

仮想マシンを支える ハードウェア技術 (AMD)

AMD processor feature optimized to support virtualization machines (AMD)

岡野浩史 日本AMD(株)

AMDと仮想化

元来メインフレームの世界で利用されていた仮想化が、近年 VMware や、Xen 等の仮想化ソフトウェアの登場により、x86 システムにおいても広く一般に利用されるようになってきた。このような状況の中、AMD では仮想化への取り組みを強化しており、2006 年に製品化した Rev. F (リビジョン F) と呼ぶ CPU (Sempron を除く) において、まず、仮想化ソフトウェアの機能を CPU がハードウェアとして支援する AMD Virtualization (AMD-V) を実装した。さらに、9 月に発表した Barcelona と呼ばれてきた第 3 世代に相当するクアッドコア AMD Opteron プロセッサでは、支援機能をさらに拡張し、Rapid Virtualization Indexing (以下「RVI」旧名称：Nested Page Table) という新機能を実装した。

AMD-Vの機能

《 仮想 OS の改変をなくす 》

AMD-V とは仮想化の支援機能だが、現在の CPU に実装されている機能は大きく 3 つある。そのうちの 1 つがこの仮想 OS の改変をなくすという機能だ。図 -1

の左が、従来のソフトウェアのみによる仮想化、右が AMD-V 対応の仮想化を示している。

仮想化環境において、仮想 OS は Hypervisor と呼ばれる仮想化ソフトウェア上に実装される。しかしながら、仮想 OS は仮想化環境で動作することを想定して作られていないため、OS を複数同時に動かそうとすると多くの問題が出てくる。その 1 つが仮想 OS が発行する特権命令だ。特権命令は、ハードウェアの変更情報を含むケースがあり、このような命令をそのまま実行してしまうと、他の仮想 OS から特定のハードウェアへのアクセスが正常に動作できなくなる等の不具合が生じる。このため、各仮想 OS が発行する特権命令を含むハードウェア変更命令を、Hypervisor が受け取り、各仮想 OS 間で不都合が生じないように調整する必要がある。x86 のシステムには、リングという概念があり、リング 0 のシステムモードからリング 3 のユーザモードまで 4 つのモードが定義されている。このリングの概念では、より低い実行権限で実行された特権命令を、高い実行権限で実行されるソフトウェアによって横取りすることが可能だ。このリングの概念を利用すれば、仮想 OS が発行する特権命令を、より高いモードで実行される Hypervisor が横取りすることができる。問題は、ハードウェアの変更を

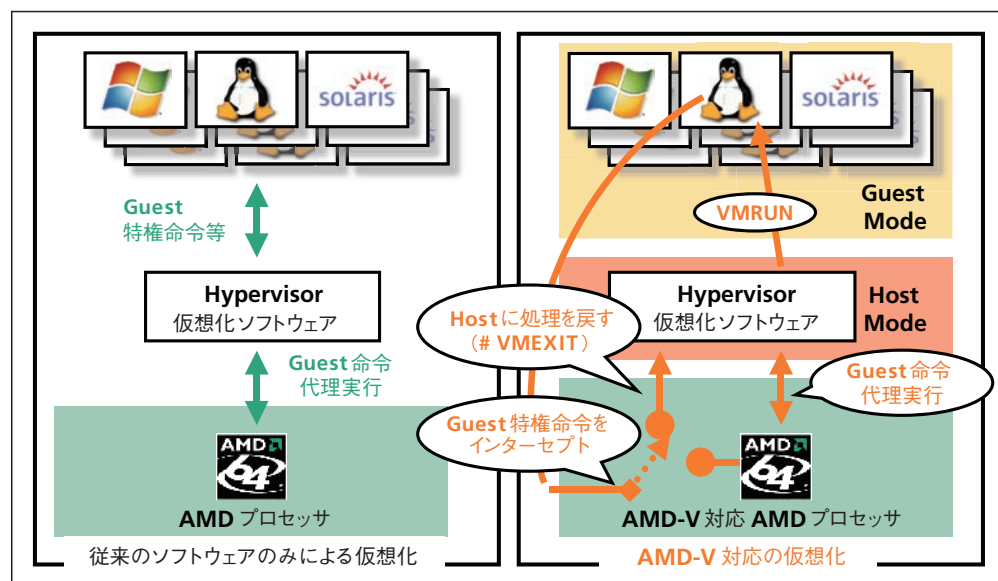


図-1 AMD-Vの機能：仮想OSの改変をなくす

伴う命令にもかかわらず、特権命令に分類されない命令だ。これには、システムのリソース状態を管理する命令などがあり、このような命令は、リングの概念を利用するだけではHypervisorから認識することができない。このため、従来のソフトウェアによる仮想化では、仮想OSにパッチを当てる必要があった。このパッチを当てる方法は、Xenなどのパラバーチャライゼーションの場合は、上記のような仮想OSが発行するハードウェアに対する変更命令を、Hypervisor経由で処理できるように、仮想OSをあらかじめ改変してインストールすることで実現する。一方VMware ESXのようなバイナリトランスレーションの場合は、リアルタイムにパッチを当てる。つまり、常時このような命令が発行されないかどうかを監視する方法だ。いずれの場合も、パッチを当てることにより、仮想OSから発行される特権命令はHypervisorでいったん処理され、Hypervisorが各仮想OSとの調整を行った上で、仮想OSの特権命令を代理実行するということが可能となる。この機能を利用することにより、Hypervisorは複数のOSを1つのハードウェア上で実行する。このようなソフトウェアによる仮想化は非常によくできているが、パラバーチャライゼーションの場合は、OSを改変する必要があるため、そもそもOSのソースコードが公開されていないところの改変自体が難しいという問題があるのと同時に、仮にソースコードが公開化されていても、OSを改変してしまうことによる動作上のサポート、保守が難しいという問題があった。逆に、バイナリトランスレーションの場合は、リアルタイムでパッチを当てるため、パフォーマンス上のオーバーヘッドが生じやすいという問題があった。

これを改善するため、AMD-Vでは、新たに、Guestモードという仮想OSが稼働するモードを追加した。この機能を実装することにより、AMDのCPUは現在CPUがHostモードの処理を行っているのかGuestモードの処理を行っているのかを理解することが可能となる。Guestモードで稼働しているとき、基本的な命令はすべてCPUで直接処理される。先ほど問題になったハードウェアの変更を伴う命令が発行されたときには、CPUがこの命令であることを判断し、処理をいったんHypervisorに戻し、Hypervisorが代行処理を行うという動作をする。こうすることにより、各仮想OSが発行するハードウェアの変更を伴う命令をHypervisorが監視する必要がなくなりパフォーマンスが向上すると同時に、OSにパッチを当てる必要がなくなるためOSの保守性が向上する。OSの改変を必要とするXenにおいて、ソースコードの公開されていないWindowsが仮想OSとして利用できるようになったのは、このCPUの仮想化支援機能が利用可能になったことが大きく寄与して

いる。

《Hypervisorの複雑さの改善》

AMD-Vの大きな機能の2つ目は、仮想化ソフトウェアの複雑さを改善しているという点だ。仮想化環境では、物理的なハードウェアを複数のOSおよびHypervisorが共用して使うため、頻繁にHostモードとGuestモードの切り換え作業が発生する。AMD-Vでは、この切り換えを高速にするため、以下の仮想化専用の命令セットを新たに9つ(セキュリティを向上する命令1つを含む)に加え、さらに、各仮想OSの状態を保存するための領域としてVirtual Machine Control Block (VMCB)を新たに追加した。追加した命令は、Guestモードに切り換えて仮想OSを起動するVMRUN、仮想マシンのプロセッサの状態をVMCBから読み込むVMLoad、実行している仮想OSの状態をVMCBに保存するVMSave等で、セキュリティに関する命令は、SKINITである。

HostモードとGuestモードの切り換えイメージを図-2に示す。たとえば、CPUがHostモードの処理を実行しているとき、VMRUNという命令が発行されると、VMCB領域に保存された仮想OSのCPUレジスタ等の状態をロードし、その状態をCPUに反映した上で、Guestモードへと処理を切り換える。逆に、Guestモードで処理が行われているときに、特権命令や、割り込み処理が発生した場合は、VMEXITという処理が行われ、VMCB領域に現在稼働している仮想OSの状態を保存し、Hostモードに切り換える。

このように、新規の命令セットとVMCB領域を追加することにより、Hypervisorがソフトウェア上で行っていたHostモード、Guestモードの切り換えと、仮想OSの状態保存に伴う複雑な処理を簡略化し、パフォーマンスの向上を実現した。

《モード切り換えの高速化》

さらに、GuestモードとHostモードの切り換えの高速化ということに関し、AMD-Vの大きな機能の3つ目として、Tagged-TLBという機能を実装した(図-3参照)。

仮想化環境が否かにかかわらず、CPUには、OSが管理するページテーブルがTLBにキャッシュとして保存されている。仮想化環境では、仮想OSごとにTLBが存在するが、AMD-Vに対応していないCPUでは、仮想OSとTLBの対応がとれないため、GuestモードとHostモードの切り換えを行うたびにTLBをフラッシュする必要があった。フラッシュを行うと、仮想OSを再度起動した際に、ページテーブルを再ロードする必要が生じ大幅なパフォーマンスのペナルティが生じる。AMD-Vでは、これを避けるため、各仮想OSのTLBをタグと

仮想化命令セット・仮想化処理を新たに実装し、複雑な処理を一括で処理

- ・VMRUN 命令：Host から Guest への処理の切り換え
 - ・#VMEXIT 処理：Guest から Host への処理の切り換え制御
- ※AMD-Vの仮想化専用命令は9つ

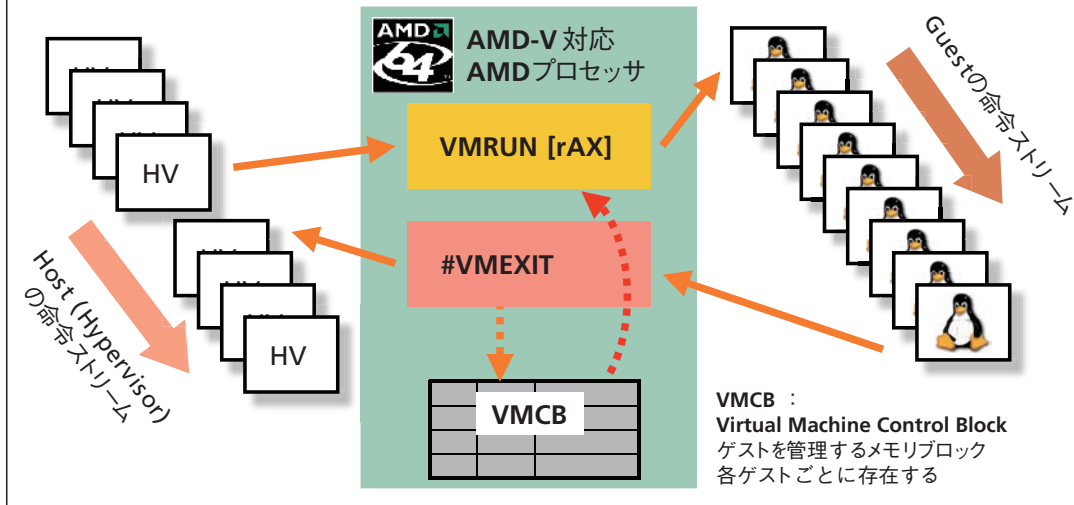


図-2 Hypervisor の複雑さの改善

課題

Guest ⇄ Host の切り換え時
TLB フラッシュによる遅延が発生

対策

TLB にタグを付け、各ゲスト、
ホストにタグ番号を付与
"Tagged TLB"

タグ番号を切り換え、

TLB フラッシュをなくし、

Guest ⇄ Host 切り換え高速化

TLB : Translation Look-aside Buffer

論理アドレスから物理アドレスへの変換で発生するページテーブル参照のペナルティを小さくするために設けられた一種のキャッシュメモリ。

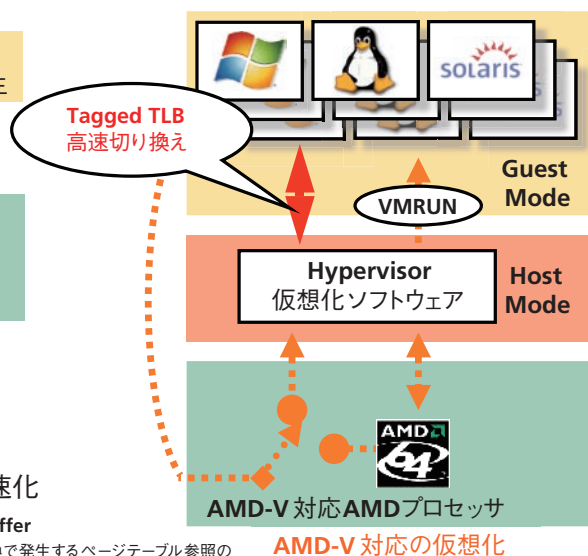


図-3 Guest, Host モード切り換えの高速化

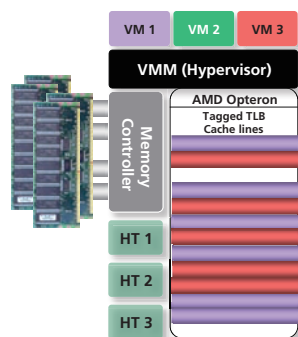
紐付けて管理することにより、Guest モードと Host モードの切り換え時の TLB フラッシュの必要性を排除し、パフォーマンスの向上を実現した。図-4 にこのイメージを示す。Hypervisor が起動して、仮想 OS が初めて起動するまで、仮想 OS に対応する TLB のキャッシュ情報は空である。次に、仮想 OS の 1 つである VM 1 が起動され、メモリからデータがロードされると、そのメモリ変換情報が TLB に記録される。この際、VM 1 の TLB は、タグと紐付けて記録されるため、CPU 側でどの仮想 OS の TLB であるか管理が可能である。このため、次に別の仮想 OS である VM 3 に処理が移った場合においても、VM 1 の TLB を保存したまま、VM 3 の TLB をキャッシュすることが可能となる。VM 1 の TLB

を残すことにより、再度 VM 1 が起動した場合、VM 1 に必要な TLB が CPU 内に残っているため、TLB の再構築が不要である。これが、TLB にタグをつけて管理する AMD-V の利点だ。

《セキュリティ機能の実装》

仮想化環境においては、各種のサービスを行う複数の仮想 OS が 1 つの物理サーバ上で実行されるので、Hypervisor を含めたセキュリティに関する考慮も当然のことながら必要となる。このような背景から、AMD では、仮想化環境にも高度なセキュリティが必要と考え、AMD-V でその機能を実装している。この様子を図-5 に示す。このセキュリティの機能は大きく 4 つあ

- TLBにタグを付けることで、各ゲストOSごとのTLBをCPUが最適に管理可能



- Tagged TLB のメリット (例):

- VM 1 VM 1が起動され、メモリからデータがロードされ、TLBに記録される
- VM 3 VM 3にゲストOS起動が移った場合、VM 1のTLBを保存したまま、VM3のTLBをロード
- VM 1 再度VM 1が起動した場合、VM 1に必要なTLBがCPU内に残っているため、TLBの再構築が不要

図-4 Tagged TLB

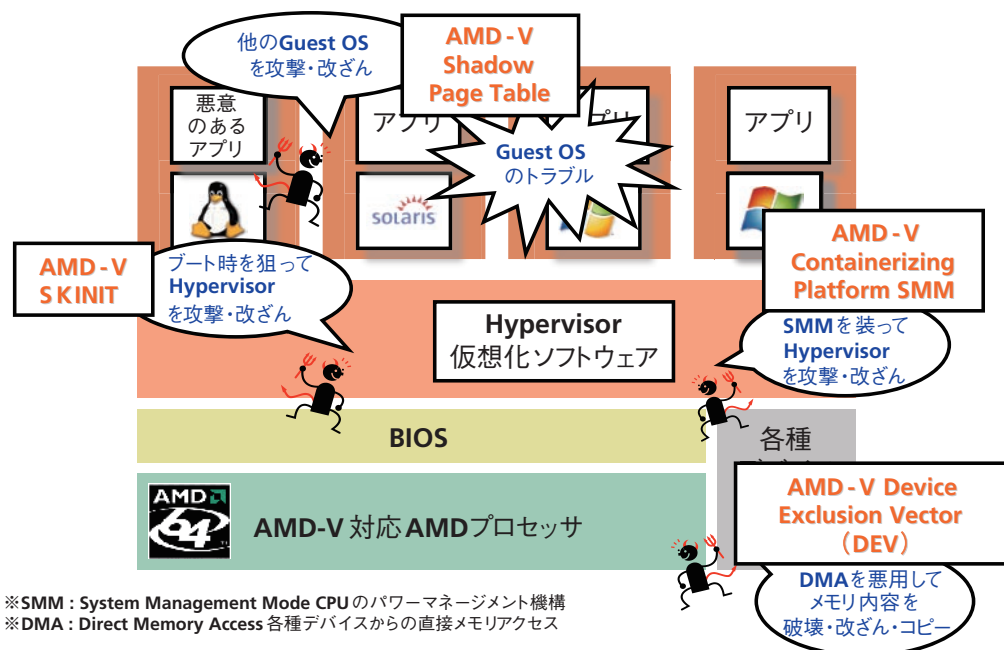


図-5 高度なセキュリティの実装

る。まず、仮にある仮想OSがVirusなどに侵されてしまったような場合、この影響が他の仮想OSに及ばないようにする必要がある。これを実現するため、AMD-Vでは、通常Hypervisor側で管理している各仮想OSごとに区切られたページテーブル（これをShadow Page Table（シャドウページテーブル）と呼ぶ）をCPU側で管理することが可能である。シャドウページテーブルは、たとえば、Virusに侵されてしまったOSから、他の仮想OSのメモリの参照や、改ざん、破壊等ができないようにして他の仮想OSを守る機能を提供するが、シャドウページテーブルをCPUで管理することにより、CPUでこの機能を実現することができる。先述の通り、この機能はHypervisorによってすでに提供されている機能であるが、ハードウェアで実装することにより、より強固な仮想OS間のメモリ境界の維持が可能となる。この機能は、メモリコントローラをCPUに内蔵するAMD

CPUならではの機能である。

次に、考慮しなければならない点は、Hypervisor自体が改ざんされた場合、仮想OSに期待されるセキュリティが実現できない点だ。たとえば、停止中に何らかの悪意を持ったコードがHypervisorに進入してしまったようなケースが考えられる。このような事態を想定し、AMD-Vには、前述のとおり、SKINITという命令を追加している。このSKINITを利用することにより、たとえば、マザーボード上に実装されたTPM（Trusted Platform Module）と連携し、改ざんされたHypervisorの起動自体をストップするという機能を実現することが可能となる。こうすることにより、意図せずアンセキュアな状態でHypervisorが起動してしまうことを防止する。

CPUの電源管理に使われるSMM（System Management Mode）を悪用したコードの進入に関し

ては、Containerizing Platform SMM と呼ぶ SMM 自体を 1 つの仮想 OS のように扱う機能により、万が一 SMM が悪意を持ったコードに侵されてしまった際にも、影響が他の仮想 OS に及ぶことを防止する。

最後に、DMA を悪用したメモリの改ざんに関しては、Device Exclusion Vector という機能を実装し、あらかじめ DMA 可能な領域と不可能な領域を定義しておくことにより DMA 不可な領域への悪意を持ったコードの侵入を防止する。この機能も、先ほどのシャドウページテーブル同様、メモリコントローラを CPU に内蔵する AMD CPU ならではの機能である。

《Rapid Virtualization Indexing の機能》

《Rapid Virtualization Indexing (RVI)》

AMD は、Rev. F 世代の CPU で、AMD-V を実装した。このことにより、OS の保守性やセキュリティの向上を実現したが、2007 年 9 月に発表したクアッドコア AMD Opteron では、Hypervisor の処理を大幅に削減することにより仮想化環境のパフォーマンスを大きく向上する RVI (Rapid Virtualization Indexing) と呼ぶメモリ管理機能を実装した。RVI が提供する仮想化環境でのメモリ管理は、Hypervisor の非常に重い処理のうちの 1 つであった。このため、CPU にページ管理機能を実装することによるパフォーマンス上のメリットはきわめて大きなものとなる。

仮想環境を伴わない (Hypervisor のない) 環境の場合の OS のメモリ管理方法は、図-6 に示すとおりだ。OS 上の仮想メモリアドレスは、OS が管理するページテーブルにより、実際のハードウェアが持つ物理アドレス (マシンアドレス) に変換される。この際、ページテーブルの一部は TLB として CPU 内部にキャッシュされる。

これに対し、Hypervisor 上に複数の仮想 OS を稼働させる仮想化環境では、仮想 OS にマシンアドレスを直接管理させてしまうと、仮想 OS 同士でアドレスの競合が起きてしまう。この競合を回避するため、仮想化環境では、Hypervisor が、シャドウページテーブルという機能を提供している。このシャドウページテーブルは、Hypervisor が管理する仮想 OS 用のページテーブルであるが、その中には、ゲストリニアアドレス→マシンアドレスのマッピング情報が記載されている。Hypervisor はこのシャドウページテーブルを適切に管理することにより、仮想 OS 同士のマシンアドレスの競合を防止し、複数の仮想 OS を 1 つのマシン上で稼働させることを可能としている。

このシャドウページテーブルは非常によくできた機

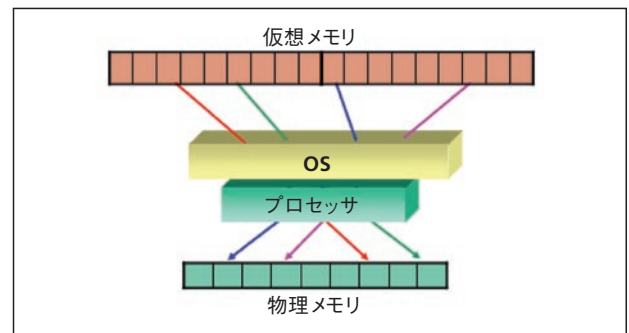


図-6 OS によるメモリ管理

能であるが、Hypervisor は、各仮想 OS が所有するページテーブルの書き換えを管理しその内容をシャドウページテーブルに反映しなければならない。このため、仮想 OS のページテーブル書き換えに伴ってシャドウページテーブルをメンテナンスする必要がある際は、処理を Hypervisor に戻す必要がある。また、仮想 OS 上でページフォルトが起こった場合も同様で、処理を一度 Hypervisor 側に戻す必要がある。このように、シャドウページテーブルによるメモリ管理では、Guest モードから Host モードへの切り換えや、Hypervisor による仮想 OS のページテーブル管理のための大きなオーバーヘッドが生じる。

このオーバーヘッドをなくすため、Barcelona では、RVI を実装し、図-7 で示すようなページ管理方法を提供し、Hypervisor で行っていたメモリ変換処理を CPU で実行できる機能を実現した。

RVI では、仮想 OS のリニアアドレス→仮想 OS 物理アドレスの変換を管理するゲストページテーブル (gPT) は、ゲストの物理メモリ中に存在し、そのアドレス変換情報は、gCR3 に保持される。一方、ゲスト物理アドレス→マシンアドレスへの変換情報を管理するページテーブル (nPT) は、システムの物理メモリに保存され、そのアドレス変換情報は、nCR3 に保持される。また、仮想 OS の稼働により実行されたゲストリニアアドレス→マシンアドレスの変換情報は、TLB にキャッシュされる。Hypervisor は、この TLB を、各仮想 OS にユニークなタグ情報と紐付けて管理している。このため、gPT に対応する各仮想 OS の TLB は、CPU 内に混在することが可能である。RVI の細かい動作に関しては文献 1) をご参照いただきたい。

RVI を CPU へ実装することにより、各仮想 OS は、自分自身で直接 CPU に問い合わせ、リニアアドレス→マシンアドレスへの変換が可能となる。先ほど問題になっていた、ページテーブルの変更や、ページフォルトの際も、Hypervisor に処理を戻す必要がなく、高速な処理が可能となる。言い換えれば、今まで Hypervisor が実行し、負荷の高かったメモリ管理の処理を CPU が肩

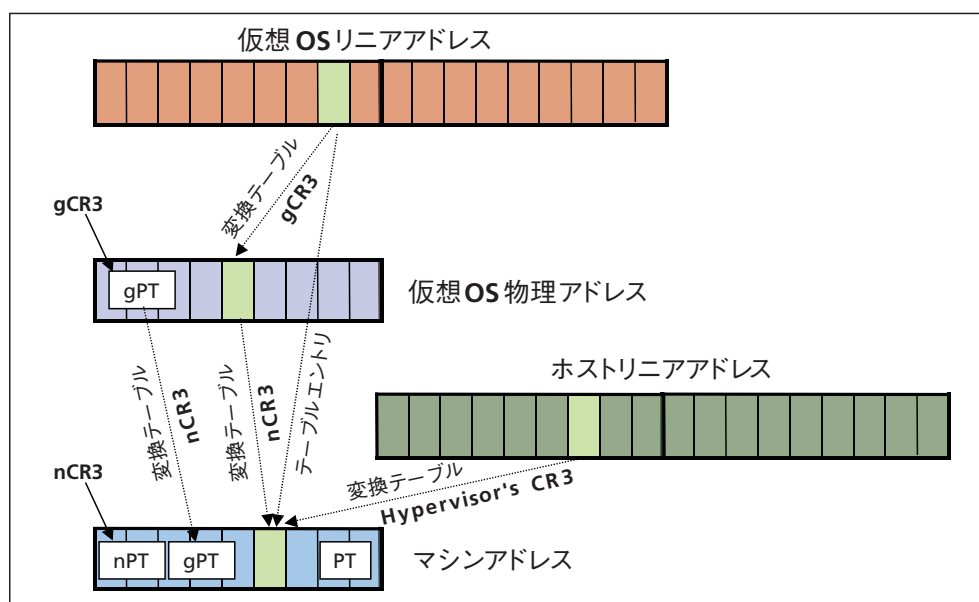


図-7 RVIのメモリ管理

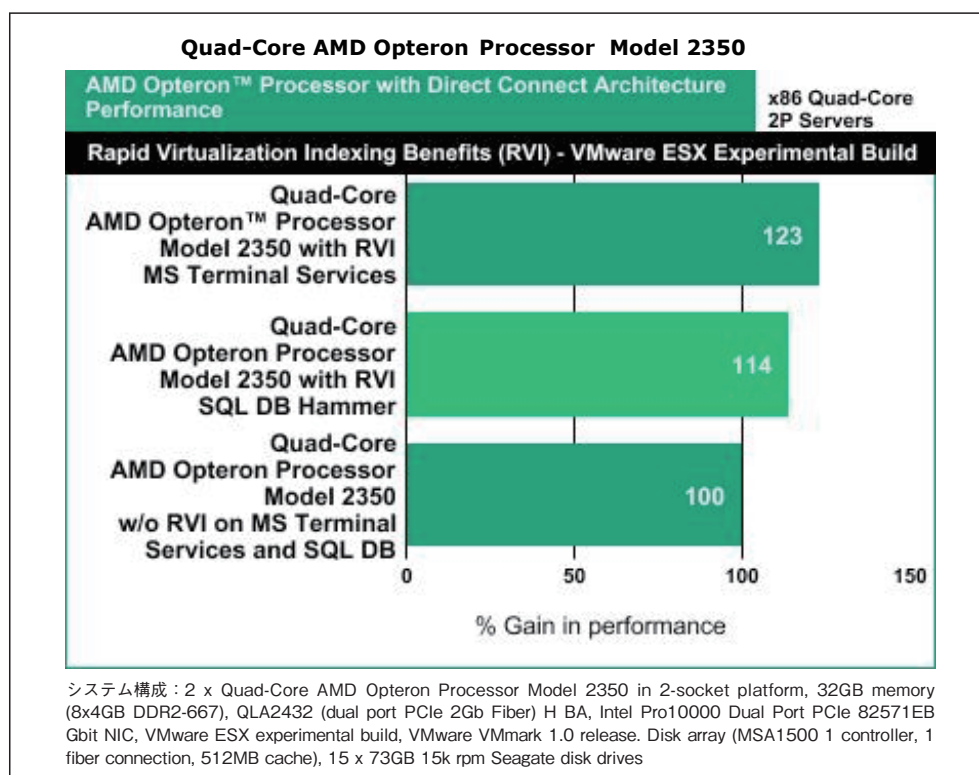


図-8 Rapid Virtualization Indexingによる効果

代わりすることが可能となる。RVIを実装することにより、Hypervisorは各仮想OSのページテーブルの管理やアドレス解決作業から解放される。この効果は非常に大きく、仮想OS上で稼働するアプリケーションのパフォーマンスが大きく改善する。

《 Rapid Virtualization Indexing の効果 》

図-8に、Rapid Virtualization Indexingを利用することによるパフォーマンスの向上をベンチマーク結果として示す（文献2）参照。

このグラフは、一番下に示されているRVIを利用しないときを100とした場合、RVIを利用することによ

り、Microsoft Terminal Service（一番上に示すグラフ）および、SQL Server（中央に示すグラフ）のパフォーマンスがどのくらい向上するかを100に対する比率で示したグラフである。RVIを有効にすることにより、Hypervisorのメモリ管理機能をCPUが実行することが可能となった結果、アプリケーション含めたシステム全体のパフォーマンスが14～23%向上していることが分かる。ここにはデータとして示していないが、RVIを利用することによるパフォーマンスの向上を、Hypervisorの処理部分であるSystem時間と、実際のアプリケーションが稼働する部分であるUser時間に分けて分析すると、User時間はほとんど変わらず、

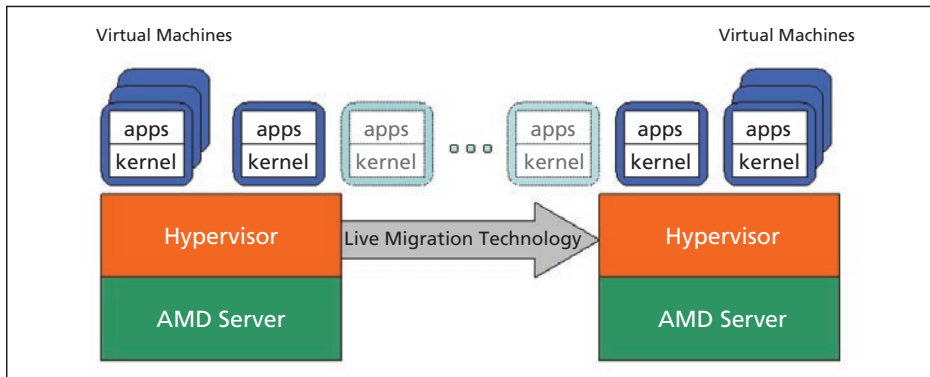


図-9 ライブマイグレーションイメージ

Hypervisor のメモリ管理を含む System 時間が劇的に短縮されていることが分かっている。このことは、RVI を利用することにより、Hypervisor の処理時間（オーバーヘッド）が大幅に改善されていることを示している。

仮想OSのマイグレーション

《ライブマイグレーションの意義》

OS を仮想化するメリットはサーバリソースの有効利用、古い OS の稼働サポートなどがあるが、仮想 OS を稼働させるハードウェアをフレキシブルに変更できる点も大きなメリットである。仮想化の世界では、このような変更処理をマイグレーションと呼ぶ（図-9 参照）。マイグレーションを行うことにより、サーバのリプレイスに伴う負荷の軽減や、サーバ間のリソースの調整、物理サーバのメンテナンス性の向上や、ひいては、ディザスタリカバリエンドユーザベースの大規模なソリューションまで対応可能で、その応用は非常に幅が広い。一言にマイグレーションと言っても、仮想 OS をシャットダウンした後、別のハードウェアで稼働させるコールドマイグレーションと、OS の稼働中に移動するライブマイグレーションの大きく分けて 2 つがある。

このうちライブマイグレーションは、サービスのダウンタイムを最小にすることが可能なソリューションで、VMware 社であれば、Vmotion という機能を利用することにより実現可能だ。この機能を使用すると、仮想 OS は ESX が稼働する複数のサーバ機器の中で、自由に移動することが可能となる。

しかしながらこのライブマイグレーションには、CPU の世代が異なるとサポートされないなど厳しい制約がある。具体的には、CPU に含まれる命令セット（たとえば、SSE3 の実装の有無など）が同一でないとライブマイグレーションはサポートされない。理由は簡単で、たとえば、ライブマイグレーションを受け取る側の CPU が、元の CPU の命令をサポートしていない場合、稼働中のアプリケーションの処理の継続実行が事実上不可能で

あるからだ。

AMD では、2003 年に Opteron プロセッサの提供を介した当初から、幅広い AMD プロセッサ間でのライブマイグレーションのサポートを可能にすることを目指し、仮想化のソフトウェアベンダとの連携を密にしてきた。ここで紹介する AMD-V Extended Migration Technology（文献 3）、4）参照）は、異なる CPU 命令セットを有する CPU 間でのライブマイグレーションの課題を解決する技術である。

《AMD-V Extended Migration Technology》

プロセッサに新機能が搭載されると、その機能の有無はビット情報として CPU 内に記録される。アプリケーションや OS は、x86CPU がサポートする CPUID という命令を実行することにより、このビット情報を得ることが可能で、この情報から CPU がサポートする命令を判別する。たとえば、あるアプリケーションは、CPUID 命令を実行することによって現在稼働中の CPU が SSE3 をサポートすることを理解した上で、SSE3 命令を使って処理を実行する。この SSE3 含め、異なる命令セットを有する CPU 間で仮想 OS のライブマイグレーションを正常に完了するには、ライブマイグレーションを行っている間、Hypervisor が、この CPUID 命令の実行結果をコントロールする。簡単に言うと、サポートされていない命令セットをアプリケーションから隠す必要がある。Hypervisor 側でこのような処置を行うことにより、稼働中にライブマイグレーションが行われた場合においてもアプリケーションの正常な実行完了が可能となる。Hypervisor においてこの機能を実現するため、AMD では、AMD-V Extended Migration Technology という技術を提供している（図-10 参照）。この AMD-V Extended Migration Technology の中で、CPUID Override MSR を提供しており、この機能を利用すると、CPU に実装されている特定の命令セットに対するビット情報をクリアすることが可能となる。具体的には、Hypervisor がこのレジスタを利用することによ

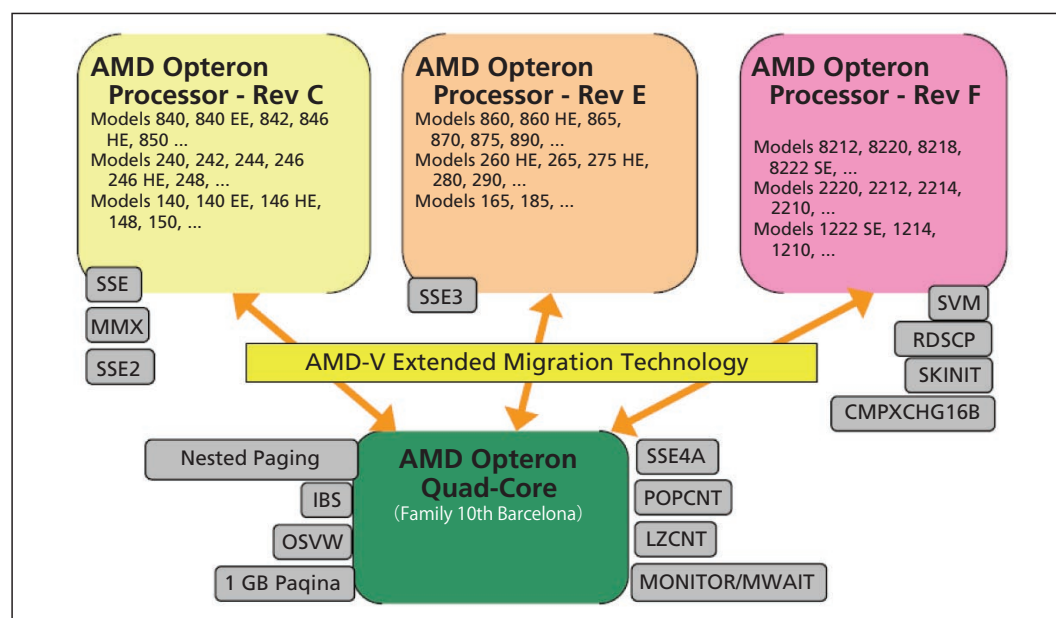


図 -10 AMD-V Extended Migration Technology

り、ライブマイグレーション中に、CPU に実装された特定の命令をアプリケーションから隠すことが可能となる。たとえば Barcelona には実装され、Rev. F には実装されていない SSE4A の命令セットを、Hypervisor がアプリケーションから隠すことが可能となり、その結果、Barcelona と Rev. F 間のライブマイグレーションが実現できる。念のために申し上げておくと、この機能は、CPU に実装されている特定の機能そのものを無効にするのではなく、あくまで、アプリケーション等から、必要に応じて CPU に実装された、ある特有の機能を隠匿する機能である。

AMD はさらに、将来、特定の命令セットを無効にするためのビットを CPU 内に実装予定である。このビットを利用することにより、Hypervisor はライブマイグレーション中に容易に特定の命令を無効にすることが可能となり、ライブマイグレーションを行うサーバファームの中に異なる命令セットを持った CPU が存在しても、ライブマイグレーションのサポートの実現が可能となる。もちろん、アプリケーションの中には、CPU の命令セットを確認する手段として、前記したような手続きを踏まないものもあるかもしれない。このようなアプリケーションの場合は、ここに示した方法は利用できない。ただ、そういったソフトウェアは、エンタープライズ系のアプリケーションにおいてはきわめて少数であることが我々の調査で明らかになっている。

ライブマイグレーションは、サーバ管理者を含めたエンドユーザ側でのサーバサービス管理の柔軟性を大幅に向上する優れたソリューションである。AMD では、命令セットの異なる CPU 間でのライブマイグレーションを可能にするための取り組みを継続的に行っている。このような取り組みが、ライブマイグレーション対応のア

プリケーションコード作成を促し、その結果、仮想化ソフトウェアベンダ側での、より幅広いハードウェア間でのライブマイグレーションのサポートに繋がると考えている。

(編集者追記)

今回は、スタンフォード大学の仮想マシン研究プロジェクトをベースに製品展開を行っている VMware, Inc の技術をご紹介します。VT や NPT、EPT 等現在巷を騒がせているハードウェアの仮想化支援機構がない時代に、どのようにインテルアーキテクチャ上で商用可能なレベルの仮想マシン・アーキテクチャが作られたか、また、これからどのような新技術が実現されるのか、ご説明します。

参考資料

- 1) AMD64 Architecture Programmer's Manual Volume 2: System Programming. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf
- 2) RVI Performance Benchmark (AMD). http://www.amd.com/jp-ja/Processors/ProductInformation/0,,30_118_8796_8800~119097,00.html
- 3) Live Migration With AMD-V Extended Migration Technology. <http://developer.amd.com/assets/Live%20Virtual%20Machine%20Migration%20on%20AMD%20processors.pdf>
- 4) Migrations with VMotion Prevented Due to CPU Mismatch—How to Override Masks. http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1993

(平成 19 年 11 月 30 日受付)

岡野浩史

Hiroshi.Okano@amd.com

日本 AMD で主にソリューション関連のエンジニアを担当しており、最近では、クアッドコア AMD Opteron プロセッサで新たにサポートされた RVI (Rapid Virtualization Indexing) のデモシステムや、Enhanced PowerNow! のデモ環境等を作成しております。皆様に分かりやすく AMD の技術における優位や良さをアピールするというが仕事です。