仮想計算機システムの制御効率を向上するための方式と実験結果†

田口敏夫†† 堀越 彌†† 栗原潤一††

仮想計算機システム (VMS: Virtual Machine System) は1台の実計算機のもとで複数個の仮想計算機を創り出し，別個のOSが同時に走行できるものであり，計算センタ業務とシステム開発・テストなどが共存できる利点がある．このVMSを利用することによって，従来のようにシステム開発のためにセンタ業務を中断させねばならない問題は解消できる．しかし，このように便利な反面，一般の利用形態ではVMSの制御プログラム VMM (Virtual Machine Monitor) が消費する CPU 時間，すなわち CPU オーバヘッドが 300～500%（システム性能が 1/4～1/6 に低下する）と大きく，計算センタ業務の処理能力に重大な影響を及ぼすことになる．このオーバヘッドは，最近一般化しているVMSの高速化機能を使用しても100%程度までしか低減できず，いぜんとして計算センタで一般的に用いるには壁がある．そこで，この点を改善する第1ステップとして，センタ業務を実行するマシン（ホスト・マシン）とシステム開発を行うマシン（テスト・マシン）とを区別し，ホスト・マシンに対するCPUオーバヘッドを大幅に低減させる方式を検討した．これは，①主メモリの管理方式，②シミュレーション処理方式，③割込み処理方式，④CPU 割当て方式，を改良することによって実現している．HITAC M-180 3MB システムのもとでホスト・マシンに VOS 3 を走行させると，CPUオーバヘッドは従来の110%から30%まで低減できる結果が得られている．

1. まえがき

仮想計算機システム (Virtual Machine System: VMS)[1),2)] とは，特殊な制御プログラム (Virtual Machine Monitor: VMM) を使って，1台の計算機システムの中に複数個の仮想計算機 (Virtual Machine: VM) を創り出し，個々の仮想計算機で別個のオペレーティング・システム (Operating System: OS) の動作を可能にするもので，従来の計算機システムとはかなり異なっている．

このVMSは，

(1) システム開発時や，

(2) システム移行時に，

複数のOSを動作させて効果的に計算機システムを利用するためだけでなく，個々の仮想計算機を利用者に割り当てて，

(3) 会話型処理を

効率的に処理するためにも使われている．

しかし，このように便利な反面，(1)(2)の作業を計算センタの業務処理と共存して実行しようとすると，一般の利用形態ではVMSの制御プログラムVMM (Virtual Machine Monitor) が消費するCPU時間，すなわちCPUオーバヘッドが300～500%（システムの性能が 1/4～1/6 に低下する）と大きく，計算センタ業務の処理能力に重大な影響を及ぼすことになる．このオーバヘッドは，最近一般化しているVMSの高速化機能[3)] (Virtual Machine Assist: 特権命令処理のファームウェア化により VMM の走行時間を削減する）を使用しても，100% 程度までしか低減できず，いぜんとして計算センタで一般的に用いるには壁がある．

そこで，筆者らはこの点を改善する第1ステップとして，VMS のもとで計算センタ業務を実行するシステム（これをHost VMという）と，システム開発用のシステム（これを Test VM という）とが共存でき，最小限1MBの主メモリを増設し，これをVMMとTest VMに占有させることにより，従来から存在する1つの Host VM に対する VMM の CPU オーバヘッドを 30% 以下に抑える方式を検討し，その実験システムを開発した．

本稿では，現在までに開発した実験システムの方式と，得られた性能測定結果について報告する．

なお，実験は，

(1) HITAC M-180[4)] システムを利用して，

(2) Host VM に VOS 3[5)] (Virtual-Storage Operating System 3) を走行させて

行った．

† Design and Experiments of a Virtual Machine System by TOSHIO TAGUCHI, HISASHI HORIKOSHI, and JUNICHI KURIHARA (Central Research Laboratories, Hitachi, Ltd.).

†† (株)日立製作所中央研究所

2. CPU オーバヘッドの要因と低減方法

2.1 CPU オーバヘッドの要因

VMS の制御プログラムである VMM (Virtual Machine Monitor) は，主メモリ，CPU (Central Processing Unit: 中央処理装置)，入出力装置などのハードウェア資源をすべて管理しており，特権モードで動作する．一方，VMS 下の OS はすべて非特権モードで動作する構造になっており，OS から発行される特権命令は VMM によってシミュレートされる．さらに，VMM は各 VM のページングやスプーリングの処理も行うので，各 VM 下で走る OS のこれらの処理と重複してくることになる．

したがって，VMS を実現するために VMM が費やす CPU オーバヘッドの要因は，図1の左側のように分類して考えることができる．すなわち，

(1) 各 OS の特権的処理をシミュレートするために要するオーバヘッド，

(2) 各 VM 間のリソースの競合を解決するために要するオーバヘッド，

(3) VMM と OS との多重管理によるオーバヘッドが，VM 下の OS に対して VMM 固有の CPU オーバヘッドとして付け加わり，この CPU オーバヘッドが，各 OS の下で動作するユーザ・プログラムの経過時間に影響を及ぼして性能の低下を招くことになる．

2.2 低減方法

2.1節で述べた VMM の CPU オーバヘッドの各要因に対して，図1の右側に示すような低減方法が，VMM の高速化機能[3]として一般的に考えられている．これらの概要を以下に述べる．

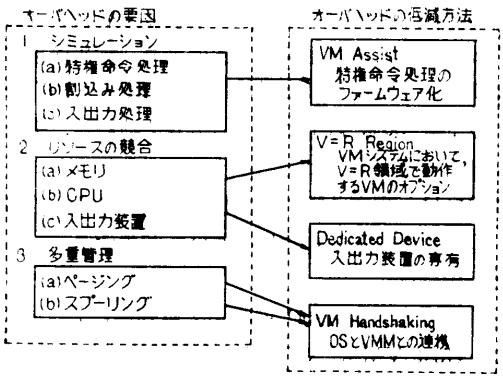


図 1 オーバヘッド要因と低減方法
Fig. 1 CPU overhead under Virtual Machine System vs. Tune-up Method.

(1) 特権命令処理のファームウェア化 (VM Assist)

これは，当社のMシリーズ計算機[4]，あるいは IBM 社の 370 シリーズの計算機[6]で用意されている 33 種の特権命令[4),6)]のうち，出現頻度の高いいくつかの特権命令の VMM によるシミュレーション処理をファームウェア化し，高速に実行するものであり，通常，VM Assist (Virtual Machine Assist) と呼ばれている．

この VM Assist は，VM 下の各 OS と同一の非特権モードで実行され，VMM が介在しないで済むため，単なるファームウェア化以上に CPU オーバヘッドの低減が達成できる．

(2) 主メモリの専有 (V=R Region) 機能[3]

この機能は，特定の VM に対して一定量の主メモリを専有させるものであり，その VM に関しては，メモリ・リソースの競合による CPU オーバヘッドが生じない．このように，主メモリを専有している VM を VMS の V=R 領域 (V=R Region) で動作する VM と呼んでおり，本報告では Host VM ということにする．

(3) 入出力装置の専有 (Dedicated Device) 機能[3]

これは，VMS 下の VM に対して入出力装置を専有させるものであり，これによって，仮想デバイス・アドレスから実デバイス・アドレスへの変換，あるいはその逆の処理が高速化される．通常，Host VM はこの機能を使用することになる．

(4) OS との連携 (VM Handshaking) 機能[7]

これは多重管理による CPU オーバヘッドの増大を防ぐために，OS が VMS の環境で動作していることを意識して，VMM と連携をとりながら処理を進める機能である．すなわち，VMM が創り出す VM とのインタフェースを，ハードウェアの動作仕様にもとづくところから部分的に OS の内部へ移すことによって，VMM の CPU オーバヘッドを低減させようというものである．

以上に述べた高速化機能を Host VM に適用すると，Host VM に対する VMM の CPU オーバヘッドは，バッチ・システムの場合，500% から 100% 程度まで減少するが，通常の計算センタで用いるにはまだ多いと考えられる．

3. 高性能化のための機能

3.1 概要とシステム構成

筆者らが検討した高性能化の方式は，Host VM に

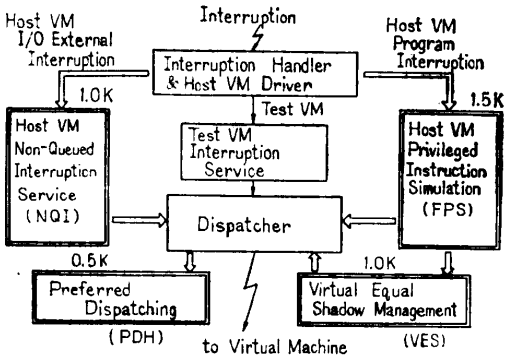


図 2 高性能システムのブロック図
Fig. 2 Block diagram of New feature for performance improvement.

対して2章で述べたオーバヘッドの低減方法に加えて，以下の機能を付加するものである．

(1) VES 機能 (Virtual Equal Shadow Feature)
(2) FPS 機能 (Fast Path Selection Feature)
(3) NQI 機能 (Non-Queued Interruption Feature)
(4) PDH 機能 (Preferred Dispatching to Host VM Feature)

これら4つの機能は，**図2**に示すように各々のモジュール化されており，それぞれ VMM に接続している．したがって，Host VM は VMM に対して，

"自分は Host VM である".

と宣言することによって，上記4つの機能のサービスを受けることができる．

図3は本稿で述べる高速化機能を使用したときの主メモリの割当て方法を表わしたものであり，低位の領域，すなわち図3の S_1～S_3 は Host VM が専有する．したがって，Host VM は 2.2 節で述べたオーバヘッドの低減方法のうち V=R 領域 (V=R Region)[3] の

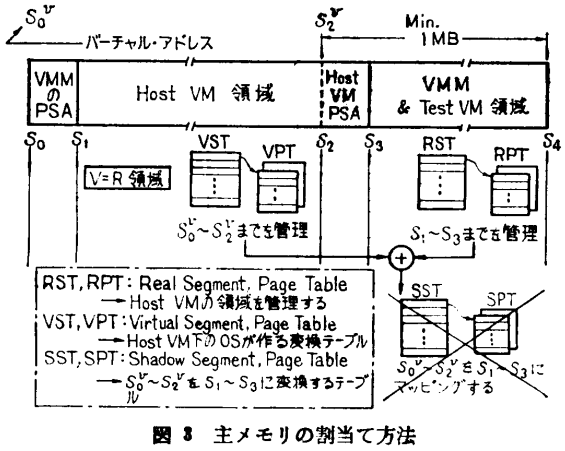


図 3 主メモリの割当て方法
Fig. 3 Storage allocation for High performance Virtual Machine System.

機能を使用することになる．一方，高位の領域，S_3～S_4 は VMM および Test VM の領域ととなり，最小限 1 MB 必要である．なお，最下位の 4 KB (S_0～S_1) は VMM の PSA (Prefixed Storage Area)[4),6)] である．

したがって，Host VM の主メモリ容量が実計算機のときと仮想計算機のときと等しく，かつ VMM の CPU オーバヘッドを 30% 以下に抑えたい場合には，最小限 1 MB の主メモリを増設する必要がある．

Host VM の領域 S_1～S_3 は，VMM が Host VM の領域を管理するためのアドレス変換テーブル[4),6)]，すなわち RST (Real Segment Table) と RPT (Real Page Table) によって管理されている．この RST, RPT は S_3～S_4 内に作成される．なお，図3の SST (Shadow Segment Table), SPT (Shadow Page Table) は，後述の VES 機能によって削除される．

以下筆者らが検討した高速化のための4機能の処理方式と効果を述べる．

3.2 VES 機能

Host VM 下の OS が当社の VOS 2[8)], VOS 3[5)] のようにバーチャル・ストーレジをサポートしている場合には，VMS におけるストーレジのアドレッシングは**図4**に示すように3段階となる．すなわち，VMS の主メモリを第1レベル・ストーレジ (Level 1 Storage) とすると，Host VM が実ストーレジとみなすストーレジは第2レベル・ストーレジ (Level 2 Storage) となり，RST, RPT によって管理される．したがって，Host VM 下の OS は，この第2レベル・ストーレジ上にアドレス変換テーブル VST (Virtual Segment Table), VPT (Virtual Page Table) を作り，ユーザ・プログラムを第3レベル・ストーレジ (Level 3 Storage) 上で動作させる．したがって，第3レベル・ストーレジ上のユーザ・プログラムは2回のアドレス変換を受けることになるが，現在のハードウェアでは1回のアドレス変換しか行えない[4),6)]．

そこで，VMM は図 3, 図4に示すように，2つのアドレス変換テーブル (VST, VPT と RST, RPT) をマージして，シャドウ・テーブル (SST : Shadow Segment Table, SPT : Shadow Page Table)[9)] を作り，第3レベル・ストーレジから第1レベル・ストーレジへの変換が1回で済むようにしている．

このシャドウ・テーブルは，

(1) Host VM の OS がアドレス空間を切換えるために LCTL (Load Control Register) 命令[4),6)]を発

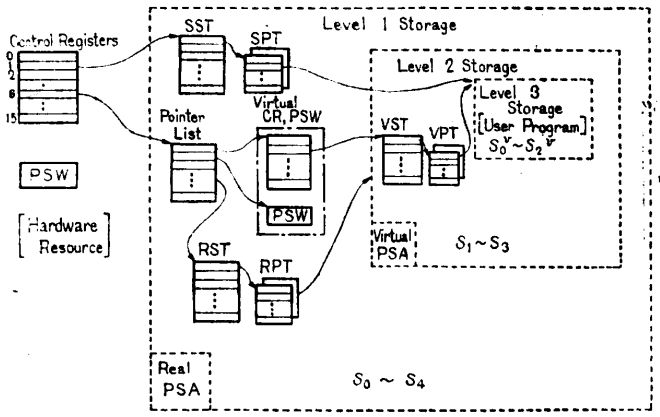


図 4 仮想計算機システムにおけるアドレッシング方法
Fig. 4 Addressing mechanism under VM System.

行したとき,

(2) TLB (Translation Look-aside Buffer) を無効にするために PTLB (Purge TLB) 命令[4),6)]を発行したとき,

(3) 第2レベル・ストーレジ上のページが, VMM によってページ・アウトされたときに,

すべてのエントリが無効にされ (これをシャドウ・テーブルのクリア処理という), 後に, あらためてそのページが参照されたときに,

(4) ページ・フォールトを起しながらシャドウ・テーブルのエントリを作成する処理

(これをシャドウ・テーブルの保守処理という) がなされる.

上記(1)~(4)は, すべて VMM の CPU オーバヘッドとなる.

VES (Virtual Equal Shadow) 機能は, 上記(1)~(4)のCPUオーバヘッドを減少させるために, 図 3, 図 4 のシャドウ・テーブルを削除し, Host VM が作る VST, VPT をシャドウ・テーブルとみなして動作させる機能である. これは図3に示したように, Host VM は VMS の V=R 領域に存在しているため, PSA を除いて第1レベル・ストーレジと第2レベル・ストーレジは1対1の対応がとれているためである. したがって, Host VM の PSA に関しては, VPT の第0エントリを図3の S_2 の値で書替えることで正しいマッピングが成立する. これによって, OS がアドレス変換モードで動作しているときに, この VST, VPT を用いることができる. なお, OS がアドレス変換モードで動作しない場合には, Host VM の領域を管理している RST, RPT を用いてアドレス変換モードで動作させることになる.

この VES 機能は, 上記(1)~(4)の CPU オーバヘッドを低減するだけでなく, Host VM で発生したセグメント・フォールト, ページ・フォールト[4),6)]のシミュレーション処理の高速化にも効果がある.

3.3 FPS 機能

この機能は, Host VM に関する特権命令のシミュレーション処理と, 入出力割込み処理を高速化するものである. これは以下の条件のもとで可能である.

(1) 図3に示したように, Host VM は VMS の V=R 領域に常駐している.

(2) Host VM は入出力装置の専有機能(Dedicated Device Option) を使用している.

すなわち, 図5に示すように, VMM は特権命令のシミュレーション処理が必要となったとき, 要求元が Host VM であるかどうかを調べる. 要求元が Host VM であるならば, VMM は図2に示した Host VM 専用の特権シミュレーションを行うモジュールへ制御を渡す. そのモジュールでは,

(1) 命令コードの解釈,

(2) シミュレーション処理

を高速に行った後, Host VM 専用のディスパッチャを介して Host VM に制御を返す.

シミュレーション処理の高速化は, 次のようにして実現している.

(1) シミュレーションの対象となる特権命令のオペランドは, 常に第1レベル・ストーレジ内に存在しているため, VMM と Host VM との間で無駄な空間の切換えを行わない.

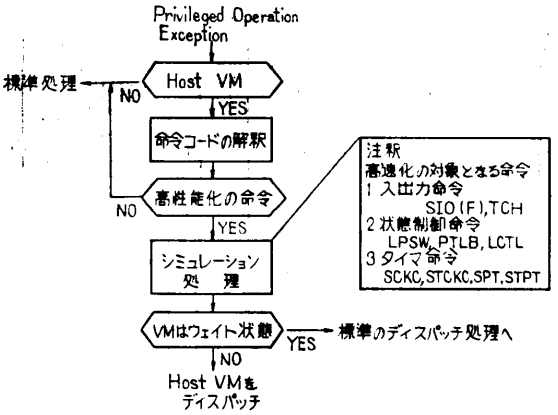


図 5 FPS 機能の処理フロー
Fig. 5 Processing flow using Fast Path Selection (FPS) Feature.

(2) 汎用的な処理ルートを介さずに，Host VM 専用の処理ルートを設けることによって，シミュレーション処理のステップ数を減少させる．

(3) シミュレーション処理の単位を大きくすることによって，サブルーチン間のリンケージを皆無にする．

(4) 入出力命令のシミュレーション処理では，入出力装置の仮想アドレスと実アドレスの変換テーブルを新設することによって，各制御ブロックのロケート処理を高速化する．

なお，この FPS (Fast Path Selection) 機能の対象となる特権命令および処理は，図5に示した，

(1) 入出力命令，
(2) 状態制御命令，
(3) タイマ命令，
(4) 入出力割込み処理

であり，VMM の処理ステップ数は，従来に比べて1/2～2/3まで減少した．

3.4 NQI 機能

この機能の特徴は，

"Host VM に関する割込みは，最優先に処理して，必ず Host VM をディスパッチする．"

ことである．したがって，割込み発生時に Test VM が走行していても，制御は Host VM へ移ることになる．

この機能のサービスを受ける割込みは，

(1) 外部割込み (External Interruption)
(2) 入出力割込み (I/O Interruption)

であり，**図6**は NQI (Non-Queued Interruption) 機能の基本概念を表わしたものである．すなわち，従来の割込み処理[9]では，"割込み情報ブロック" をディスパッチャの待ち行列に登録し，後に VMM のディスパッチャがこの "割込み情報ブロック" を取出して，各 VM の状態記述テーブル (VMBLOK: Virtual Machine Control Block) に割込み情報を反映する．

一方，この NQI 機能では，Host VM に関する割込みであれば，"割込み情報ブロック" を待ち行列に登録することなく，直ちに Host VM の状態記述テーブルに割込み情報を反映し，必ず Host VM をディスパッチすることになる．

以上をまとめると，この機能の効果は，

(1) VMM のディスパッチャの処理ステップ数が削減できること，
(2) 割込み処理が高速化できること，
(3) Host VM に関する割込みが発生したならば，必ず Host VM がディスパッチされること

である．

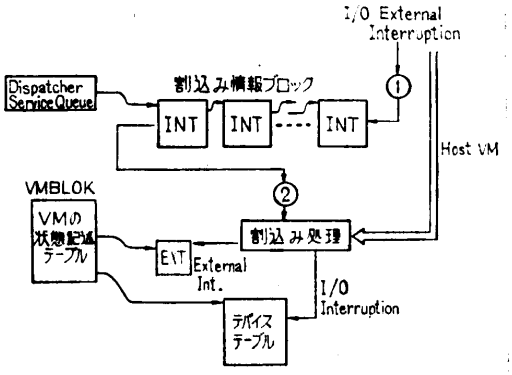


図 6 NQI 機能の概念図
Fig. 6 Concept of Non-Queued Interruption (NQI) Feature.

3.5 PDH 機能

この機能は，

"Host VM がウェイトしない限り，VMM は Host VM をディスパッチする．"

ことを原則としている．したがって，PDH (Preferred Dispatching) 機能では VMM のスケジュリング方法[9]を変更することになる．

VMS においては，各 VM の走行状態を監視するために，**図7**に示すような管理方法を採用している．すなわち，各 VM は図7のどれか1つの状態に入っており，大別して以下の2つのグループに分類される．

(1) IN-QUEUE グループ

VM が次の状態になったときに，このグループに入る．

(a) 走行可能な状態であり，VMM から CPU 資源を割当ててもらい，すでに走行しているとき．
(b) 走行可能であるが，CPU 資源の割当てを待っ

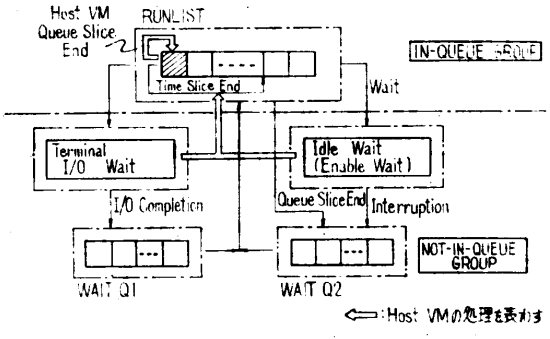


図 7 仮想計算機システムのスケジュリング方法
Fig. 7 Virtual Machine Scheduling Flow.

ているとき.

(c) 走行中に入出力命令を発行して，VMMの入出力サービスを待っているとき.

上記のうち(a)(b)は RUNLIST に登録されているが，(c)は RUNLIST には登録されていない．なお，このグループ内に存在しているときに消費できる CPU 時間（これを Queue Time Slice という．以降 QSL と略す）は決っており，この QSL を使いきると(2)の NOT-IN-QUEUE グループへ移る．また，RUNLIST 内では RUNLIST の先頭に存在できる経過時間（これを Time Slice という．以降 TSL と略す）を使いきると，他の VM を走行させるために RUNLIST の後に再登録される.

(2) NOT-IN-QUEUE グループ

VM が次の状態になったとき，このグループに入る.

(a) ウェイト・ビットがオンの LPSW (Load Program Status Word) 命令[4),6)]を実行したとき.

(b) 端末の入出力を実行したとき.

(c) IN-QUEUE グループ内で消費できる CPU 時間を使いきったとき.

(d) VMS のコマンドを実行したとき.

通常，このグループから(1)の IN-QUEUE グループへ戻る前に，図7の WAITQ 1, WAITQ 2 に登録される.

この PDH (Preferred Dispatching to Host VM) 機能は，Host VM に対して図7の太線で示すように WAITQ 1, WAITQ 2に登録することなくサービス処理を行う.

具体的には，PDH 機能によって以下のような処理がなされる.

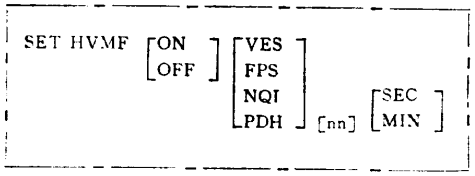
(1) Host VM がウェイトしない限り，Host VM を IN-QUEUE グループ内に存在させる.

(2) QSL を使いきっても，Host VM が消費した CPU 時間を計算し，再び RUNLIST の先頭に登録する.

(3) Host VM の TSL は Test VM よりも大きいため，RUNLIST の先頭に存在する時間が Test VM よりも長くなるようにする.

したがって，Test VM は Host VM がウェイト状態になったときのみ走行できることになる.

この PDH 機能は CPU 資源を Host VM に優先して割当てるものであるが，Test VM のサービスを多くしたい場合には，図8に示すコマンドを投入し，



VES: Virtual Equal Shadow Feature
FPS: Fast Path Selection Feature
NQI: Non-Queued Interruption Feature
PDH: Preferred Dispatching Feature

図 8 高性能化のためのコマンド形式
Fig. 8 Command format of Highly-performance option.

Host VM の TSL の値を動的に変更することができるようになっている.

4. 実験結果

筆者らは，3章で述べた高性能化のための機能の効果を把握するために，実験システムを開発した．この章では，バッチ・システムを例として，この実験システムの性能測定結果を種々の角度から検討する.

なお，この実験システムの開発ステップ数は，図2に示したようにアセンブラ言語で4kステップであり，さらに図8のコマンド処理や VMS のイニシャライズ処理，ログ・オン処理などを含めると合計7kステップで実現できた.

4.1 ベンチマーク・ジョブ

バッチ・システム用の負荷として，30題のジョブよりなるベンチマーク・ジョブを設定した．これらのジョブは，表1に示す特性を有しており，

(1) 技術計算，

表 1 ベンチマーク・ジョブの特性
Table 1 Characteristics of Benchmark Job.

	項目		値
1	ジョブ数		30題
2	課金の対象となる CPU時間	最大	162秒
		最小	2秒
		平均	39秒
3	ジョブ当りの入出力回数	最大	8,776回
		最小	57回
4	ジョブ当りのメモリ使用量	最大	1,220 KB
		最小	200 KB
		平均	420 KB
5	使用する磁気テープ台数		6台
6	磁気ディスク台数	システム・ファイル用	3台
		ユーザ・ファイル用	3台

表 2 実計算機システムでの性能
Table 2 Performance data on Native Mode. (Bare Machine)

	項目		値
1	中央処理装置		HITAC M-180
2	主記憶装置の容量		2 MB
3	オペレーティング・システム	システム名	VOS 3*
		起動したイニシェタ数	6
4	性能データ	CPU 時間	19.5分
		経過時間	38.0分

* Virtual-Storage Operating System 3

(2) ファイル処理,
(3) X-Y プロッタ処理,
などの各ジョブを含む平均 420 KB のプログラム群からなり, M-180 を使った計算センタの標準像の1つと考えられるものである.

表1に示したベンチマーク・ジョブを表2に示すような条件で実行すると,
(1) CPU 時間: 19.5(分)
(2) 経過時間: 38.0(分)
で終了するものである. なお, これらの測定にはハードウェア・モニタ[4]を用いている.

4.2 高性能化の機能を用いないときの性能

4.1節のベンチマーク・ジョブを, VMS のもとで3章で述べた高性能化の機能を用いずに実行したときの, VMM の CPU オーバヘッドを検討する. なお, 2.2 節で述べた一般的に考えられている高速化機能は用いることにする. それらは,
(1) 特権命令処理のファームウェア化(VM Assist),
(2) V=R 領域の機能,
(3) 入出力装置の専有機能,
(4) OS との連携機能
である. したがって, Host VM に対しては 2 MB の主メモリを専有させ, 残りの 1 MB は VMM, および Test VM が使用し, 合計 3 MB の主メモリのもとで VMS が動作することになる. なお, Host VM の主メモリ容量は表2で示した実計算機のときの容量と同一である.

図9の上段は, 上記の条件でベンチマーク・ジョブを実行したときの VMM の CPU オーバヘッドを表わしている. 図9より, VMM の CPU オーバヘッドは 110% であり, 特権命令のシミュレーション処理に 75% を要していることが分かる. また, 非特権モードで実行される VM Assist の CPU オーバヘッドは 25% であり, シャドウ・テーブルの保守処理に 18% を要していることが分かる.

4.3 高性能化の効果

図9の下段は, 3章で述べた高性能化の機能を付加したときの VMM の CPU オーバヘッドを表わしたものである. なお, 主メモリの割当て方法は 4.2 節と同様であり, 図3に示したとおりである.

図9より, 3章で述べた高性能化のための4機能を付加すると, VMM の CPU オーバヘッドは 30% まで減少し, 従来に比べて 3.7 倍の性能向上となった. このように, VMM の CPU オーバヘッドが 110% から 30% まで減少したのは, 以下の効果が大きいものと考えられる.

(1) VES 機能によって, PTLB 命令, LCTL 命令のシミュレーション処理が, 従来に比べて約6倍に高速化できたこと.
(2) VES 機能によってシャドウ・テーブルが削除されたために, VM Assist によるシャドウ・テーブルの保守処理が削減できたこと.
(3) FPS機能によって, 入出力命令などのシミュレーション処理が約2倍に高速化できたこと.
(4) NQI 機能, PDH 機能によって, 割込み処理が約 1.7 倍に高速化できたこと.

次に経過時間について検討する. この経過時間については, 次式に示すような RBT (Relative Batch Throughput) とよばれる評価指標を導入する.

$$RBT=\frac{T_N}{T_V} \qquad (1)$$

T_N: 実計算機で実行したときの経過時間
T_V: 仮想計算機で実行したときの経過時間

本稿で述べた高性能化の機能を用いないときの RBT は 0.69 であるが, 高性能化の機能を付加すると 0.97 となり, 実計算機のもとで実行した場合とほと

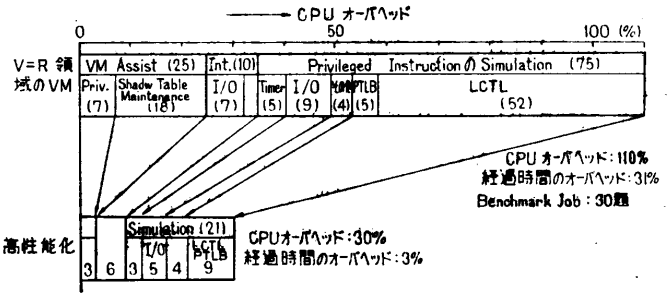


図 9 高性能化の効果
Fig. 9 Effectiveness of performance improvements.

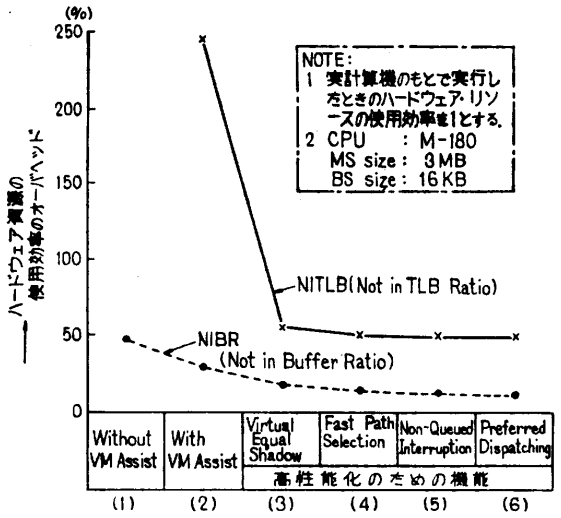


図 10 ハードウェア資源の使用効率
Fig. 10 Utilization overhead of hardware resource under Virtual Machine System.

んど差がなくなる．このようにRBTが0.97程度（経過時間のオーバヘッドが3%）であれば，計算センタで一般的に用いても処理能力に重大な影響を及ぼさないものと思われる．なお，Host VM をV=R領域で動作させないときの RBT は0.49であった．

4.4 ハードウェア資源の使用効率

図10は，ベンチマーク・ジョブを VMS のもとで実行させた場合に，ハードウェア資源の使用効率が実計算機で実行したときに比べて，どの程度悪くなっているかを表わしたものである．図 10 において(1)(2)は V=R 領域で実行させたものであり，(3)〜(6)は筆者らが検討した高性能化の機能を順次付加したときのものである．

(1) NITLB (Not in TLB Ratio)

V=R 領域で実行したときの NITLB は，実計算機のときに比べて246% (3.46倍) と悪いが，高性能化の機能を付加すると 49% のオーバヘッドまで減少する．しかし，実計算機の場合に比べてまだ悪く，今後は，この NITLB を実計算機の値に近づけるための方式を検討する必要がある．

(2) NIBR (Not in Buffer Ratio)

Host VM を V=R 領域で実行したときの NIBR は，実計算機のときに比べて 30% のオーバヘッドであるが，高性能化の機能を付加すると 12% まで減少する．

この NIBR は，バッファ・ストーレジの容量を増加させることによって，実計算機との差は減るものと思われる．

5. むすび

以上，仮想計算機システムにおいて，特定の VM (Host VM) に対する VMM の CPU オーバヘッドを30% 以下に低減させるための方式と，実験システムについて述べた．この実験システムを VOS 3[5] (Virtual-Storage Operating System 3)で制御されるバッチ・システムに適用すると，VMM の CPU オーバヘッドは現状の 110% から 30% へと減少した．

この実験システムは，従来のシステムに最小限 1 MB の主メモリを増設し，その増設された主メモリ内で VMM と Test VM を走行させ，従来から存在する主メモリ部で走行する1個の Host VM に対する VMM の CPU オーバヘッドを低減させたものであるが，今後，

(1) Host VM が複数個存在するときにも CPU オーバヘッドを低減し，また，

(2) Test VM に対する CPU オーバヘッドをさらに低減するための

第1段階として意味があるものと考える．また，これだけでも，一般のバッチ・センタの用途に対しては十分効果を発揮できるものである．

最後に，本研究の遂行にあたって，開発の機会を与えていただいた(株)日立製作所中央研究所堤副所長，大西第7部長，ならびに御協力をいただいた(株)日立製作所ソフトウェア工場，山本，片岡主任技師，同神奈川工場，井上副部長，小高主任技師，同システム開発研究所石原主任研究員，大町研究員，および実験の便宜を図っていただいた中央研究所高徳元計算センタ長，桑原計算センタ長に感謝いたします．また，中央研究所の長島，宮本，吉住の各研究員には直接御指導を受けたので，ここに感謝の意を表します．

参考文献

1) Goldberg, R. P.: Survey of Virtual Machine Research, Computer, pp. 34-45 (June, 1974).
2) IBM 社: VM/370 Introduction, IBM 社マニュアル, GC 20-1800.
3) IBM 社: VM/370 System Programmer's Guide, IBM 社マニュアル, GC 20-1807.
4) 日立製作所: M 170/M 180 処理装置解説, 日立マニュアル, 8080-2-001.
5) 日立製作所: VOS 3 概説, 日立マニュアル, 8090-3-001.
6) IBM 社: IBM System/370 Principles of Opera-

tion, IBM 社マニュアル, GA 22-7000.
7) Rothstein, P. J.: VM Handshaking for OS and OS/VS, GUIDE 42 Proceedings, pp. 1219-1240 (1976).
8) 日立製作所: VOS 2 概説, 日立マニュアル, 8080-3-001.
9) IBM 社: VM/370 System Logic and Problem Determination Guide, IBM 社マニュアル, SY 20-0885.

(昭和 53 年 10 月 19 日受付)
(昭和 54 年 1 月 18 日採録)