連載 仮想マシン道しるべ ⑦

仮想マシンの舞台裏（ヴイエムウェア）

VMware Virtualization Technology, Past and Future

名倉丈雄 ヴイエムウェア（株）ストラテジック クライアンス

ヴイエムウェア社設立の背景と現在の市場 ～インフラ全体の仮想化へ～

今でこそ x86 アーキテクチャ・マシンでの仮想化というキーワードは一般的になってきましたが，ヴイエムウェア社が創立された 1998 年頃は，一部の開発者が使用する趣味性の高いテクノロジーであるとの認識が主でした．しかし，現在では実業務のシステムでも使用されていますし，CPU 等ハードウェアの仮想化支援のためのアーキテクチャは年々進化してきています．これは，仮想化が x86 アーキテクチャの世界でも重要な意味を持ちつつあることの現れと言えると思います．

元々仮想マシン（以下 VM）テクノロジーは，40 年ほど前からメインフレームの世界では高価なハードウェア資源を有効活用するために利用されていました．現在の x86 アーキテクチャでの仮想化は，ハードウェア資源の有効活用という従来の目的以外にも，

（a）IT インフラの整理
（b）計算資源の抽象化・ユーティリティ化
（c）アベイラビリティの向上
（d）社会資源の有効活用（省エネルギー）

等の目的で使用されています．

仮想化を行うことで従来の「ハードウェア＝システム」の図式を変え柔軟に運用ができ，完全仮想化（Full Virtualization）によりハードウェアと VM の独立性が実現されることから，ハードウェアと VM のライフサイクルの分離が可能になり，VM はソフトウェアから制御可能であるという特徴を利用して，計算資源をユーティリティ的に使用する（必要な VM リソースを，必要なときに，必要な場所で実体化）ような使用方法もあります．

また，ヴイエムウェアのテクノロジー，VMotion や SoftwareFT 等（後述）を利用しハードウェア障害から VM を守り，アベイラビリティを高めることも可能です．

最後の「省エネ」には驚かれるかもしれません．しかし，ハードウェアを仮想化により有効活用することで，データセンタの多方面のコスト（コンピュータ電源・冷却・設置面積・運用コスト等）を減らすことができ，従来よりも少ないエネルギーで同一のコンピューティングパワーを引き出すことが可能になるので，「省エネ」目的での利用例も増えています．

ヴイエムウェアの VM テクノロジーの歴史も，やはりメインフレームに由来しています．ヴイエムウェアの創設者，チーフ・サイエンティストである Mendel Rosenblum（スタンフォード大学 コンピュータサイエンス 准教授）が 1991 年から開始したオペレーティングシステム研究プロジェクト（Disco Project）は，メインフレームのテクノロジーを x86 アーキテクチャに持ち込む，というコンセプトでした．この研究から x86 アーキテクチャ初の VM monitor が作られ，ヴイエムウェアはこの monitor をベースにした商用製品を開発するために 1998 年に設立され，Workstation（1999 年）・GSX Server（2001 年）・ESX Server（2001 年）とデスクトップ・サーバ仮想化製品をリリースしています．

CPUの仮想化

ヴイエムウェア製品のいわゆるハイパーバイザレイヤは，vmkernel と呼ばれています．vmkernel の特徴として，Intel VT や AMD-V 等 CPU の仮想化支援機構を使用せず（使用するケースもあります，後述），ソフトウェアのみで完全仮想化を行うハイパーバイザである，ということです．

x86 アーキテクチャにおいては，CPU が 4 つの特権レベルを持ち，これを Ring0-3 と言いますが，通常のマルチタスク OS では，ユーザモードとして Ring3，スーパーバイザ（またはカーネル）モードとして Ring0 を使用するのが一般的です（一部，IBM OS/2 等は Ring1 を含んだ 3 モードを使用しますが，これは例外的な OS と言えます）．Ring3 でユーザアプリケーションを動作させ，CPU スケジューリング・IO 制御・メモリページテーブル変更のようなカーネルコードを動作させるために，Ring0 を使用します．当然，Ring3 で動作するアプリケーションコードからは，ハードウェア環境を直接操作するような命令は実行できません．このことによりプロセスの独立性・OS の安定性を確保しています．

では vmkernel はどのように CPU の Ring を使用しているのでしょうか？ 完全仮想化を実現するには，まず各 VM の独立性の確保が最優先課題です．このため **図 -1** のように，vmkernel を Ring0 で動作させ，VM

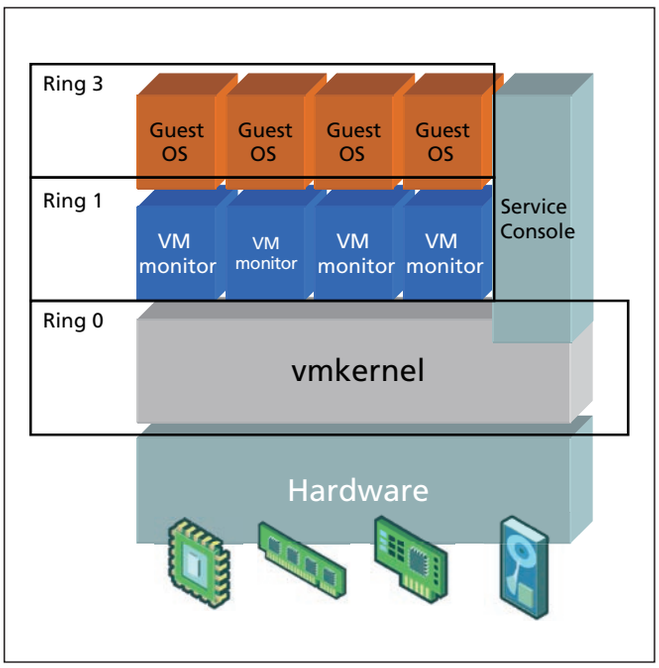

図 -1　vmkernel と VM monitor

内の OS（以下ゲスト OS）カーネルコードを Ring1 上で動作させます．ゲスト OS プロセスのユーザコードは，通常通り Ring3 を使用します．Ring0 で動作することを前提に作られている OS カーネルコードは，もちろんそのままでは Ring1 では動作しません．そこで，各 VM には VM monitor と呼ばれる vmkernel のモニタモジュールがロードされ，Ring0 命令のトラップ～ vmkernel への引き渡し，割り込み・IO 制御，等の中継を行います．

以上が大まかな構造になりますが，これを素直に実装すると，使用に耐えられない速度になります．Ring0 命令のたびに発生する追加の Ring 遷移（Ring3 → Ring1 → Ring0 → Ring1 → Ring3 と，実機に比べ遷移が増えます）が余計な CPU サイクルを消費するからです．ここで vmkernel が持つ BT（Binary Translation）テクノロジーが考え出されました．BT は，リアルタイムに VM 内の CPU インストラクションを先読みし，特定のパターンの Ring0 コードを見つけると，それを等価の Ring1 で動作可能なインストラクションセットに置き換えます．ここで，Translation Cache と呼ばれるインストラクション・マップ・テーブルを使用し，VM monitor が VM 内 CPU のインストラクションをチェック・置換し，可能な限り Ring 遷移を抑えながら VM の動作を継続させます．この BT テクノロジーにより，CPU 仮想化のオーバーヘッドが非常に低い値（実測値 5%程度）に抑えられています [1)]．

現在では，Intel VT・AMD-V 等 CPU に仮想化支援機構が実装されています．vmkernel では，64bitVM を制御する際にこの機能を使用し，32bitVM では BT テクノロジーを使用しています．この決定の理由ですが，まず前提として，CPU 仮想化支援機構も CPU 自身にハイパーバイザモード・通常モードの 2 つを持ち，相互のモード遷移には相当のサイクルコストがかかるという事実があります．BT か，CPU 仮想化支援機構か，どちらを使用するかの調査・開発の段階で，BT が CPU 仮想化支援機能より速度的に優れていたこと，XeonCPU の 64bit モードが，セグメント境界チェック機能を持っておらず，VM のアドレス空間独立性を確保するために 64bitVM では CPU 仮想化支援機構を利用し独立性を確保している，等が主な要因です．

また，vmkernel 自身は 32bit モジュールですが，VM monitor は 32bit・64bit 共に用意され，ゲスト OS の種類により選択されます．vmkernel は 32-64 のアドレス変換を行っており，同一マシンでの 32/64VM の混在使用も可能です．

メモリ管理テクニック

メモリの仮想化に関する一番の問題は，ハイパーバイザがどのように VM の実際のメモリページ確保・解放を検知するか，にあります．メモリは現在のハードウェアの中で，比較的高価な部品です．ゲスト OS がアロケーションするメモリを実際に用意しなければならないとなると，大容量のメモリ搭載が必要になり，結果コストメリットが出せない，マシンあたりの VM 数が増やせない，仮想化の意味がなくなる，という悪循環となります．ヴイエムウェアのアーキテクチャは完全仮想化であり，ゲスト OS になんら変更を加えずに仮想化していますので，この問題を解決する必要があります．

通常 OS は，プロセスの要求に応じ自身の管理するメモリページをプロセスにマッピングします．ところが，プロセスがメモリを解放した際には（OS 自身のみがメモリを使用しているという前提で動いていますので）空きページとしてのマーキング・処理をするのみです．ここでは，ハイパーバイザが，VM からメモリページを他の VM にマップするために再利用するきっかけがありません．

この問題を解決しメモリ利用効率を高めるために考え出されたのが，VM 間ページシェア・バルーンニングです．

VM 間ページシェアとは，単一マシン内での VM 間のメモリページの共有です．複数の VM で同一種類のゲスト OS が動作する場合，同一のカーネルコード・共有ライブラリ・プロセス等が動くことになります．これらのプログラムコードが格納されるページは，ページ単位で同一内容になり，しかもコードのロード時以外の書き込みは発生しないのが通常です．vmkernel は VM か

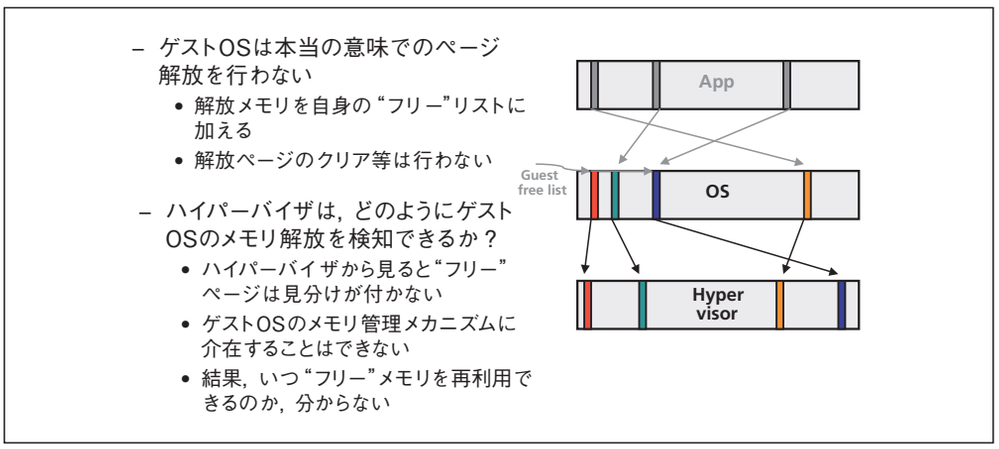

図 -2　メモリページ再利用の問題

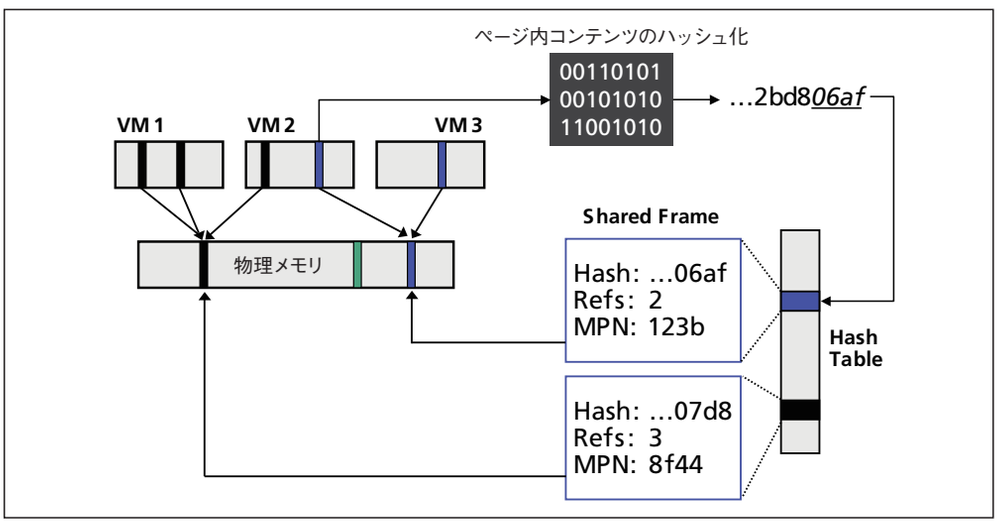

図 -3　VM 間ページシェア

らページアロケーション要求があると，ページを割り当て，同時に**図 -2** にあるハッシュテーブルにエントリを追加します．1 エントリは 16 バイトとコンパクトであり，キーにはページ内のデータをベースにしたハッシュ値を使用します．次に他の VM がページをアロケートしたとき，同様にハッシュ値を算出後，ハッシュテーブルから同一の値を持つエントリを検索します．存在すれば参照カウントをインクリメントし，新ページ要求は既存のページにマップされます．これにより，複数 VM のページが共有されます（**図 -3**）．

ここで，共有しているページに対して VM から書き込みがあると，COW（Copy-On-Write）で新ページを作成し，書き込みを行う VM には，その新ページを再マップします．

vmkernel は，バックグラウンドで 1 スレッドを用意し，ページのチェック・ハッシュテーブルのメンテナンスを VM のスケジューリングとは非同期に実行しています [2)]．

バルーニングとは，VM 間でメモリの解放・確保を促進するためのものです．バルーニングは，物理メモリの空きが少なくなり，VM へのメモリアロケーション要求に応えられなくなる可能性が出てきた段階で稼働します．ここで，VM-A と VM-B があるとします．VM-A は十分にメモリを持ち動作しており，VM-B が新規にメモリ要求を行ったとします．ここで空き物理メモリが足りない場合，vmkernel が VM-A に対してメモリの解放を要求し，VM-A はダミーでメモリのアロケーションを行い（Linux：get_free_page(), Windows：MmAlloccatePagesForMdl() / MmProbeAndLockPages() 等の API を使用），vmkernel にページを返却します．解放されたメモリが VM-B にマッピングされることになります（**図 -4**）．この実装は，VMware Tools という VM 内にインストールする仮想デバイスドライバパッケージに入っています．

将来の仮想ハードウェアではメモリのホットプラグ技術を導入するよう計画していますが，その際には，バルーニングと同等の処理を，メモリホットプラグの形で実装することもあわせて計画されています．

これらのテクノロジーの結果，実際に稼働するすべての VM が必要とするメモリサイズよりも少ないサイズの物理メモリで動作させることができる（メモリのオーバーコミットメント）ようになっています．

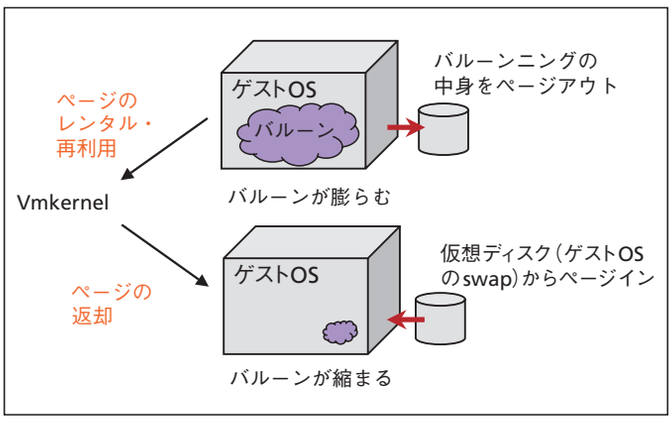

図 -4　バルーンニング

また，最近 CPU で実装が始まった NPT・EPT への対応も進めています．この実現により，メモリ管理のオーバーヘッドの軽減が予想されています．

VMotion

vmkernel には，VMotion という機能があります．これは，動作している VM を無停止で別の物理サーバに移動させるというテクノロジーです．元々 VM と Motion を組み合わせた造語ですが，現在は Live Migration という言葉が一般的に使われ出しています．ここでは，VMotion の動作をご説明します（**図 -5**）．

サーバ A（転送元 ESX）で VM-A が動作しており，他の ESX Server（転送先 ESX）への VMotion 命令が出されると，まず転送元 ESX の VM-A の構成情報を転送先 ESX にコピーします．次に，VM-A のメモリ書き込みを禁止し，以降のメモリ書き込みは別のメモリ領域（ダーティページ）に行い，二重管理を行います．この状態で，書き込み保護されたメモリを，転送先 ESX にコピーし，VM 作成の準備を行います．メモリ転送の完了後，VM-A の動作を停止させ，CPU のレジスタセット・IO ステート等の最低限 VM 動作と同期させなければならないデータを転送先 ESX にコピーし，完了後，転送先 ESX で VM-A を次のインストラクションから再開させます．これで，VM-A が無停止で ESX 間を移動することになります．メモリ・CPU に関してはコピーを行い，VM のディスク領域に関しては，共有ストレージに置いたものを，双方の ESX で使用します．移動はせず，VM のディスク領域に対するロック権限の移譲のみとなります．

ここで，VM メモリのコピー中に発生した書き込みはどうなるか，ですが，ダーティページのインデックスを VM と同時に転送先 ESX にコピーしており，VM の移動後にダーティページが必要になると，オンデマンドで転送元 ESX からコピーされマップされます．また，一定時間後にすべてのダーティページが転送・マップされます．

基本的に VMotion は外部から透過的に行われますが，ネットワークにのみ痕跡が残ります．VMotion を行う

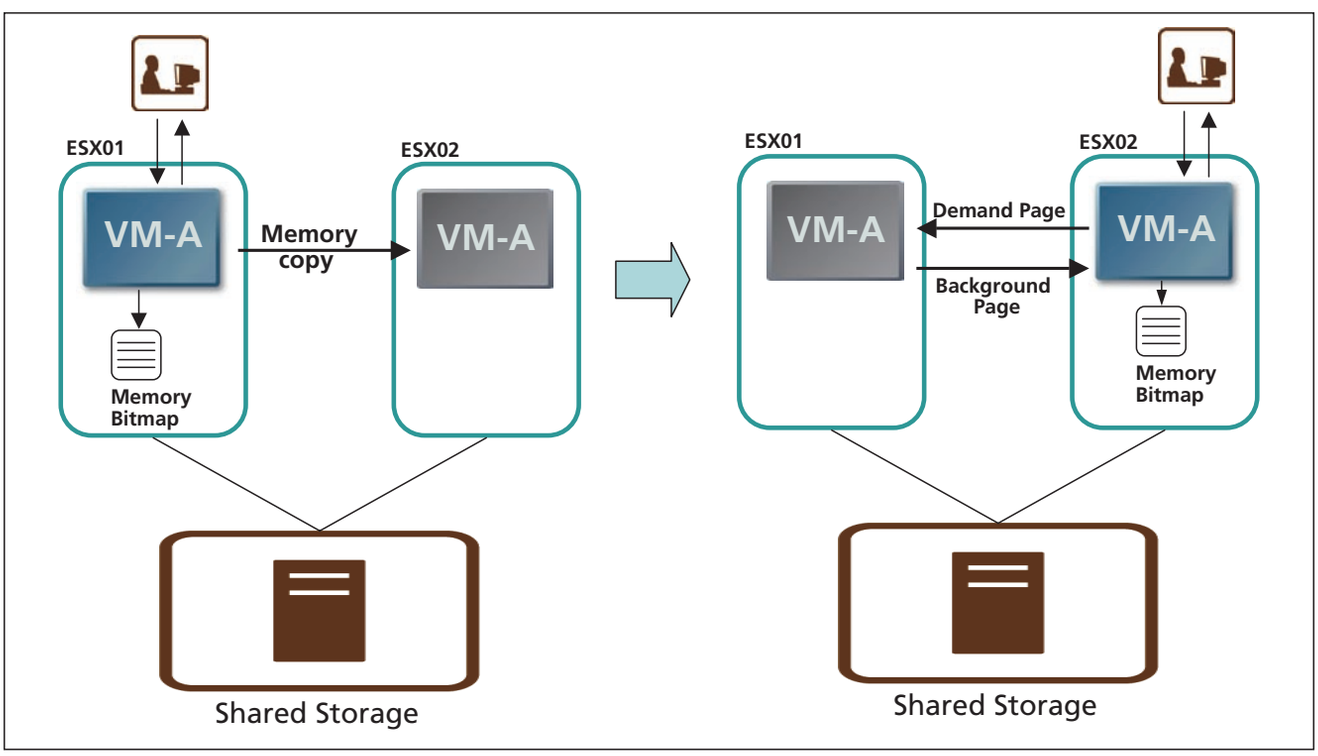

図 -5　VMotion

と，VM の NIC が VMotion 後に別のスイッチ（またはポート）に接続されることになり経路変更が発生します．この経路変更を解決するために，vmkernel が外部スイッチの ARP テーブル更新を促す処理を行っているからです．

VMotion は，その機構上，CPU タイプが同一でなければならないという制限がありましたが，最近 CPU ベンダがこの制限を回避するアーキテクチャを実装し始めており，以降の展開に期待できます．また，VMotion を応用し，サーバ群のリソース負荷状況に応じて自動的に VMotion を行い全体リソースの平準化を行ったり，省エネ対応として，サーバ群の電源使用を最少化するように VMotion を自動実行する，等のソリューションが実装されています [3]．

将来の展望

最後に最近の弊社のトピックスをお話して終わりたいと思います．

- 組み込み ESX

2006 年 9 月に発表をしました ESX の組み込みタイプです．管理インタフェースとして使用していた Linux を排除し，vmkernel と関連モジュールを 32MB 程度のサイズに抑えました．各サーバベンダからサーバ製品に組み込まれた形で出荷されます．仮想化の新しい形として注目が集まっています．

- SoftwareFT

こちらも，2006 年 9 月にテクノロジーデモを行いました．ある VM（プライマリ VM）での動作，CPU のステップ・メモリの IO・各種デバイス IO 等をすべて同期的に，別のサーバ上で動作するセカンダリ VM に転送しながら動作させ，プライマリ VM やホストサーバに障害が発生した際に，セカンダリ VM が処理を継続する，というものです．VMotion と同様に，CPU のインストラクションレベルで同期が取られていますので，シームレスに VM の切り替えが可能になります．市販の x86 サーバを使用し，ソフトウェアのみで FT（Fault Tolerant）環境が実現されることになります．

仮想化を取り巻く状況は常に進化しています．弊社ヴイエムウェアも，仮想化の価値を高め，市場のニーズに対応するために，これからも新しいテクノロジーを提供していきたいと思っています．

（編集者追記）

次回は，基幹サーバへの適用に向けた仮想化技術を，富士通の Itanium2 プロセッサを搭載した基幹サーバ（PRIMEQUEST）向けの Xen 開発を題材に解説する．

富士通は，2005 年末より Xen 開発に参加したが，当時は基幹サーバ向け機能が不十分のため，大規模メモリ管理，I/O 処理高速化，障害分析等の技術開発が必要になった．これらの技術，および，その開発経験より Xen コミュニティの実際の様子を述べる．

参考文献
1）すべてわかる仮想化大全 VMware/Virtual Server, 日経 BP 社（2006）．
2）Waldspurger, C. A. : Memory Resource Management in VMware ESX Server - Fifth Symposium on Operating Systems Design and Implementation (OSDI '02) (Dec. 2002).
3）すべてわかる仮想化大全 2008　実例に学ぶ構築・運用法，日経 BP 社（2007）．

（平成 19 年 12 月 18 日受付）

名倉丈雄
tnagura@vmware.com
Unix 系 OS 開発，PC 用 RDB，第 4 世代言語処理システム，オブジェクト指向 DB 等のシステムソフトウェア企業を経て，ヴイエムウェア（株）設立に参加，現職．