の機構を利用する．

ドメインは仮想メモリ上のメモリ領域の集合を表すことのできる機能であり，仮想メモリ上の各ページは 16 個のうちいずれかのドメインに属している．ドメインにはドメインアクセスタイプと呼ばれるアクセス制御を指定でき，“Manager”はアクセス許可，“No access”はアクセス不可，“Client”はドメインによるアクセス制御は行わず，ページングによるアクセス制御を行う．

本研究でゲスト OS として使用する Linux はカーネルプロセスをユーザプロセスから保護するためにページングによるアクセス制御を利用しているため，ドメインは常に“Client”に設定されている．そのため，本研究ではドメインアクセスタイプを表 1 のように設定し，非特権モード上に仮想的に 2 レベルの保護を提供する．これによって，特権と非特権の 2 レベルしか存在していなかった ARM アーキテクチャの保護レベルを仮想的に 3 レベルにすることが可能となる．

4 実験・評価

4.1 実験環境

実験環境は以下の通りである．

- OS　Linux 2.6.26-at6
- ボード　Armadillo-500
- CPU　ARM1136JF-S (400MHz)
- メモリ　64MB

4.2 実装量評価

実装量は以下の通りである．ゲスト OS への変更は代理特権命令への置換が大半であり，その変更量は非常に少ないものとなっている．

- C 言語　6835 行
- アセンブリ言語　190 行
- ゲスト OS への変更　76 行

4.3 性能評価

ネイティブ Linux と SIVARM を導入した Linux でマイクロベンチマークを実行した．その結果を表 2 に示す．なお，実行結果の単位はマイクロ秒である．

実行結果より，システムコールの速度比が他のベンチマークに比べて高いことが分かる．これはシステムコールの実行する命令数が少ないため，VMM を導入したことによるオーバヘッドが他のベンチマークに比べて相対的に大きくなってしまったことによるものであると考えられる．また，他のベンチマークについても性能の低下が大きい．ARM アーキテクチャは組み込み機器用の CPU であるため，リアルタイム性が要求される場面での使用が想定される．そのため，VMM の性能をより向上させることが必要となる．

表 2: マイクロベンチマークの実行結果

Benchmark	NativeLinux	SIVARM	速度比
fork+exit	834.95	5,576.2	6.68
fork+exec	926.35	5,864.0	6.33
pipe	42.240	225.89	5.35
syscall	0.77250	14.745	19.09

5 関連研究

5.1 XenARM

XenARM[3] は Xen[4] を ARM アーキテクチャに移植するプロジェクトである．この実装には準仮想化という仮想化方式が用いられており，ゲスト OS への変更量が大きい．本研究で開発した SIVARM はゲスト OS への変更量が小さいため，例えば新しい Linux のバージョンにもすぐ対応することができる．

5.2 TrustZone

TrustZone[5] は ARM1176JZF-S から採用された，CPU による仮想化支援機能である．これを用いることで ARM アーキテクチャのための VMM を容易に実現できる可能性があるが，VMM を導入できる ARM アーキテクチャのバージョンが限定されてしまう．一方，本研究で開発した SIVARM は全ての ARM アーキテクチャを対象とすることができる．

6 まとめ

本論文では SIVARM の設計と実装について述べ，ARM アーキテクチャのための VMM を設計実装する際に必要な要素を示した．今後は TLB・キャッシュのロックダウンや VMM への遷移回数を減らすための工夫を施すことで SIVARM の性能を向上させると共に，複数 VM の提供を可能にするような機能拡張を目指す．

参考文献

[1] R. P. Goldberg: Survey of Virtual Machine Research, IEEE Computer, Vol.7, No.6, pp. 34-45 (1974)

[2] Andrew N. Sloss, Dominic Symes, Chris Wright: Arm System Developer's Guide Designed and Optimizing System Software, Morgan Kaufmann Pub (2004)

[3] Joo-Young Hwang, Sang-Bum Suh, Sung-Kwan Heo, Chan-Ju Park, Jae-Min Ryu, Seong-Yeol Park, Chul-Ryun Kim: Xen on ARM: System Virtualization using Xen Hypervisor for ARM-based Secure Mobile Phones, CCNC 2008, pp. 257-261 (2008)

[4] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield: Xen and the Art of Virtualization. SOSP 2003, pp. 164-177 (2003)

[5] Alves, T. and Felton, D. TrustZone: Integrated Hardware and Software Security, ARM (2004)