

# 仮想マシンを支える ハードウェア技術(インテル)

Hardware Assisted Virtualization Technology

岩本成文 インテル(株)

## 【インテル・アーキテクチャと仮想化技術】

### 《インテル・アーキテクチャ上の仮想化手法》

かつてメインフレームなど大型の計算機システム上でのみ利用されていた仮想化技術は、現在では、インテル・アーキテクチャ上で<sup>☆1</sup>、ハードウェア・レベルのサポートが提供されるに至る段階まで広がってきている。本稿では、現在も大きな広がりを見せているインテル・アーキテクチャ上の仮想化技術に関して、重要と思われる技術の概要と今後の発展にも言及することとしたい(図-1)。

インテル・アーキテクチャ上で仮想化を実現する技術には、いくつかのアプローチがある。大別すると、ソフトウェア処理のみで実現する手法とハードウェア支援機能を活用する方法があり、またソフトウェア処理のみで実現する手法にも、VMware社のバイナリ・トランスレーションと呼ばれる技術や、Xenに代表されるパラ・パーチャライゼーション技術といった異なる複数の技術がある。本稿では、これらの技術を広く解説するよりは、なぜインテル・パーチャライゼーション・テクノロジー

(以下、インテルVTと言う)などのハードウェア支援機能が必要となるのか、という点に焦点を絞って解説する。この点を理解するためには、インテル・アーキテクチャ上でのソフトウェア処理による仮想化の課題を理解することが重要となる。

### 《ソフトウェアによる仮想化技術とその課題》

Xenに代表されるパラ・パーチャライゼーション技術は、ゲストOSが直接ハードウェアを呼び出すのではなく、VMMを呼び出す形に修正して、呼び出されたVMMがハードウェア・アクセスを調停することにより仮想化を実現する手法である。しかし、この手法では、ゲストOSの書き換えが必要となるため、対応可能なゲストOSに限られてしまう。一方で、バイナリ・トランスレーションに代表されるダイナミックに特権命令を調停する手法は、ゲストOSを本来の特権レベルと異なる特権レベルにインストールし(図-2)、それによって発生する特権命令のフォールト・ハンドラを経由することにより、複数OSの調停機能を提供する<sup>☆2</sup>。ただ、後者の手法によっても、たとえば、VMM用のメモリ空間

☆1 本稿では、紙面の都合で、主にIA-32およびIntel64上のVTアーキテクチャを前提とさせていただくことをご了承いただきたい。Itaniumアーキテクチャ向けのVT-iも基本となる概念は同様であり、詳細については別途文献2)等を参照してほしい。

☆2 パラ・パーチャライゼーション技術においても、ゲストOSはRing 1などの本来の特権レベル(Ring 0)にインストールされるのが一般である。

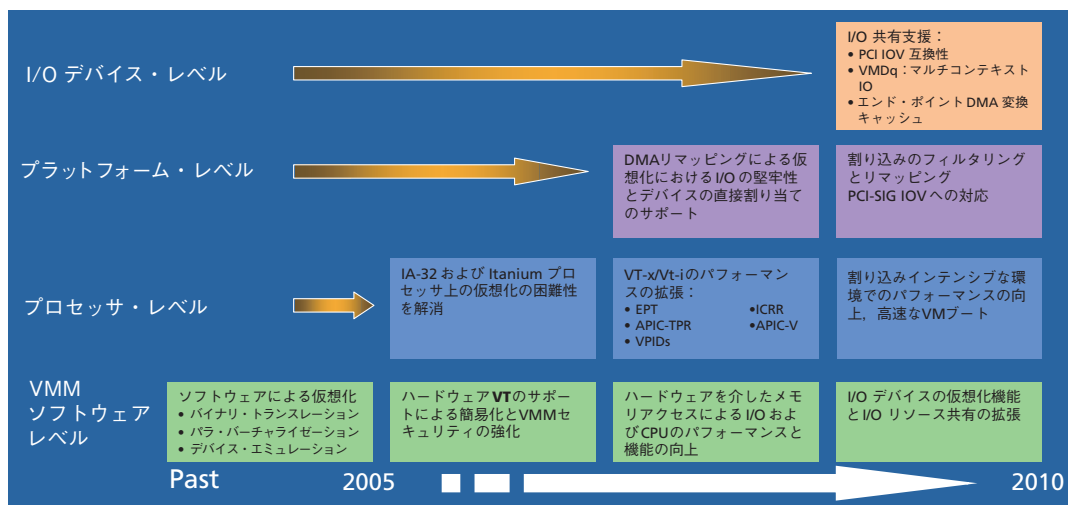


図-1 インテル・パーチャライゼーション・テクノロジーロードマップ



図-2 ソフトウェアベースの仮想化とインテルVT

の保護にセグメンテーション境界を利用する場合は、仮想マシン上ではセグメンテーションがない64ビットモードは利用できないことになる可能性がある。

いずれの手法も、きわめて高度な優れたソフトウェア技術であって、現実の利用形態に着目し、これに伴う制限を受け入れ可能な限り、ソフトウェア処理による仮想化は十分実用的である。しかし、このようなゲストOSを修正したり、ゲストOSを本来開発されたモード以外で動作させたりといった手法は、OSも含めた互換性およびサポートという点でなお不安を残す。結局のところ、CPUのアーキテクチャそのものが、VMM専用のモードを持つことが望ましいことに疑いはないだろう。そこで、インテルVTは、このような、従来のインテル・アーキテクチャに内在する仮想化の困難性を解消し、非仮想化環境と同じアーキテクチャを仮想マシン上でサポートするアーキテクチャを定義している。以下、まずインテルVTの技術概要を簡単に紹介し、続いてI/Oの仮想化について解説する。

## インテルVT概要

### 《 Virtual Machine Extensions (VMX) 》

前述のように、ゲストOSを本来あるべき特権モード(Ring 0)で動作させつつ、仮想化を実現するためには、CPUレベルで、従来の特権構造とは独立した、いわばVMM専用モードを設けることが必要となる。そこで、インテルVT-xでは、「Virtual Machine Extensions (VMX)」として新しいCPU動作モードを定義することによってこれを実現している<sup>1)</sup>。VMXモードには、「VMX Rootモード」というVMM専用のモードと、

“VMX non-rootモード”というゲストOSが動作するモードがあり、これらのモードは、“Ring”と呼ばれる特権レベルの構造とは独立した概念として定義されているため、それぞれのモード内で、従来の4つの特権レベルがすべて利用可能である。したがって、ゲストOSは設計上想定された特権レベルで動作でき、また、アプリケーションにとっても、OSがアプリケーション開発時と同じRing 0で動作しているため、特にRing 0で動作するコンポーネントを含むアプリケーションの場合は、互換性が高い。

### 《 モード間の状態遷移：VM EntryとVM Exit 》

そして、これらのモード遷移を制御するため、「VM Entry」と「VM Exit」と呼ぶ2つの新しい状態遷移が定義された。すなわち、VMX non-rootモード下のあるゲストOSから、調停が必要な特権命令が発行された場合、ハードウェアによりVM Exitという状態遷移が発生し、制御がVMX Rootモードに移行する(図-3)。このような仕組みにより、複数のOSでは両立し得ないCPUリソースの仮想化を可能にしている。逆に、VMMが必要な処理を行った後、すでに起動している仮想マシンに制御を戻すには、VMRESUME命令を実行する。これにより、再びVM Entryという状態遷移が行われ、プロセッサの動作モードは、VMX non-rootモードに移る<sup>☆3</sup>。このVMX non-rootモードは、VM Exitという状態遷移が発生し得る以外は、従来のCPUと同じ機能を有しているため、制限を伴わないフルセットのCPU

☆3 この際、プロセッサは状態遷移前のVMX non-root時のステートに復元される。そのステート情報や復元における細かなオプションを指定するのが後述のVMCSである。

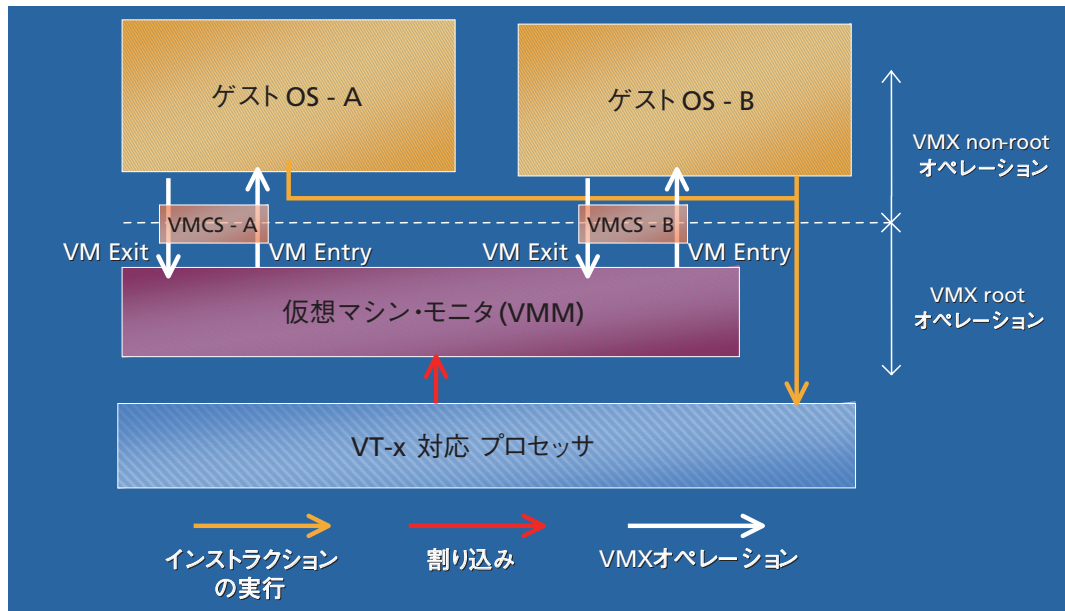


図-3 インテル VT-x アーキテクチャ概要

の仮想化が可能となる。

さらに、この遷移を管理・制御するデータ構造として、仮想マシン制御構造 (Virtual Machine Control Structure : VMCS) が定義され、また、VMX オペレーション用と VMCS 管理用の新しい命令セットも定義されている (VMX インストラクション)。VMCS および VMX インストラクションの詳細については、文献 1) 等を参照してほしい。

## 【ダイレクト I/O 対応インテル VT 技術概要】

### 《 I/O の仮想化の要件 》

I/O の仮想化において要求される要件仕様はさまざまであり、かつそれぞれの要件の重要度は、利用モデル (とりわけエンタープライズ・サーバ統合とクライアントにおける仮想化) において異なる。もっとも、これらの要件に重要な共通要素を見出すこともでき、インテルでは以下の 2 点を要件として重視している。

- ① 1 つ目は、任意の仮想マシン (VM) から I/O リソースへのアクセスを保護して、同一プラットフォーム上にある他の VM の動作に干渉しないようにすること (ハードウェア・レベルの VM の分離)。この VM 間のハードウェア・レベルでの分離は、仮想化の可用性、信頼性を確保する上で必須といえよう。
- ② 2 つ目は、複数の VM で I/O リソースを共有できること。多くの場合、任意のプラットフォーム上にある VM ごとに I/O リソース (ストレージ・コントローラやネットワーク・コントローラなど) を複製するのは現実的でない。

本稿では、I/O の仮想化におけるハードウェア支援の

必要性を最も理解しやすいと思われるデバイスの VM への直接割り当て (パススルー) モデルを念頭に、ハードウェア支援の必要性を検討しながら、ダイレクト I/O 対応インテル・パーチャライゼーション・テクノロジー (以下 VT-d と呼ぶ)<sup>3)</sup> の技術概要を概説する。この場合、VM 間のデバイス共有が課題となるが、その点については、PCI-SIG (PCI-Express) の IOV スペック等を参照してほしい<sup>4)</sup>。

I/O デバイスを扱うための処理には大きく分けて、(1) デバイスの検出、(2) 制御 (処理のスケジューリング)、(3) データ転送、(4) 割り込みなどがある。このうち、(1) および (2) は CPU の仮想化メカニズムで対応可能であるため、VT-d では DMA (Direct Memory Access) と割り込みの仮想化をハードウェアでサポートしている。

### 《 DMA のリマッピング 》

#### (1) ホスト物理アドレスとゲスト物理アドレスの不一致

通常、ゲスト VM のメモリ空間は、実際のホストシステムの物理アドレス (HPA, Host Physical Address) ではなく、そのサブセットのアドレス空間 (GPA, Guest Physical Address) が使われる。すなわち、ゲスト VM がホストの物理メモリだと思っているものは、実際には、物理システムのアドレス空間のサブセット<sup>☆4)</sup> にすぎないのである。そして、CPU を介さずにシステムメモリに直接データを転送する DMA 転送においては、ゲスト VM のデバイスドライバが GPA を使用し、デバイスそのものは、HPA を使用する。したがって、ゲスト

☆4 厳密には、アドレス空間のサブセットではなく、インテル VT 上では、別個の異なるアドレス空間である。ここでは、実際の物理メモリの部分集合であることを、理解の便宜のため、こう表記した。

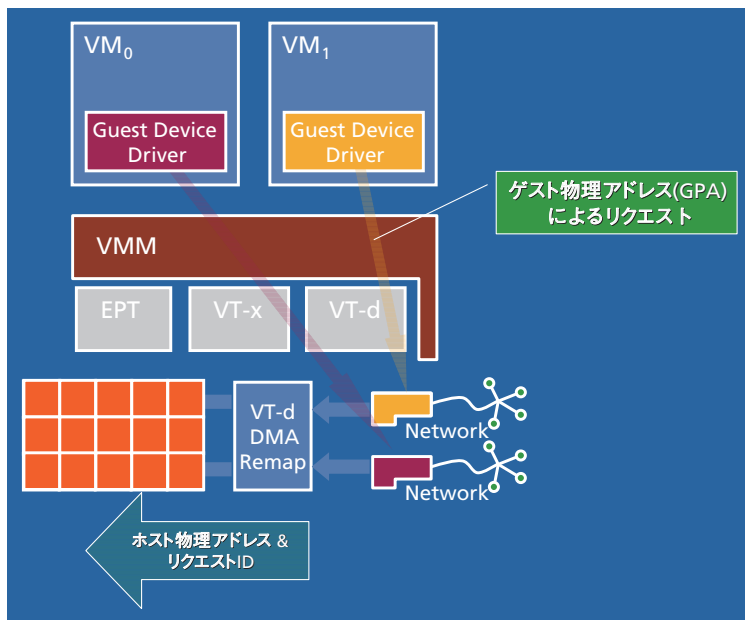


図-4 DMA リマッピング

VM が持つメモリ空間に DMA 転送を行う際には、誰かが GPA を、HPA に変換しなければいけないのである (図-4)。

ここで、ゲスト VM のデバイスドライバと物理デバイスのあらゆる通信をインターセプトするパススルードライバを用いてこのアドレス変換を行うというソフトウェア手法もある。有効な方法ではあるが、デバイス固有のコマンドをデコードする必要のある場面も多く、VMM の複雑性が相変わらず高いことは否定できない。信頼性や VM 間の分離という点では、やはりハードウェアによるサポートの必要性が高い部分といえよう。また、ソフトウェアによる対応のみだと、当該ソフトウェアに不具合があれば、ある VM から異なる VM のホスト上の物理アドレスへのアクセスが許容されてしまう

こともあり得るので、そのセキュリティ上のリスクも看過できない。

(2) DMAリマッピングのハードウェア・サポート

DMA リマッピングは、従来より、さまざまなコンテキストでハードウェアに実装されている。たとえば、PCI のルートブリッジに実装された伝統的な I/O メモリ管理ユニット (IOMMU) や、AGP の GART (Graphics Aperture Remapping Table) などである。ただ、これらの伝統的な IOMMU は、通常、アパーチャ・ベースのアーキテクチャをサポートしており、リクエスト元とは無関係に、ホストの物理アドレス空間にプログラムされたアパーチャ・アドレス範囲を DMA 変換する。そのため、限定的なレガシーデバイスのサポートには便利だが、VM 間を超えて DMA の分離を必要とする I/O 仮想化の用途には必ずしも適切とはいえない。そこで、インテル VT-d では、複数の DMA 保護ドメインを作成できる IOMMU アーキテクチャを採用している (図-5)。

具体的には、インテル VT-d では、DMA リマッピング・ハードウェアが各 DMA リクエストの発行元であるデバイスを識別するために、PCI バス、デバイスおよびファンクションの番号をデバイスのリクエスター識別子とする<sup>☆5</sup>。そして、保護ドメインへの I/O デバイスのマッピングのために、①ルートエントリ・テーブルと

☆5 仮想化における IOMMU アーキテクチャにおける、DMA 要求を行ったデバイスおよびファンクションを識別することの意義を留意されたい。およそデバイスとは無関係に一定範囲のホスト物理アドレス範囲を変換することも可能だが、その場合、ハードウェアレベルの VM の分離が達成できない。

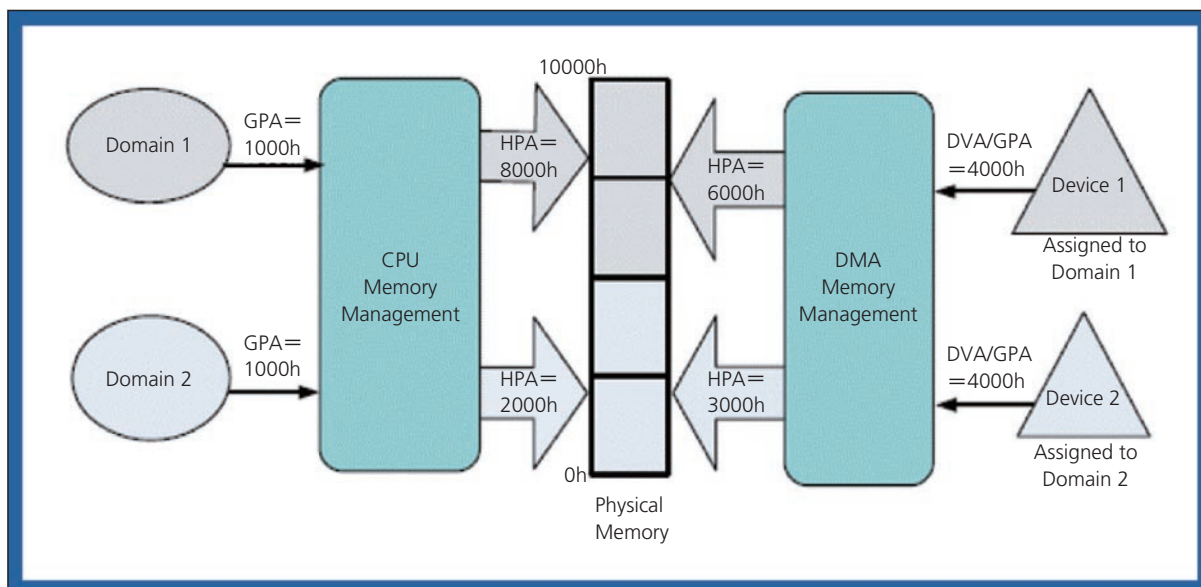


図-5 保護ドメインとデバイス

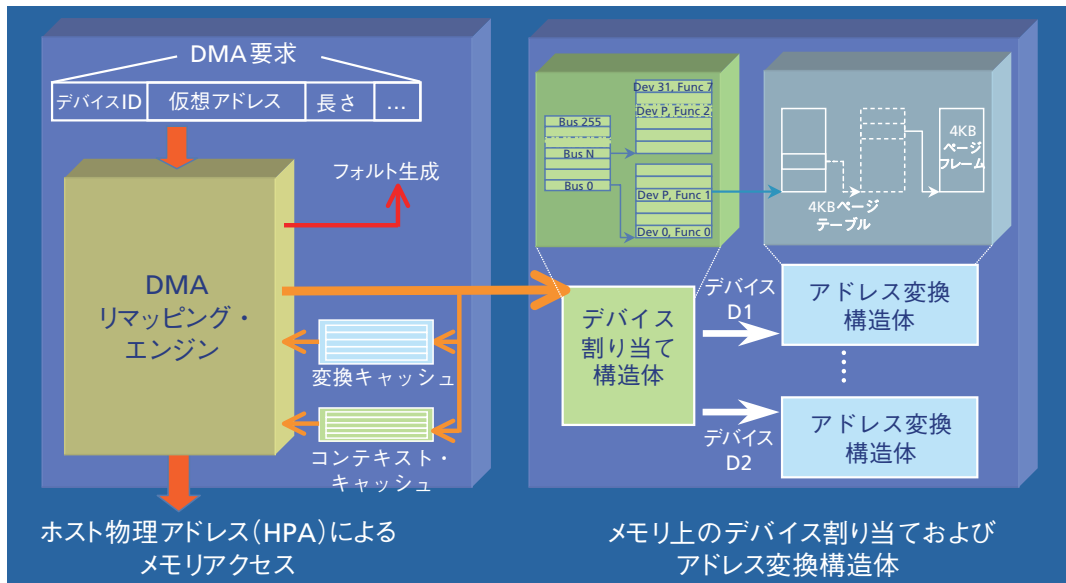


図-6 ダイレクトI/O対応インテルVTアーキテクチャ

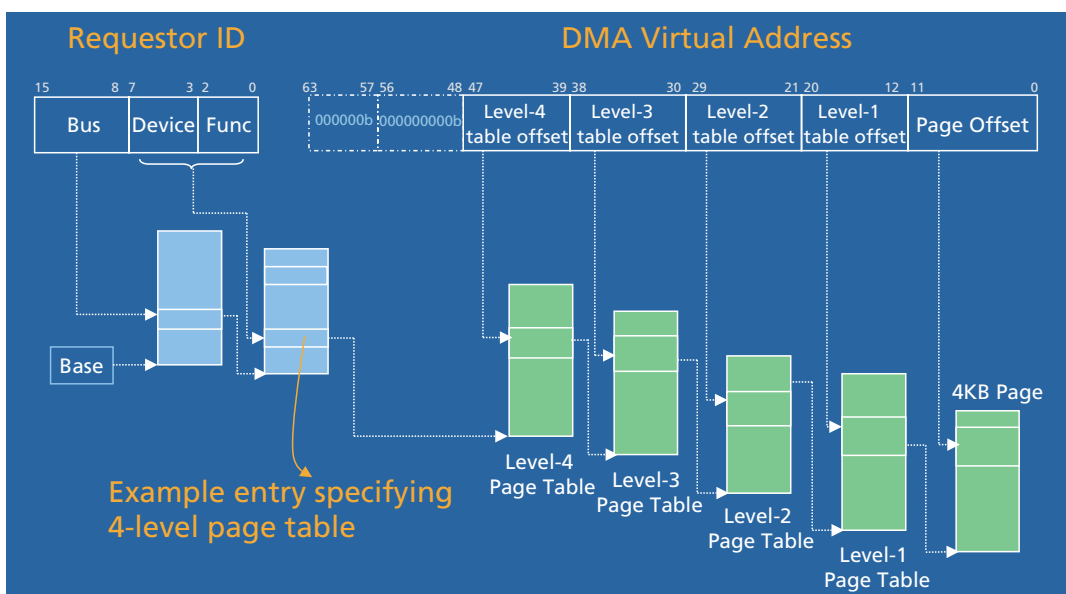


図-7 DMAリマッピング：ページウォーク

②コンテキスト・エントリ・テーブルの2つのデータ構造を、デバイス割り当て構造体として定義している(図-6)。

①ルートエントリ・テーブル：各エントリがトップレベルの構造体として機能し、個々のPCIバスにデバイスをマッピングする。そして、存在するルートエントリは、それぞれコンテキスト・エントリ・テーブルへのポインタを保持している。

②コンテキスト・エントリ・テーブル：各エントリは、バス上の個々のI/Oデバイスを割り当てられた保護ドメインにマッピングする。そして、存在するコンテキスト・エントリは、それぞれDMA要求のアドレスを変換するアドレス変換構造体へのポインタを保持している。

さらに、ルートエントリ・テーブルに含まれるDMA

アドレス変換用のアドレス変換構造体は、IA-32プロセッサのページテーブルに似たマルチレベルのページテーブルとなっており、ソフトウェアが4KB以上のページ粒度でメモリを管理可能になっている(図-7)。

#### 《 割り込みのリマッピング 》

割り込みの効率的な実行(スケジューリング)は、原則としてCPUの仮想化メカニズムが担当する。すなわち、VMMは、ゲストに伝達すべき割り込みを保持している場合、割り込みウィンドウによるVM Exitコントロールをセットしておけば、ゲストの割り込みがアンマスクされたタイミングでVM Exitを発生させることができ、VMMは、複雑なソフトウェア処理なくして、容易に制御を獲得することができる。さらに、VMMは、VM Entry制御フィールドを設定すれば、VMCSのゲスト・

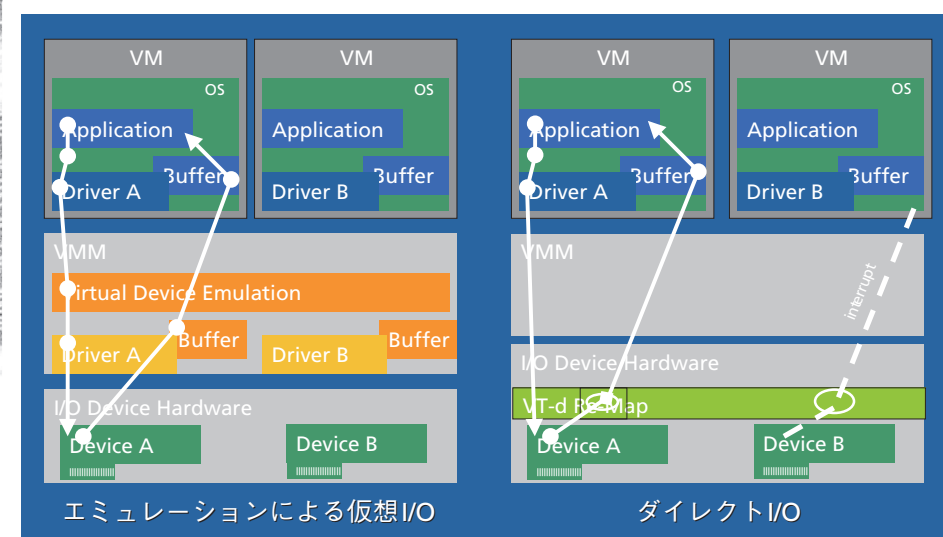


図-8 インテルVT-dによるデバイスの直接割り当て

ステート領域からプロセッサ・ステートをロードするVM Entryの際に、続いてゲストのIDTを使用して割り込みまたは例外を挿入することも可能となっている。

もっとも、割り込みといっても、たとえば、あらかじめ設定されたアーキテクチャ・アドレス範囲にDMA書き込み処理として発行されるメッセージ・シグナル割り込み(MSI)などもある。この場合は、既存のアーキテクチャでは、特にVM間のハードウェア・レベルでの割り込みの分離を達成できない。そこで、VT-dでは、割り込みメッセージの形式を再定義することにより、I/O割り込みコントローラ(IOAPIC)を含む、あらゆるソースの割り込みメッセージと、PCIの仕様で規定されている全種類のMSIとMSI-X割り込みのリマッピングを可能にするアーキテクチャを採用している。この場合、DMA書き込み要求は、ハードウェアによって割り込み要求とみなされ、リクエスターIDが、テーブル構造によってリマッピングされる。割り込みリマッピング・テーブルの各エントリは、デバイスからの固有の割り込みメッセージ識別子に対応し、必要な割り込み属性(目的のプロセッサ、ベクトル、配信モードなど)をすべて備えている。

#### 《ハードウェアによるキャッシングと無効化》

以上のリマッピング構造体はメモリ上に構築され、VT-dでは、オーバーヘッド低減のために、ハードウェアによって、よく使われるリマッピング構造体のエントリをキャッシュできるようになっており(図-6)、これをIOTLBと呼ぶ。そして、キャッシュの整合性を維持するため、ソフトウェアがエントリを無効化できる仕組みも定義されている。

なお、アドレス変換キャッシュのスケーリングを容易にするため、PCI-SIG(PCI Special Interest Group)に

よってPCI-Expressのプロトコル拡張(アドレス変換サービス:ATS)の標準化が進められている<sup>5)</sup>。興味のある読者はこちらも参照してほしい。

#### 《リマッピングエラーの処理》

DMAリクエストのリマッピング中に検出されたエラーやパイオレーションは、リマッピング・フォールトとして処理される。フォールトを生成したDMA書き込みリクエストはリマッピング・ハードウェアによってブロックされ、デバイスにエラーを返す。そして、ハードウェア・ログに、リマッピング・フォールトの原因であるDMAリクエストが詳細に記録され、フォールト・イベント(割り込み)を使ってソフトウェアに報告する。以上のような仕組みによって、インテルVT-dでは、ハードウェア・レベルでのVM間のデバイスの分離を実現している。

#### 《I/Oの仮想化の課題と今後》

DMAリマッピングがハードウェアでサポートされることにより、デバイスドライバは、ゲストの物理アドレス環境(GPA)でそのままDMAリクエストを行うことが可能となる。その結果、ゲストOS上では、非仮想化環境のデバイスドライバをそのまま利用することができる。ハードウェア・サポートによるデバイスのVMへの直接割り当ては、特別なデバイスドライバをも必要とせず、エミュレーションによるオーバーヘッドを低減する手法としてきわめて有用だといえよう(図-8)。

もっとも、課題もある。ゲストVMへのデバイスを直接割り当てれば、そのままでは、複数のゲストVMによる効率的なデバイスの共有ができなくなり、冒頭で指摘したデバイスの共有という重要な要件を達成できないことになる。この点については、PCI-SIGが、デバイ

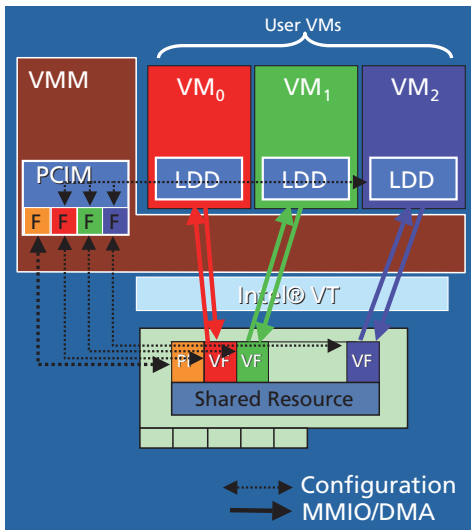


図-9 VT-dとIOV

ス共有の実現に向けて PCI-Express の仕様強化を行っている (IOV)。図-9 にその概要を示すので、詳細は PCI-SIG の IOV スペックを参照してほしい<sup>4)</sup>。

### 【拡張版インテル・バーチャライゼーション・テクノロジー】

拡張版インテル VT に搭載される機能はいくつかあるが、本稿では、読者の興味が高いと思われる Virtual-processor ID (VPID) と Extended Page Table (EPT) に絞って解説する。

#### 《 Virtual-processor ID (VPID) 》

TLB (Translation look-aside buffer) は仮想アドレスから物理アドレスの変換をキャッシュするが、前述のように、仮想マシン上における物理アドレス (GPA) と VMM が扱うホストの物理アドレス (HPA) はそれぞれ異なるものであるため、VMX オペレーションの遷移が起こると、TLB はフラッシュされてしまう。そのため、特にその必要がない場合には、仮想マシンのパフォーマンスに影響を与える。そこで、拡張版インテル VT では、この TLB フラッシュによるパフォーマンスのペナ

ルティを避けるため、独立した複数のアドレス・スペースのキャッシュが可能な TLB のアーキテクチャを定義し、VM 実行コントロールに追加した。これを Virtual-processor ID (VPID) と呼び、VPID 環境では、VMM などのホストソフトウェアが 0 番の ID を、それ以外の仮想マシンは非 0 番の固有の ID を利用する。そして、この実行コントロールがセットされていると、ゲスト仮想マシン実行中は、ゲストの VPID が利用され、VM Exit/Entry の際も、TLB フラッシュは起こらない。もちろん、処理内容によっては TLB をフラッシュさせる必要がある場面もある。そこで、VPID 環境では、VMM が VPID ごとに TLB をフラッシュさせることのできる新しいインストラクションも追加されている。

#### 《 Extended Page Table (EPT) 》

初代インテル VT の実装は、ゲスト仮想マシン上でのコントロール・レジスタへのアクセスやページフォルトがあると、VM Exit をし、実行を VMM に遷移させる仕様となっている。ページのコントロールを VMM の管理下におき、数多くあるメモリ仮想化のアルゴリズムを最大限にサポートするためである。

ここにおけるメモリの仮想化アルゴリズムの代表的なものに、仮想 TLB (Virtual TLB) がある<sup>1)</sup> (図-10)。これは、以下の 2 つによって管理されるアクティブ・ページ・テーブルから構成される。

- (1) 仮想マシン上のリニア・アドレスと仮想マシン上の物理アドレス (GPA) の変換を行い、ゲストによって読み書きされるゲスト・ページ・テーブル。
- (2) ゲスト上の物理アドレス (GPA) とホストの物理アドレス (HPA) の変換を行う、VMM。

この場合、VMM は、たとえばゲストのページテーブルがページフォルトを起こした場合に、アクティブ・ページ・テーブルを更新することによって、メモリの仮想化を実現する (図-10)。

この手法はきわめて優れたメモリの仮想化アルゴリズムの 1 つであるが、特にゲストのページフォルトに

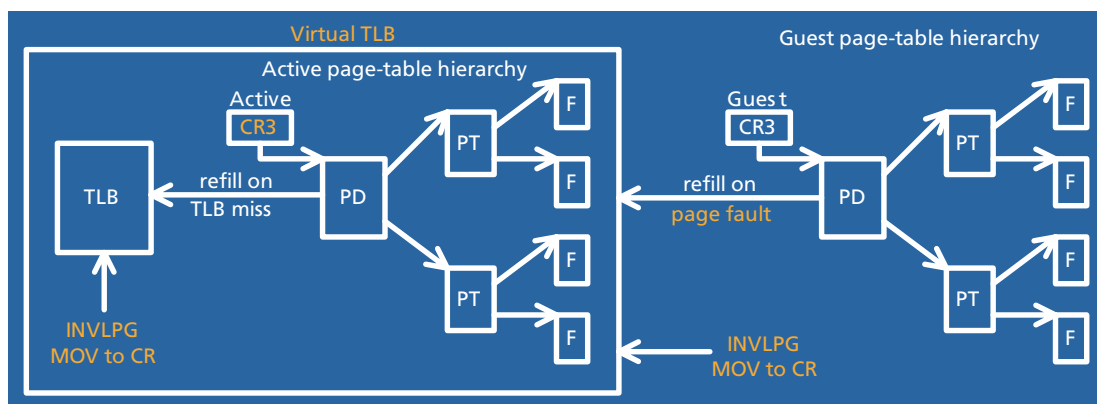


図-10 仮想 TLB

より常に VM Exit が発生する点で、ゲスト上のアプリケーションのページ利用状況によっては、パフォーマンス上のボトルネックになる可能性は否定できない。

そこで、拡張版 Intel VT では、ページフォールト、INVLPG や CR3 レジスタへのアクセスで VM Exit が生じないようにすることが可能な新しいページテーブル構造を採用した (図-11)。

すなわち、ゲストのリニア・アドレスをゲストの物理アドレス (GPA) に変換する従来のページテーブルに加え、新たにゲストの物理アドレス (GPA) を、ホスト環境の物理アドレス (HPA) に変換する拡張ページテーブルを追加し (Extended Page table : EPT), この新しいページテーブル構造は、EPT ベース・ポインタによって参照される。ページウォークの仕組みは、従来の Intel 64 ページ構造に似ており、4KB ページの場合、512 個の 8 バイトのエントリを持ち、4 段階の階層構造を持つ (図-12)。そして、読み・書き・実行の許可ビットがあり、不許可のアクセスは EPT バイオレーションとして処理され、VM Exit を生じさせる。

なお、VPID は、TLB を効率的に利用するための技術であり、EPT は特定の原因から生じる不必要な VM Exit を避けるための技術である。それぞれ技術的には別々の技術であることに留意されたい。これらの技術を評価する際には、問題となっている環境において、技術上何が課題となっているのかを十分に解析する必要があるだろう<sup>☆6</sup>。以上の仮想化技術を 1 つにまとめたものが

☆6 筆者自身は、時間の関係で、まだこれらの技術を評価できていないので断言は避けたいところだが、一般的に考えれば、EPT はメモリの仮想化におけるシャドウ・ページングのオーバーヘッドの削減に、VPID は VM の切り替えの高速化に貢献しやすいと考えられるだろう。

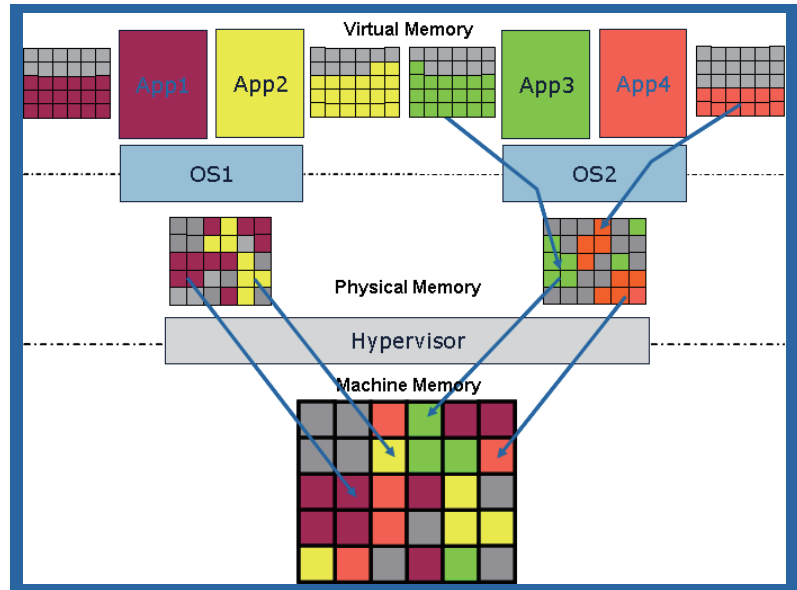


図-11 エクステンディッド・ページ・テーブル (EPT)

図-13 である。

## 仮想化技術の今後の展望

### 《 仮想化の用途の拡張 》

Intel VT は新しいアーキテクチャを定義しただけでなく、新しい利用形態の提供をも意図している。たとえば、既存の OS はそのままに、セキュリティ上の観点から、ネットワークコンポーネントのみを独立した VM として提供し、これを既存の OS と物理ネットワークに介在するセキュリティ・アプライアンスとして利用するなどである。Intel では、このような仮想化を利用したアプライアンスを、特に、バーチャル・アプライアンス (以下 VA という) と呼んで、新しい利用形態として提案している。仮想化技術によってクライアントに組み込み機器の実装が可能になったと言ってもよい (図-14)。

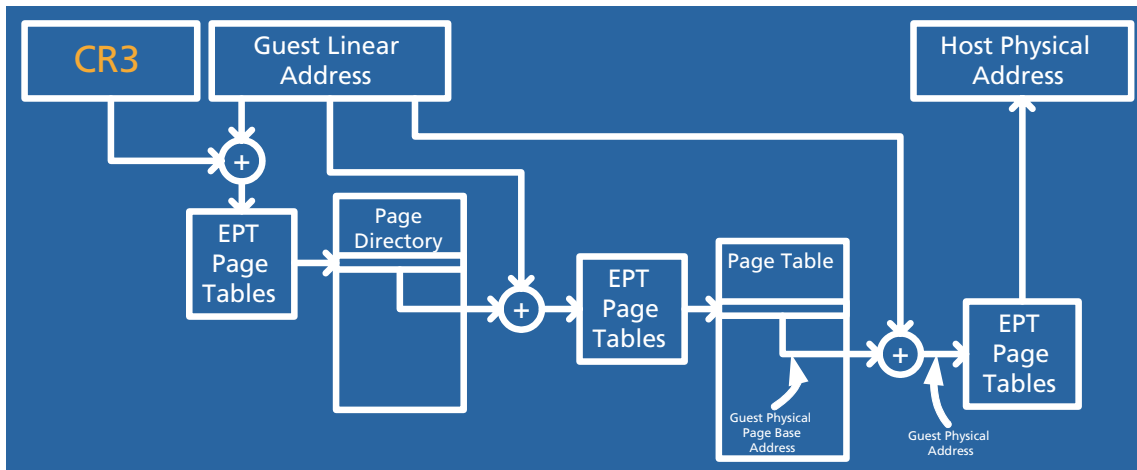


図-12 EPT ページ・ウォーク

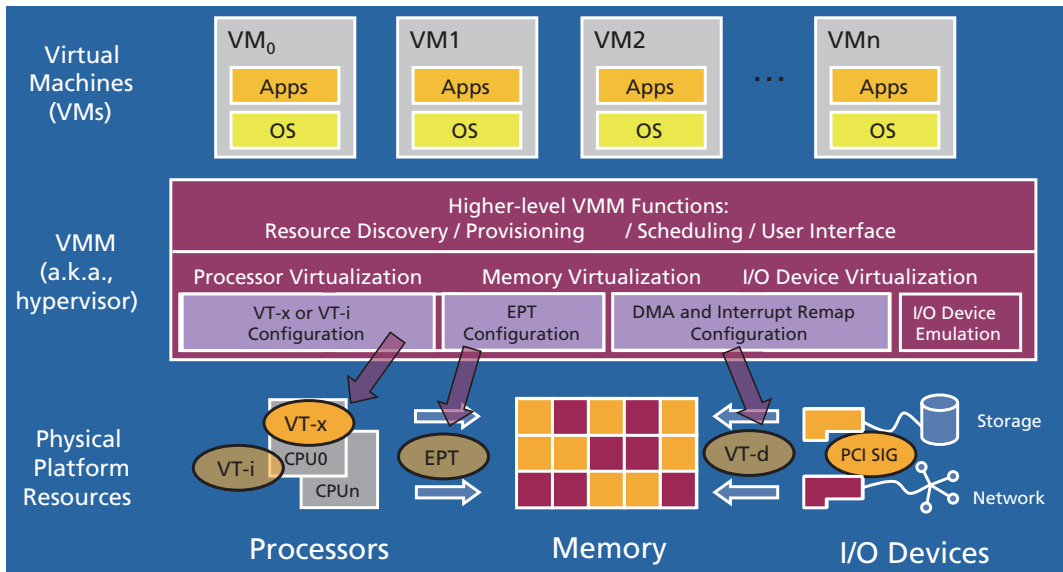


図-13 全体のまとめ

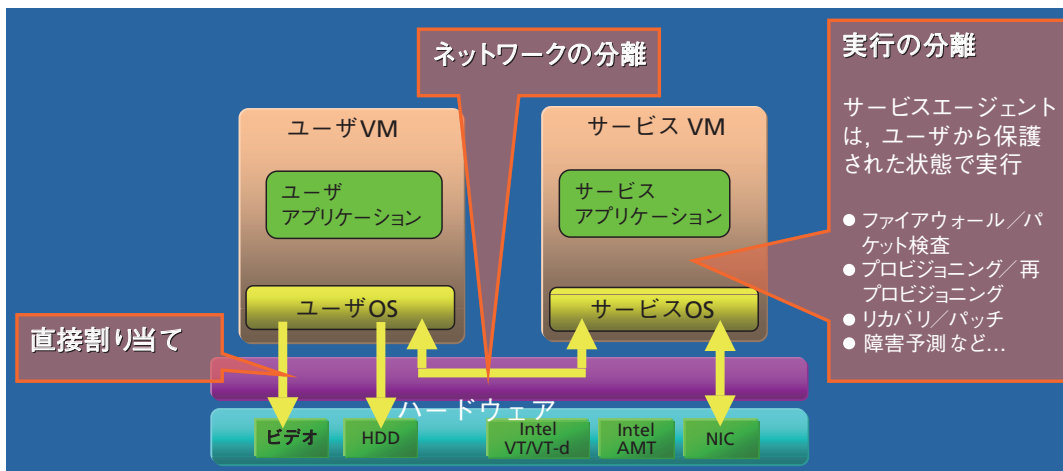


図-14 バーチャル・アプライアンスの例

ここでは、VA そのものの解説は別の機会に譲るが、このVAという利用形態は、技術的な点で、仮想化技術のあり方に対する新しい内容を示唆している点を指摘しておきたい。たとえば、従来のOSの脆弱性の影響を排除するため、ネットワークセキュリティを担う機能を、独立したゲストVMとして<sup>☆7</sup>用意する場合、このゲストVMは画面出力を用意しなければならない必然性はない。物理的なアプライアンスと同様にネットワーク越しに設定可能であればよいからである。むしろそうすることにより、画面出力は、VT-dを利用して、ユーザOSに直接割り付けすることが可能となり、UIを担当するユーザOSのパフォーマンスをそのまま維持しつつ仮想化を応用していくことが可能となる。

このように考えてみると、仮想化において、画一的なゲスト環境を提供しなければならないという決まりがあるわけではないことに気づく。より問題なのは、どのよ

<sup>☆7</sup> これをサービスOSと呼ぶ。

うな利用形態の中で何を実現するかであって、技術的には、個々のどのハードウェア・コンポーネントを仮想化するのか、しないのか、またはどのような技術を利用するのか (VT-d) という視点である。今後は、画一的な仮想マシン環境の提供という枠にとらわれることなく、利用形態から必要なハードウェア・コンポーネントの仮想化手法を検討するという新しいアプローチの検討も必要であろう。

### 《VMMの改ざんに対する対策》

仮想化技術をセキュリティ用途として応用可能であることは上述の通りだが、ただ、この際に考慮しなければならないのは、VMM そのものの改ざんに対する対策である。

#### (1) 仮想化とセキュリティ

非仮想化環境上における従来のOS上では、メモリ内容を不正に取得しようとする攻撃者は、カーネル空間、すなわちRing 0と呼ばれるモードへ侵入することに成

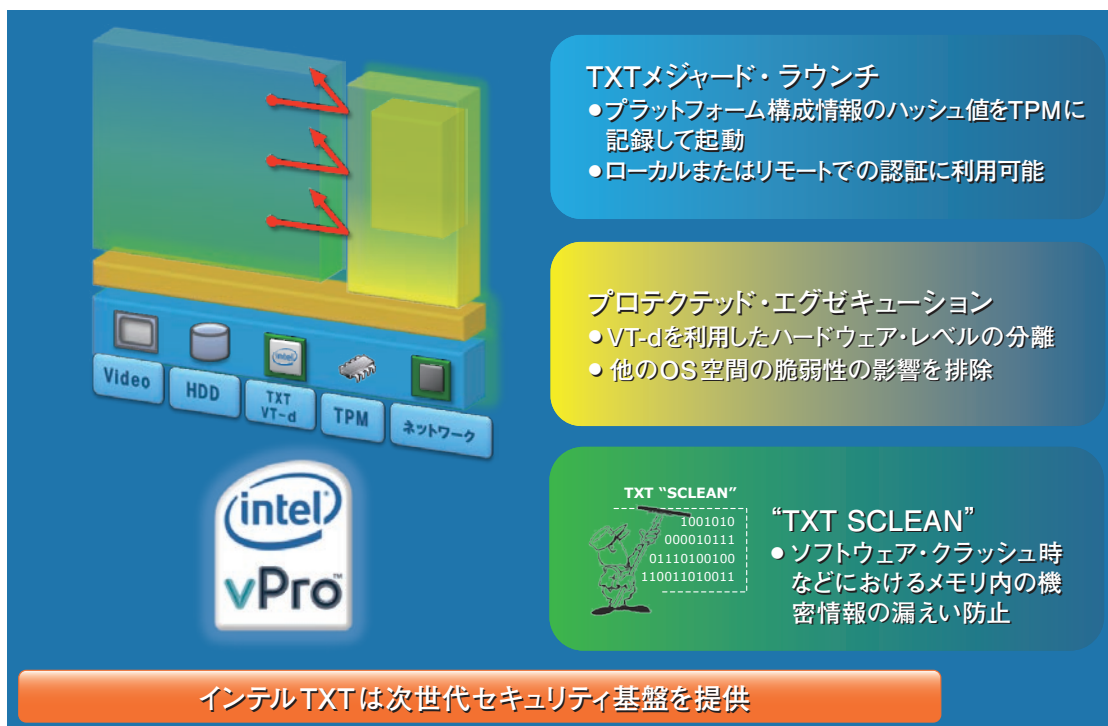


図-15 インテル・トラステッド・エグゼキューション・テクノロジー

功すれば、システムの物理メモリの内容を直接参照することが可能になる。そして、現代のOSは2段階の特権レベルしか利用せず、ユーザが入れ替え可能なドライバのカーネルコンポーネントはRing 0で動作する。したがって、攻撃者はドライバとして侵入することに成功すれば、物理メモリの内容を取得・漏洩させることが可能となる。

これに対し、仮想化環境においては、一般に、VMMが物理メモリ（ページ）へのアクセスの調停を行い、仮想マシン上の物理メモリ（GPA）は、システム上の一部のメモリにすぎない。したがって、メモリ内容を不正に取得しようとする攻撃者は、仮に従来のOSのカーネル空間への侵入に成功しても、少なくとも他のVMの物理メモリを参照することはできない。その意味では、VMMは、セキュリティ上、物理メモリに対する防波堤になっていると言える。

もっとも、そうだとすると、これをもって仮想化環境が、従来の非仮想化環境と比べ、セキュリティ上優れていると判断することはできない。なぜなら、VMM自体の脆弱性などにより、VMM自身が悪意のあるソフトウェアで書き換えられれば、結局、システムの物理メモリの参照・取得が可能となる余地があるからである。この場合、単に従来の“Ring 0への侵入”が、“VMXルートモードへの侵入”（インテルVTの場合）と名称を変えただけにすぎない。対象となる技術が変化した分、攻撃者が新たな攻撃手法を開発する必要があり、技術的なハードルは高いが、その可能性は依然否定できない<sup>☆8</sup>。

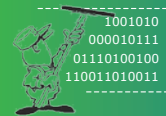
#### TXTメジャード・ラウンチ

- プラットフォーム構成情報のハッシュ値をTPMに記録して起動
- ローカルまたはリモートでの認証に利用可能

#### プロテクトド・エグゼキューション

- VT-dを利用したハードウェア・レベルの分離
- 他のOS空間の脆弱性の影響を排除

#### TXT "SCLEAN"



#### “TXT SCLEAN”

- ソフトウェア・クラッシュ時などにおけるメモリ内の機密情報の漏えい防止

インテルTXTは次世代セキュリティ基盤を提供

## (2) VMMの完全性を担保する新しい技術 — Intel TXT

したがって、仮想化環境においては、物理メモリの防波堤としてのVMMの完全性を担保すること、すなわち、VMMが1 bitの改ざんもされていないことを確認することがきわめて重要となる。とりわけファイアウォールが存在しないクライアントにおける仮想化では、VMM自体の改ざんに対する対策は、同時に検討しなければならない重要技術課題といえよう。

インテルでは、この点について、インテル・トラステッド・エグゼキューション・テクノロジー（Intel Trusted Execution Technology、略してIntel TXT、LaGrande Technologyというコードネームで呼ばれていた技術）という新しいアーキテクチャで対応する（図-15）。VMXモードに加え、Safer Mode Extension（SMXモード）および新しいインストラクションセットをサポートしたアーキテクチャ<sup>6)</sup>がTrusted Platform Module（TPM）と連携することにより、ネットワーク経由でも、VMMの改ざん痕の有無を確実に確認することが可能になる（アテストーション）。これを応用すれば、医療カルテや個人の住民票など高度なセキュリティを必要とするITインフラの構築が可能となるだろう。

なお、インテルTXTはインテルVTの拡張として導入される。なぜなら、VT-dによってハードウェア・レベルで完全なVMの分離を実現しつつ、VMMの改ざん

☆8 極端な例ではあるが、Blue PillやSubVirtといった仮想化技術を採用したマルウェアの可能性が指摘されていることをご存知の読者もいるだろう。

痕の検出を可能にすることが、クライアント PC における次世代のセキュリティの基盤となるからである。本稿では、仮想マシンをセキュリティの観点から支える技術としてインテル TXT を紹介するとともに、Trusted Computing Group (TCG) の技術および規格を学ぶことが重要であることを指摘するとともに<sup>7)</sup>。

### 《最後に》

本文でも述べたように、インテル・パーチャライゼーション・テクノロジーは、従来のインテル・アーキテクチャに内在する仮想化の困難性を解消し、フルセットの IA アーキテクチャを効率的に仮想化するプラットフォーム上の技術である。そして、この技術への理解は、PCI-SIG の IOV やインテル TXT など、多方面に派生していくさらに新しい技術のベースとなる。もっとも、筆者は、いずれの技術もその射程・応用範囲がとても広く (IOV は Multi Root Complex へ、TXT は Trusted Computing Technology へ)、全貌をキャッチアップするのが難しくなりつつあると感じている。本稿が読者の今後の研究のヒントの 1 つにでもなれば幸いである。

### (編集者追記)

今回は、近年急速に x86 の世界で浸透してきた仮想化ソリューションを支えるクアッドコア AMD Opteron プロセッサを中心とした、最新の CPU テクノロジーおよび AMD-V (仮想化支援機能) について解説する。

### 参考文献

- 1) Intel Corp. : IntelR 64 and IA-32 Architectures Software Developer's Manual Volume 3B : System Programming Guide, <http://www.intel.com/design/processor/manuals/253669.pdf>
- 2) Intel Corp. : IntelR ItaniumR Architecture Software Developer's Manual - Volume 2 : System Architecture, Revision 2.2 ; <http://download.intel.com/design/Itanium/manuals/24531805.pdf>
- 3) Intel Corp. : Intel Virtualization Technology for Directed I/O, [http://download.intel.com/technology/computing/vptech/Intel\(r\)\\_VT\\_for\\_Direct\\_IO.pdf](http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf)
- 4) PCI-SIG : <http://www.pcisig.com/>
- 5) "PCIe Address Translation Services and I/O Virtualization", WinHEC 2006 : <http://www.microsoft.com/whdc/winhec/pres06.msp>
- 6) Intel Corp. : Intel®Trusted Execution Technology Preliminary Architecture Specification : <http://www.intel.com/technology/security/>
- 7) Trusted Computing Group : <https://www.trustedcomputinggroup.org/home>

-----  
 インテル・パーチャライゼーション・テクノロジーを利用するには、同テクノロジーに対応したインテル プロセッサ、BIOS、パーチャル・マシン・モニタ (VMM)、および、用途により、同テクノロジーが有効になっている特定のプラットフォーム・ソフトウェアを搭載したコンピュータ・システムが必要です。機能性、性能もしくはその他の特長は、ご使用のハードウェアやソフトウェアの構成によって異なり、BIOS のアップデートが必要になることもあります。ご利用になる OS によっては、ソフトウェア・アプリケーションとの互換性がない場合があります。詳細については、各アプリケーション・ベンダにお問い合わせください。  
 -----

(平成 19 年 11 月 12 日受付)

### 岩本成文

[narifumi.iwamoto@intel.com](mailto:narifumi.iwamoto@intel.com)

インテル(株)ソフトウェア・エンジニア。ハードウェアに導入された新しいインストラクションやアーキテクチャの効率的な利用やソフトウェアによる最適化を担当。