VM（Virtual Machine：仮想マシン：仮想計算機）のその後の発展と現状

Further Virtual Machine Development and its Present Status

梅野英典 熊本大学

はじめに

VM 草創期，VM の発展の話を引き継ぎ，今回は，VM のその後の発展と現状に関して，メインフレームの仮想計算機システム（VMS）のその後の発展と現状について述べ，PC サーバ/WS での VMS についても言及します．

メインフレーム用VMSのその後の発展と現状

《 各社のメインフレーム用VMSの発展 》

VM の発展で述べたように，VMS の高性能化としては，実計算機とほぼ同等の性能を持たせ得る 1 台の V=R VM と，ファームウェアとソフトウェア VMM の改善により，そこそこの性能を持つ常駐 VM という形で技術的に落ち着くかと思われました．しかし，システム開発設備および新旧オンラインシステムの並列運用という実績，すなわち，仮想計算機システムの持つ可用性・柔軟性が世の中一般に評価され，さらにそれを性能面で強化しようという動きが現れました．

これらの動きは，VM の性能要件である実計算機とコンパラブルな性能を持つという必要条件を満たそうとするものです．第三世代の計算機におけるトラップ＆シミュレート方式の VM ではソフトウェアの改善やマイクロコード VMA の導入によって，ある程度の性能向上は見られても，ゲスト OS が多重仮想空間を構成する場合や，オンラインシステムなどの高負荷に対してはそのシミュレーションオーバヘッドが大きくなり，実計算機とコンパラブルな性能を持たせることはできません．この限界を超えるために，各社はハードウェアアーキテクチャとして VM の性能を支援する機構をサポートしました．これを以下に概説します．

富士通社は 1982 年末に AVM/EF という高速 VM 機構をリリースしました [8]．これは，常駐 VM（富士通社は V=D VM と呼びました）に対しても，V=R VM と同様に，実計算機と同等の性能を持たせようとする機構です．当時富士通社と提携していたアムダール社は 1985 年第二四半期に MDF（Multiple Domain Feature）をリリースしました [11]．これは主記憶をいくつかの区画に分割し，各区画（アムダール社はドメインと呼んでいますが実態は常駐 VM です）においてホスト実計算機システムのアーキテクチャを与える機構です．さらに，システムコンソールから各区画のメモリの大きさと装置等のリソースを定義し，VMM はハードウェア内蔵プログラムとして隠されました．

これに刺激されてか，IBM 社も 1988 年末に V=F VM という複数台の優先 VM 方式（実体は常駐 VM 方式）を含む VM/XA SP をリリースしました [14]．このとき，同時に，アムダール社の MDF とほぼ同等の方式である PR/SM（Processor Resources Systems Manager）もリリースしました．IBM 社の場合は，この方式を論理分割方式と呼び，各区画を論理区画（logical partition）と呼んでいます [15), 17)]．日立製作所もまた同年，少し遅れて常駐 VM のハードウェアサポートである VM/EX 機構を含む VMS/ES をリリースしました．さらに，少し遅れて，IBM 社の PR/SM と同様の論理分割方式 MLPF をリリースしました．

これらの機構は，いずれもハードウェアアーキテクチャとして仮想計算機システムをサポートするものであり，なかんずく，複数台の常駐 VM や V=R VM に対して実計算機と同等の性能を与えるためのハードウェア機構です．論理分割方式は，その可用性・柔軟性の高さから，海外におけるメインフレームの利用方式の大勢を占めるようになり，国内でもユーザが増えていきました．この論理分割方式は Unix Open Server にも広がっています．

《 ハードウェア・アーキテクチャにおけるVMS高性能化技術 》

前述で，各社の製品としての VMS の発展経緯を見ましたが，これらは，VM の性能要件である実計算機とコンパラブルな性能を持つという必要条件を満たそうとするものです．このため各社は前述のようにハードウェア・アーキテクチャとして VM の性能を支援する機構をサポートしました．これらは，VM の CPU，メモリ，I/O を VMM によるシミュレーションにより実現するのではなく，センシティブ命令やセンシティブ事象まで含め

て，ハードウェアによる直接実行や直接支援により実現しようとするものです．これらは以下に解説する技術（多くは特許として出願されています）に基づくものです．

これらの多くは，Goldberg, R. P. が 1973 年 5 月に特許出願した Hardware Virtualizer（H.V.）の範疇に入るものと考えられますが，H.V. は数学的なモデルとして高性能な仮想化機構の原理を示したものであり，以下に解説する方式は具体的技術を示すものです．

●センシティブ命令，センシティブ事象の直接実行・直接支援方式

（1）VM モードの導入

この方式は，計算機システムに VM モードを導入し，リソースとそのアクセス方法を VM モードのときとネイティブモード（本来の単一 OS で動作させるモード）のときで別に考え，VM モードのときは VMM の介入なしに VM のリソースに直接アクセスする，というものです．これは仮想計算機システムの基本的な高性能化方式といえるでしょう．考え方としてはこれで良いのですが，この直接アクセスを実現するために具体的にハードウェアは何をすれば良いのでしょうか．まず，VM モードのときは，ネイティブモードのときと同じすべての保護レベル・特権レベルを直接使用可能とします．たとえば，ネイティブモードのとき，0，1，2，3 の 4 つの特権レベルが存在するのであれば，VM モードのときも同じく 0，1，2，3 の 4 つの特権レベルが直接使えるようにします．これにより，ゲスト OS とその配下のアプリケーションがセンシティブ命令を直接実行したとしても特権レベルによる機能差は避けることができます．この考え方の基本は昭和 53 年 2 月に（株）日立より特許出願されています [5]．この考え方・方式は IBM 社の Sie 機構 [9] やインテル社の仮想化技術 Intel VT（Virtualization Technology）[24), 25)] や，AMD 社の仮想化技術 Pacifica[26] にも取り入れられています．

（2）センシティブ命令の直接実行方式

VM モードを導入し，ゲスト OS やその配下のアプリケーションがセンシティブ命令を直接実行してもシステム全体の完全性・信頼性に悪影響を及ぼさず，適切に実行されるためには，ハードウェアはどういう機構をサポートすれば良いでしょうか．センシティブ命令はセンシティブリソースにアクセスし，それを更新したりします．たとえば割込みマスクや制御レジスタなどがそうです．センシティブリソースがゲストに関してもホストに関しても同時にアクティブでなければならないときは，実行中のゲストがそれを直接参照・更新するためには，少なくとも実行中のゲスト用のリソースと，それとは独立なホスト用のリソースが必要です．さらに事象の区別（ゲスト用かホスト用か）と別々の制御論理が必要です．リソースによっては，走行中のゲストのみならず，各ゲスト用に別々に必要なこともあります．しかし仮想化ということは，単に各ゲスト用にハードウェアを用意することではありません．実のハードウェアは 1 つにして，それをいかに多重化し，性能／コスト，さらに機能を上げるかということが重要です．

この方式はセンシティブリソースにより実現方式が異なります．ここでは，リアルタイマとその割込みを例に取ります．リアルタイマの割込みは走行中のゲストのみならず，待機中のゲスト，さらに VMM にとっても必要なものです．元々リアルタイマ割込みの頻度はそれほど高くはなく，したがって，そのホスト割込みは可能として，VMM にリアルタイマ割込みを入れて，VMM により，シミュレーションする方式をとる方が得策です．これによるオーバヘッドはそれほど問題とはなりません．これにより，リアルタイマ自体は，VMS においても 1 つで済みます．

そのかわり，リアルタイマを設定する命令や，割込みマスクを更新するセンシティブ命令を直接実行が可能になるように工夫します．すなわち，割込みは VMM が受け取り，該当 VM で割込みが発生したようにシミュレーションしますが，もし，該当 VM が割込み不可能である場合は VMM が割込み保留とします．そして，該 VM をディスパッチするときに割込み保留フラグを設定します．ハードウェア論理は，この割込み保留フラグと走行中ゲスト用の割込みマスク（これはホスト用とは異なる）を保持し，走行中ゲストが割込みマスクをオンにするセンシティブ命令を発行すると，ハードウェアの論理によってそれが検出され，VMM に制御が移されます．これを，割込みインタセプションといいます．これにより，VMM は該 VM に対して該割込みをシミュレーションすることができます．もし割込み保留要因が存在しないときは，VMM は該割込み保留フラグを設定しません．このときは，上記の割込みインタセプションは発生せず，該センシティブ命令は割込みマスクをどのように更新しても関係なく直接実行されます．このときの走行ゲスト用の割込みマスクは，該ゲストの最近のマスク値を保持するだけであり，ホストシステムの割込みを制御する機能は持ちません．この方式は昭和 54 年 2 月に富士通社より特許出願されています [6]．また，インテル社の仮想化技術においても interrupt-window exiting としてサポートされています [27]．

メインフレームの場合 CPU タイマがありますが，これは，走行中のゲストだけ進めればよく，該プロセッサでは，スケジュール待ちのゲストや VMM 用に進める必要はありません．したがって，待避回復操作により，該プロセッサの CPU タイマは走行中のゲストが使用可

能であり，値の設定命令や，その割込みは直接実行することができます．

制御レジスタについては，ハードウェアは走行中のゲスト用の制御レジスタ（その機能は制限されますが）を持てばよく，制御レジスタに対する参照・更新命令は，多くの場合，直接実行されます．

以下に VM のメモリ，すなわち，主記憶，仮想記憶の直接サポート方式について解説します．

● VMのメモリ，仮想記憶の直接サポート

説明を分かりやすくするために用語を説明します．仮想記憶空間（または単に仮想空間ともいいます）は論理的な記憶空間と見ることができます．また，通常バイト単位にアドレスをつけてアクセスしますので，仮想アドレス空間ともいいます．ホスト実計算機の実記憶をホスト主記憶といいます．それは実記憶空間であり，同様にアドレス付けしますので，実アドレス空間ともいいます．ホスト実計算機上の VMM が構成する仮想空間をホスト仮想空間といいます．ホスト仮想空間も同様にアドレス付けされますので，ホスト仮想アドレス空間といっても同じです．VM にとっての主記憶をゲスト主記憶またはゲスト実記憶といいます．これもまた同様にアドレス付けされますのでゲスト実アドレス空間ともいいます．また，VM にとっての仮想記憶空間をゲスト仮想記憶空間といいます．同様にゲスト仮想アドレス空間ともいいます．解説の便宜上，ネイティブマシン（ネイティブモードで動く実計算機）の仮想記憶とアドレス変換方式の説明から始めます．

(1) ネイティブマシンの仮想記憶とアドレス変換方式

ネイティブマシンは OS の用意するアドレス変換テーブルを用いて仮想アドレスを実アドレスに変換する機能を持ちます．これをネイティブ DAT（Dynamic Address Translation feature）と呼びます．それは仮想アドレス空間から実アドレス空間への一段のアドレス変換となります．

(2) VMモードのときのメモリアドレス階層

ゲスト OS 自身が仮想記憶空間を構成するときは，このネイティブ DAT ではゲスト OS の用意するアドレス変換テーブルだけを用いて，アドレス変換（ゲスト仮想アドレスからホスト実アドレスへの変換）を行うことはできません．この理由を**図 -1** に示します．図 -1 に示すようにゲスト OS 自身はネイティブモードで動作するときと同様に自分自身の仮想記憶空間を構成します（①）．ゲスト OS のアドレス変換テーブル（④）は，このゲスト仮想空間からゲスト実記憶空間（②）へのアドレス変換を与えます．しかし，このゲスト実記憶空間は，ホスト主記憶空間（③）とは異なります．ホスト主記憶空間（③）は VMM が管理しますから，ゲスト実記憶空間②は VMM によって与えられます．これが VMM の構成する仮想空間により与えられるときは，この VM を V=V VM と呼びます．このときは，ホストアドレス変換テーブル（⑤：VMM が構成したもの）により，ゲスト実アドレスからホスト実アドレスへのアドレス変換が与えられます．ネイティブ DAT は，二段アドレス変換機能，すなわち，ゲスト仮想アドレスからゲスト実アドレスへの変換，さらに連続して，ゲスト実アドレスからホスト実アドレスへの変換を行う機能は持っていません．

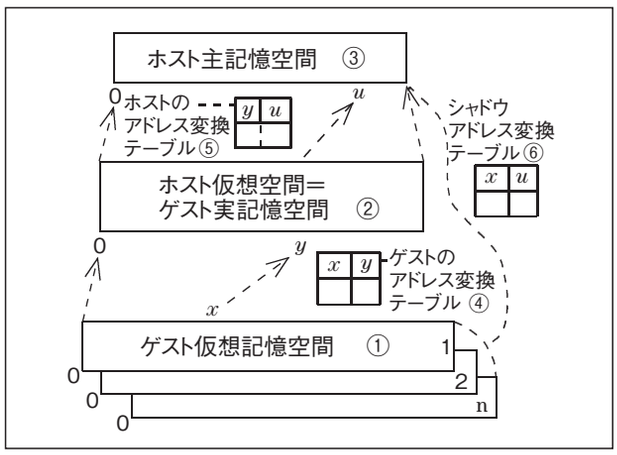

図 -1 V=V VM のメモリアドレス階層

VMM はネイティブ DAT を用いて V=V VM を動かすために，ゲストアドレス変換テーブル④とホストアドレス変換テーブル⑤を合成してシャドウアドレス変換テーブル⑥（シャドウページテーブルともいいます）を構成します．このシャドウアドレス変換テーブルによりゲスト仮想アドレス空間からホスト実アドレス空間への一段のアドレス変換が構成されますのでネイティブ DAT を用いて V=V VM を動かすことができるようになります．しかし，ゲスト OS が多重仮想空間を構成するような OS であるときなどは，このシャドウアドレス変換テーブルの保守・管理が大きなオーバヘッドとなります [12]．

(3) 多段アドレス変換方式

このオーバヘッドを除外するために，シャドウアドレス変換テーブルを不要とする方式が提案されました．これが**図 -2** に示す二段アドレス変換方式です．そこにおいては，まず，ゲスト仮想アドレス x をゲスト実アドレス y に変換するのにゲストアドレス変換テーブル ⓖ(すなわちゲスト DAT）を検索しようとします（①）．ゲストアドレス変換テーブル ⓖ はゲスト実アドレスで構成されていますが，このゲスト実アドレスは VM にとっての実アドレスですが，一般には，ホスト実アドレスではありません．このためホストアドレス変換テーブル ⓗ（すなわちホスト DAT）によりホスト実アドレスに変換して検索しなければなりません（②，③）．検索して

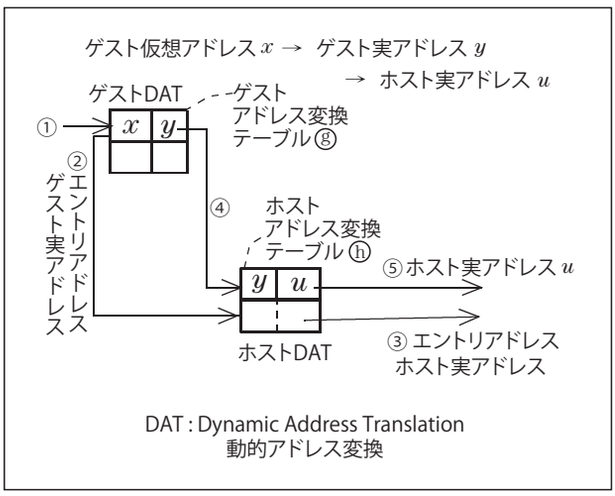

図 -2　二段アドレス変換方式（V=V VM 用 DAT）

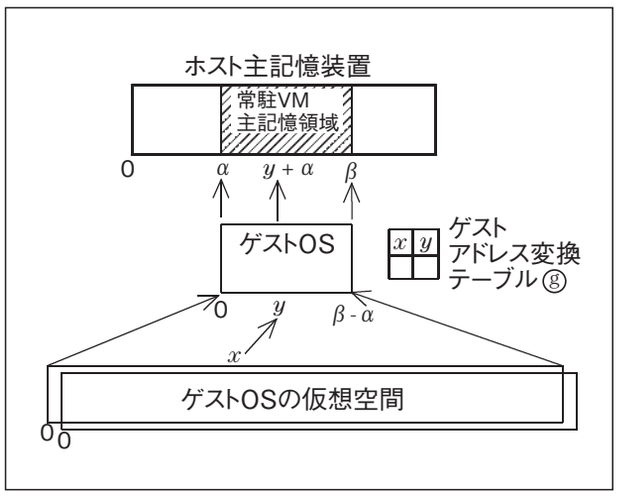

図 -3　常駐 VM のメモリ階層

得られたゲスト実アドレス y を，再びホスト DAT により変換することにより，目標とするホスト実アドレス u を得ることができます（④, ⑤）．すなわち，ゲスト仮想アドレス x をホスト実アドレス u に変換するのに少なくとも，ゲスト DAT 1 回，ホスト DAT 2 回が必要となります．この二段アドレス変換機能を V=V VM 用 DAT と呼びます．これはネイティブ DAT に比べて 3 ～ 4 倍の時間がかかります．この変換遅延時間はアドレス変換バッファ（TLB）にゲスト仮想アドレスとホスト実アドレスを登録することにより解消することができます．

これを一般化した多段のアドレス変換方式は，すでに，(株) 日立より昭和 52 年 8 月に特許出願されています [3]．IBM 社の 1981（昭和 56）年リリースの Sie アシストはこの二段アドレス変換を製品化しています [9]．AMD 社の仮想化技術 Pacifica の仕様書には，Nested Paging Facility として，図 -2 の二段アドレス変換機能が示されています [26]．

また，V=V VMの場合はゲストOSのページングとVMMのページングによる二重ページングに基づく paging anomaly（異常な振舞い）が発生することがありますが，それを防止するアルゴリズムが提案されました [32]．

準仮想化を行う Xen VMM においては，V=V VM をサポートしますが，ゲスト OS がページテーブルエントリを構成するとき，VMM をコールし，VMM によって与えられた実メモリページアドレスを自らのページテーブルエントリに設定することにより，ゲスト OS のアドレス変換テーブル自体が，ゲスト仮想アドレスからホスト実アドレスへのマッピングを与えます [30]．

(4) 常駐VM用アドレス変換方式

筆者らは，常駐 VM に対して，シャドウページテーブルを不要とし，多段アドレス変換方式よりも簡単で性能の良い常駐 VM 用アドレス変換機能を提案しました [4]．これは，ネィティブ DAT にわずかのハードウェア機能を追加するだけで，それと同等性能の，ゲスト仮想アドレスからホスト実アドレスへのアドレス変換機能を実現するものです．常駐 VM のメモリ階層を**図 -3** に示します．この常駐 VM に対するアドレス変換方式を示すハードウェアブロックを**図 -4** に示します．

この方式は，図 -3 のゲストアドレス変換テーブル ⓖ（図 -4 の場合は GST：ゲストセグメントテーブルと GPT：ゲストページテーブルからなる）を直接検索します．そのために GST と GPT 自体がゲスト実アドレスで構成されているため，この常駐 VM にとっての主記憶アドレスの下限 α を加えてホスト実アドレスに変換しながら検索します．最後に，該当 GPT エントリの検索後にも α を加えて，対応するホスト実アドレスを求める機能を持ちます．これはネイティブ DAT と比べて，最後に常駐 VM の下限 α の加算と上限 β のチェックが追加されるだけなので，ほとんど同等の性能となります．この方式は（株）日立より昭和 52 年 5 月と 10 月に特許出願されました [2), 4)]．

(5) TLB内VMID

TLB には，ゲスト仮想アドレスとホスト実アドレスが登録されて，TLB にヒットすればアドレス変換性能はネイティブマシンのときと同じとなります．しかし，TLB ミスのときはメモリ上のアドレス変換テーブルを検索するため，大きな時間がかかります．その時間はネイティブ DAT と常駐 VM 用 DAT ではほとんど同じですが，V=V VM 用 DAT の場合は，ネイティブ DAT と比較して 3 ～ 4 倍の時間がかかります．このため，TLB ミスが 1 ～ 2% であっても平均命令実行時間は 10 ～ 20% 低下します [20]．

TLB のヒット率は計算機性能に大きな影響を持ちます．このため，VMS 運用においても TLB ミス率を低減することが重要です．このため，TLB 内に VMID（VM の識別子）を持つ方式が提案され実装されています．これに

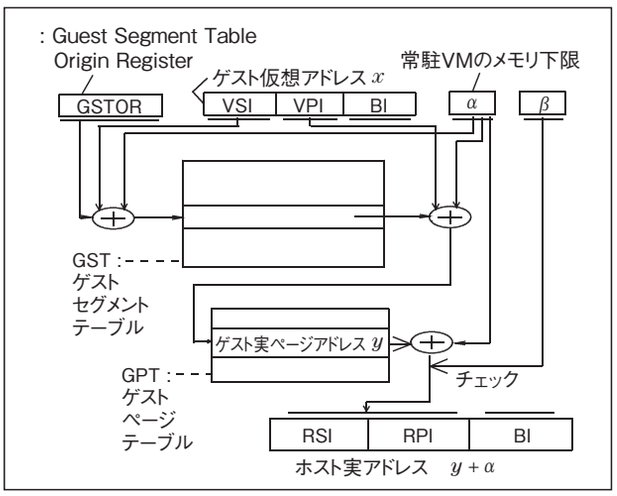

図-4 常駐VM用アドレス変換方式
ゲストアドレス変換テーブル（GST/GPT）を直接検索

より1つのプロセッサにおいて，VMが切り替わったとしてもTLBの内容を無効化する必要はなくなります．また，ゲストOSがTLBエントリを無効化する命令を発行したとしても，該当VMIDを持つTLBエントリを無効化対象とすればよく，他のVMに悪影響を与えないで済みます．AMD社の仮想化技術Pacificaの仕様書によれば，tagged TLBというものがサポートされており，TLBにはゲストエントリとホストエントリを区別する識別子が入ります[26]．これによりVMモードへの出入口でのTLBの無効化を防止することができます．これらにより，VMS運用下においても全体的にTLBヒット率低下を防止することができます．この基本的なアイディアは昭和52年1月に（株）日立より特許出願されています[1]．

● I/O直接実行方式

(1) I/Oシミュレーションオーバヘッド削減の必要性

Goldbergの Hardware Virtualizerにおいても，I/Oに関する仮想化は触れられておらず，その頻度が少ないことからシミュレーションによるとしています．しかし，センシティブ命令（I/O以外）のシミュレーションオーバヘッドを直接実行により削除しても，最後に残るのはI/O命令とI/O割込みのシミュレーションのオーバヘッドです．リアルタイマの割込みシミュレーションは残りますが頻度が小さく問題となりません．しかし，I/O発行頻度の高い負荷，たとえば，高負荷オンライントランザクションなどにおいてはI/Oシミュレーションオーバヘッドは I/Oスループットの低下を招き，大きな性能低下要因となります．

このため，筆者らは先のVMの発展で述べたように，ソフトウェア的にI/Oシミュレーションの高速化方式を提案し，実装してきました．しかしそれも限度があり，高I/O負荷に対して，実計算機と同等性能を持つVMを実現するには，どうしても，ハードウェア・アーキテクチャサポートによるI/O命令，I/O割込みの直接実行方式が必要でした．以下に解説するI/O直接実行方式はメインフレーム技術ですが，PCサーバ/WSにおけるI/O仮想化技術に部分的に，あるいは，さらに発展した形で取り入れられつつあると思います[28]．

(2) I/O命令直接実行方式

先のVMの発展で述べたように，メインフレームのチャネルシステムの機能を利用すれば，サブチャネルの占有化により，チャネルパスや装置のスケジューリングはすべてゲストOSとチャネルシステムの機能にゆだねることができます[13]．それと，次に示す常駐VMにおけるCCW直接実行機能により，常駐VMに対しては，すべてのI/O命令をハードウェアにより直接実行することが可能となります[10)，16)，18)，19)]．

- 常駐VM用CCW直接実行機能

常駐VMの場合は，その主記憶はシステムの実記憶に常駐であり，さらに，ゲスト実アドレス＋α＝ホスト実アドレスとなっています．したがって，チャネルシステムが，このαを加算する機能を持てば，ゲストCCWを直接実行することができます．すなわち，常駐VMを定義し，起動する（logon）するときに，そのVM番号，その下限アドレスαと上限アドレスβをチャネルシステムに登録するようにします．起動するすべての常駐VMに対して，それらを登録します．ゲストOSがI/O命令を発行したときは，カレントなVM番号が，同時にチャネルシステム側へ送信されるようにします．これにより，常駐VMに対してVMMによるCCW変換オーバヘッドを完全に削除することができます．

(3) I/O割込み直接実行方式

さて，I/O命令は直接実行が可能になりましたが，I/O割込みはどうでしょうか．

占有化されたサブチャネルに対しては(2)で述べたようにI/O命令を直接実行することができます．筆者らは，同様に占有化されたサブチャネルに対して，I/O割込みもまた直接実行する方式を提案しました[10)，16)，18)，19)]．そのためには，VMのI/O割込み可能性の判断，VMの割込み優先順位の判断，VMの割込みベクトルの処理等を，ソフトウェアであるVMMの介入なしに，ハードウェアで行う必要があります．しかし，そのために，アクティブなVM台数分のハードウェアをつぎ込むことはコスト上好ましくありません．

そこで以下の方式を提案しました．I/O命令の直接実行と同じように，走行中のVMのI/O割込みはハードウェアにより直接実行し，その他のVM（待ち状態のもの）のI/O割込みについてはソフトウェアであるVMMが介

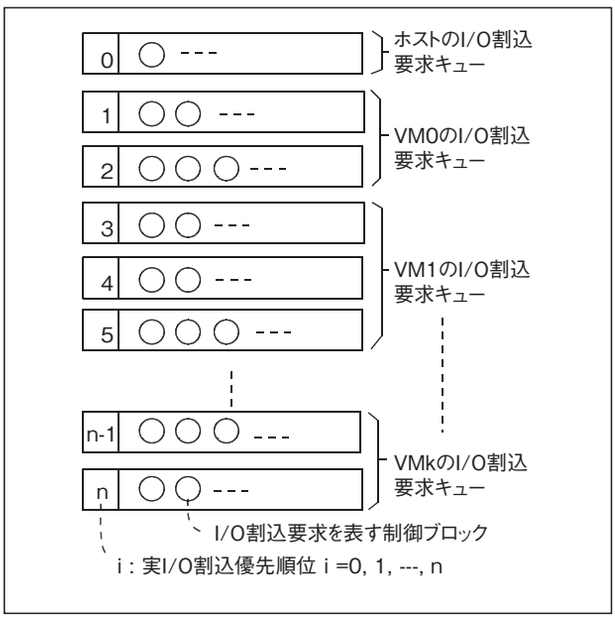

図 -5　VMS における I/O 割込要求キュー

入する余地を残すのは妥当であろうと思われます．そうすれば走行中の VM に反映すべき I/O 割込みが発生しても，その割込みは，該走行中の VM に直接反映され，その走行を継続することができ，無駄な VMM へのモード移行を避けることができます．一般に，VM モードへの出入りには，大きなプロセッサ時間がかかるためなるべくその頻度を減らさなくてはなりません．

メインフレームのチャネルシステムの場合，**図 -5** に示すように，すべての I/O 割込み要求は，優先度 0，1，2，･･･，n に従ってハードウェアが管理するメモリ上にキューイングされます．数字が小さいほど優先度が高いとします．これを実割込み優先順位と呼びます．この優先順位ごとに割込みマスクと保留フラグがあります．この優先順位を各 VM に対して忠実にサポートすると各 VM ごとの優先順位キューが必要となり，複雑な，ソフトウェアのシミュレーションが必要となります．しかし，VMS の運用においては，これを分割占有化させることを考えることができます．たとえば，図 -5 に示すように優先順位 0 を VMM に，1，2 を VM0 に，3，4，5 を VM1 にというように割り当て，分割占有化させる方法です．これにより，実質的に各 VM が使用可能な優先順位の個数は削減されますが，実運用においてはほとんど問題はありません．

さて，このように実割込み優先順位を分割占有化させ，VM 自身が使用する割込み優先順位（これを仮想割込み優先順位と呼びます）と 1 対 1 に対応させることにより，VM の I/O 割込み可能性と優先順位をハードウェアにより直接判断できるようになります．**図 -6** にゲスト用割込み制御論理を示します．図 -6 の場合，仮想割込み優先順位 0，1，2 を実割込み優先順位 3，4，5 に 1 対 1

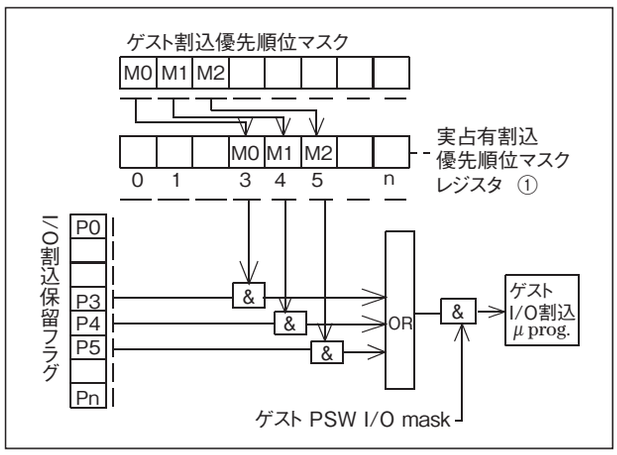

図 -6　ゲスト I/O 割込制御論理

に対応させて，そのマスク値（0 or 1）を仮想割込み優先順位のマスク値と一致させます．割込みマスクレジスタと割込み可能性判断論理は VM 用（といっても現在走行中の VM 用）とホスト（すなわち VMM）用との 2 組を用意します．割込みベクトルもまた走行中の VM 用とホスト用に用意します．ハードウェアとしては，ホスト側が割込み可能であれば，VMM へ割込み，走行中の VM 側が割込み可能であればその VM 側へ割り込むように動作すれば良いのです．VMM は VM を走行させるとき，VMM とハードウェアとのインタフェースである制御ブロックを通して，その VM に分割占有化させた実割込み優先順位に対応する仮想割込み優先順位のマスク値をハードウェアに伝えます．そうすると，VM 走行時にはそのマスク値が VM 用の割込みマスクレジスタ（図 -6 の実占有割込み優先順位マスクレジスタ①）に設定されます．走行中の VM が仮想割込み優先順位のマスク値を動的に変更しても，そのときは，新しいマスク値が VM 用の割込みマスクレジスタ（①）に設定されます．このようにして，ハードウェアにより，走行中 VM の割込み可能性が判断され，割込み可能なときは該 VM の割込みベクトルに従い直接該 VM に割り込むことができるようになります．以上が I/O 割込み直接実行の概要です．

(4) 待ち状態の VM の I/O 割込みの取扱い方法

上記の工夫により，走行中の VM の I/O 割込みはハードウェアにより直接実行可能となりました．さて，待ち状態の VM に対しても，それに反映すべき I/O 割込みは同時に発生しますが，それらは，どのように処理すべきでしょうか．これに対しては，大きくわけて 2 つの方法があります．1 つは，ホスト I/O 割込みとして VMM に割り込む方法，他の 1 つは，ハードウェア側に保留させる方法です [22]．前者は割込みを早く検知して，該当 VM に通知することが可能となりますが，ホスト I/O 割込みのオーバヘッドと VM モードへの出入りオーバ

ヘッドが発生します．後者は，該当ゲストがディスパッチされるときに直接実行されるので，そのようなオーバヘッドは発生しませんが，I/O 割込みの遅延が発生し得ます．

メインフレームにおけるVMの性能

《 メインフレームにおけるVMのCPU性能 》

以上のハードウェアアーキテクチャによる VM 実行支援機能により，メインフレームにおける VM の性能は，高 CPU 負荷に対しても，高 I/O 負荷に対しても，実計算機とほとんど同等の性能，すなわち，実計算機の 95% 以上の性能を達成することができました．

VM の CPU 性能について測定した結果を**図 -7** に示します．

この図で（1）はソフトウェアによる改善と VMA による改善のレベルです [7]．これに対してシャドウページテーブルの更新方法の改善により（2）のレベルまで向上します [12]．これに対して制御命令，すなわち I/O 以外のセンシティブ命令の直接実行により（3）のレベルまで改善されます．I/O 命令・割込み直接実行により Near-Native Performance のレベル（4）にまで到達します [16), 19), 20)]．（1）（2）（3）のレベルまでは V=V VM および常駐 VM に対して適用可能です．高速 CCW 変換は常駐 VM に対してだけです．（4）のレベルまで到達可能なのは V=R VM および常駐 VM だけです．この段階で，まだ VMM によりシミュレーションされるものは，ほとんど実タイマ割込みシミュレーションだけとなります．

《 メインフレームにおけるVMSのシステム性能 》

論理分割方式の VMS がリリースされて以来，そのユーザが増加し，海外のメインフレームは，ほとんどのユーザが論理分割方式の VMS を運用していましたが，そこでの運用形式はサーバ統合形式でした．すなわち，1 台の大型マルチプロセッサメインフレームに，2 ～ 6 台の VM を運用していました．そこで，筆者らは同等の規模を持つベンチマーキングを行い，スループットを測定し，ネイティブマシンとのシステム性能を比較しました [22]．その結果，VMS のシステム性能は，高 I/O 負荷および高 CPU 負荷に対してほとんどネイティブマシンと同等（ネイティブマシンのほぼ 95% 以上）でした．

PCサーバ/WSのVMの性能の考察

PC サーバ /WS においてもハードウェアによる VM 実行支援機能がサポートされています．インテル社の VT-x [24]，VT-i [25]，Directed I/O [28] や，AMD 社の Pacifica [26] 等がそれに相当します．しかしこれらを使用した VMM においては，VM モードへの出入りのオーバヘッドが高く，しかも多くの VM モードへの出入りが発生しするため，ソフトウェアによる VMM よりも良い性能を示すことは，ほとんどないという結果が報告されています [29]．これはハードウェアによる VMM の場合も多くのセンシティブ命令・事象のインタセプションが発生することを示唆しています [27]．

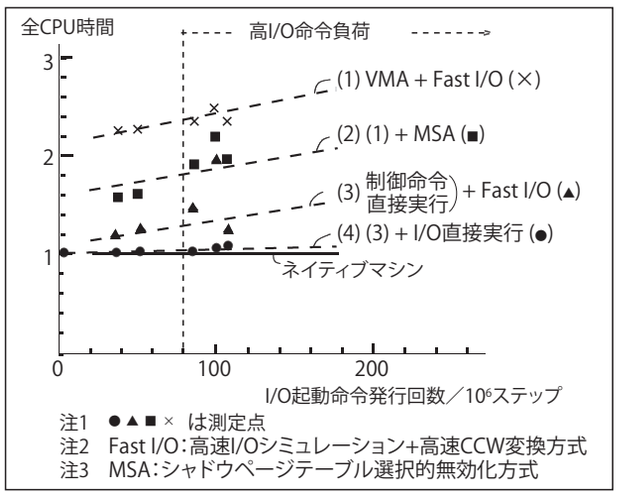

図 -7 VM の CPU 性能

ソフトウェアによる VMM の場合は，アダプティブなバイナリトランスレーションによる方法により，センシティブ命令はあらかじめ安全な命令に置き換えられ，プログラムの基本ブロックが同一特権モードで実行されます．システムセンシティブな管理情報の更新についてはさまざまなメモリトレース機能を設定して VMM に報告するようにしています [21]．しかし同一仮想空間に VMM が存在しなければならないためゲスト OS の使用できるところは制限されます．また，メモリトレース機能に誤りがあるとシステムが暴走しかねません．

I/O 命令に関しては，DMA ハードウェアがメインフレームのチャネルシステムのような機能，すなわち，チャネルパスのスケジューリングや装置をオブジェクトとして表現するサブチャネルの機能は持たないために直接実行はできません．

I/O 割込みにおいては，メインフレームチャネルシステムのように I/O 割込み要求をハードウェア専用メモリ領域に優先順位に従ってキューイングする機能を持たず，各ゲスト用とホスト用に分離することができません．IA32 アーキテクチャにおいては，割込みマスクは 1 つで，外部割込みマスクとして I/O 割込みやタイマ割込みと共通であり，IRQ ごとに割込みマスクが有るわけではありません．IRQ は割込みハンドラを指し示すのみです [23]．したがって，直接実行することはできません．

PC サーバ/WS における VM の性能は，まだまだネイティブマシンに比べて，相当低下することが多いといえます．センシティブ命令・事象は必ずしも直接実行されておらず[27), 29)]，I/O 命令・割込みはシミュレーションされるからです．さらに，常駐 VM 方式は採用しておらず，V=V VM 方式を採用していると思われます．なぜならホスト主記憶は VM 間でページングにより共有していると考えられるからです．この場合シャドウページテーブルを使用すると，その保守オーバヘッドがかかります．また，二段アドレス変換方式を使用するとアドレス変換性能はネイティブマシンに比べて少なくとも 10 ～ 20% 低下します．したがって，PC サーバにおける VM の CPU 性能は，図 -7 における（2）または（3）のレベルにあると推定されます．すなわち高 I/O 負荷に対してはその CPU 実行時間は 2 倍近いものとなるでしょう[31)]．

おわりに

VM のその後の発展と現状についてかなり駆け足で解説しました．ここに述べられた VM 高性能化技術を参考にして，性能や機能そして運用面をよく検討した上で仮想化を導入したいものです．安易に仮想化を導入すると，性能上の大きな問題にぶつかるかもしれません．しかし，Linux 等のオープンソースの活用により，メインフレームよりも，ユーザのシステム運用での柔軟性や多様性が高く広いことから，今後，PC サーバ/WS における仮想化技術は，性能面・機能面で改善されるでしょう．すなわち，性能面ではメインフレーム並となり，機能面・運用面では，VM ネットワークや，VM クラスタの利用技術が広がり，より大きな発展がなされると思われます．

次回は，VM 技術を用いた応用システムに関する近年の研究動向を概説します．特に，セキュリティおよびディペンダビリティの実現を目指した応用システムを中心に説明される予定です．

参考文献

1) 広沢，他 2 名：アドレス変換装置 (VMID in TLB)，(株) 日立，特願昭 52-007734，出願日昭 52.1.28, P1177428.
2) 広沢，他 2 名：記憶制御装置（常駐 VM)，(株）日立，特願昭 52-056386，出願日昭 52.5.18, P1150558.
3) 加藤，他 3 名：アドレス変換装置 (多段アドレス変換)，(株) 日立，特願昭 52-100501，出願日昭 52.8.24.
4) 梅野，池田，源馬：情報処理装置におけるアドレス変換装置（常駐 VM)，(株) 日立，特願昭 52-119050，出願日昭 52.10.5, P1358336.
5) 池田，他 2 名：データ処理装置 (ハードウェアの多重化)，(株) 日立，特願昭 53-15226, 出願日昭 53.2.13.
6) 金田，他 4 名：仮想計算機システム (PSW と保留フラグ)，富士通 (株)，出願昭 54-19440, 出願日昭 54.2.21, P1146366.
7) Umeno, H., Ohmachi, K., Hino, A. and Imura, J. : Development of a High Performance Virtual Machine System and Performance Measurements for it, J. Information Processing, Vol.4, pp.68-78 (July 1981).
8) 富士通 (株)：FACOM ジャーナル，Vol.8, No.9 (1982).
9) Gum, P. H. : System/370 Extended Architecture : Facilities for Virtual Machines, IBM J. Res. Develop., Vol.27, No.6, pp.530-544 (Nov. 1983).
10) 梅野，久保，萩原，佐藤，澤本：I/O 命令実行方法，I/O 割込処理方法およびそれらを用いた計算機システム，（株）日立，出願日昭 59.1.18, P1895108.
11) Amdahl Announces Dual Operating System Option, Computer World (Dec. 3, 1984).
12) Umeno, H. et al.: Reduction of 2-0-Translation Table Maintenance Overhead in a Virtual Machine System, J. Inform. Processing, Vol.8, pp.28-39 (Mar. 1985).
13) IBM System/ 370-XA Principles of Operation SA22-7085.
14) IBM : Virtual Machine/Extended Architecture System Product (VM/XA SP) Release 2, Programming Announcement (June 11, 1987).
15) ジョージ・ヘンリー・ビーン , et al. : データ処理システムの制御方式, 優先権 1987.7.29, IBM PR/SM™ の特許 .
16) Umeno, H. and Tanaka, S. : New Methods for Realizing Plural Near-Native Performance VirtualMachines, IEEE Transactions on Computers, Vol.C-36, No.9, pp.1076-1087 (Sep. 1987).
17) IBM : IBM 3090 Processor Resource/Systems Manager (PR/SM) Feature, IBM ProducTM t Announcement (Feb. 17, 1988).
18) 田中，大築，佐藤，澤本，山縣，渡部，梅野，原口：仮想計算機の入出力実行方式，(株)日立，出願日昭 63.6.30, P2629278.
19) 梅野，井上，田中，池ヶ谷：仮想計算機システムにおける I/O 直接実行方式の提案，第 38 回情報処理学会全国大会論文集 (May 1989).
20) 梅野，久保，田中，井上，阿久津： 仮想計算機システムの高性能化方式，情報処理学会誌，Vol.31, No.12, pp.1665-1680 (Dec. 1990).
21) VMWare, Inc. : Virtualization System Including a Virtual Machine Monitor with a Segmented Architecture, United State Patent, Patent No. : US6,397,242 B1, Filed (Oct. 26, 1998).
22) 梅野，久保，今田：仮想計算機システムにおける論理プロセッサをスケジュールする新方式の開発と評価，情報処理学会論文誌，Vol.44, No.3, P.868-882 (Mar. 2003).
23) Intel : IA-32 Intel® Architecture Software Developer's Manual Volume 3 : System Programming Guide.
24) Intel : Intel® Virtualization Technology Specification for the IA-32 Intel® Architecture, C97063-002 (Apr. 2005).
25) Intel : Intel® Virtualization Technology Specification for Intel® Itanium® Architecture (VT-i), Revision 2.0, 305942-002 (Apr. 2005).
26) AMD : AMD64 Virtualization Codenamed "Pacifica" Technology Secure Virtual Machine Architecture Reference Manual, Revision 3.01, 33047 (May 2005).
27) Uhlig, R., Neiger, G. and Santoni, A. L. et al. : Intel Virtualization Technology, Intel, Computer, IEEE, pp.48-56 (May 2005).
28) Intel : Intel® Virtualization Technology for Directed I/O Architecture Specification, D51397-001 (Feb. 2006).
29) Adams, K. and Agesen, O. : A Comparison of Software and Hardware Techniques for x86 Virtualization, ASPLOS '06, San Jose, California, USA, (Oct. 21-25, 2006).
30) University of Cambridge Computer Laboratory : The Xen™ Virtual Machine Monitor, http://www.cl.cam.ac.uk/research/srg/netos/xen/documentation.html
31) Umeno, H., Teramoto, K., Kawano, M. and Inamasu, H. et al. : Performance Evaluation on Server Consolidation Using Virtual Machines, SICE-ICCAS 2006, BEXCO, Busan, KOREA (Oct. 18-21, 2006).
32) Ohmachi, K., Nishigaki, T. and Takasaki, S. : Analysis of PAWP/VMS : Paging Algorithm to Prevent Double Paging Anomaly in Virtual Machine Systems, J. Inform. Processing, Vol.4, No.2, pp.55-60 (July 1981).

(平成 19 年 9 月 12 日受付)

梅野英典（正会員）
umeno@cs.kumamoto-u.ac.jp
昭和 45 年九州大学理学部数学科卒業．同年 (株) 日立製作所中央研究所入所．昭和 51 年同システム開発研究所．平成 5 年同汎用コンピュータ事業部．平成 8 年博士（理学：東京大学）．平成 10 年熊本大学教授．研究分野：次の性能・機能・信頼性向上方式：オペレーティングシステム，仮想計算機，データベース管理システム．ACM，IEEE Computer Society 各会員．