# ハイパーバイザの作り方〜ちゃんと理解する仮想化技術〜 第2回 Intel VT-x の概要とメモリ仮想化

## VMCS の構造

それでは、前回のハイパーバイザのライフサイクルの説明でも度々紹介したハイパーバイザ側のコンフィグレーション情報である VMCS（Virtual Machine Control Structure）の内部構造（図 1）を以下に説明します。

▼図 1　VMCSの内部構造

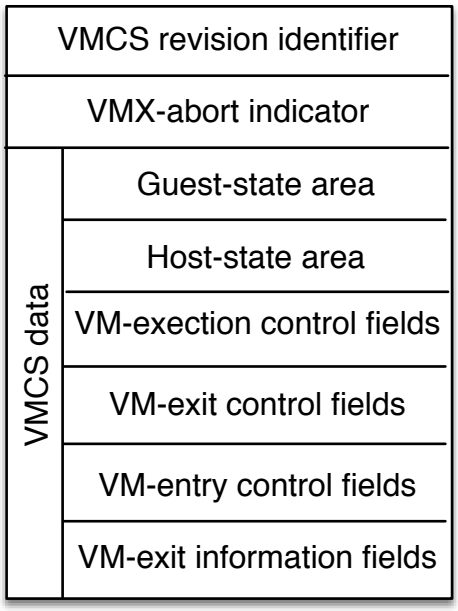


図 1　VMCS の内部構造

### VMCS revision identifier

VMCS のデータフォーマットのリビジョン番号。VT-x が拡張され、古い CPU と新しい CPU では VMCS のフォーマットが異なる可能性があるため、バージョンチェックのためリビジョンが書き込まれる。ハイパー

バイザでサスペンド／レジュームやマイグレーションを実装する時に異なる Intel CPU 間で VMCS をセーブ／ロードした時に不整合が発生するのを防ぐことを意図している

### VMCS-abort indicator

VMExit 中にエラーが発生し、正常に VMExit 要因などのデータが VMCS へ書き込めなかった場合、エラーコードがここに書き込まれる。「VMExit が失敗した場合」なので、正常に動作している限りは使われない

### VMCS data

VMCS の本体部分で、通常はこの領域のいずれかのフィールドを読み書きする

### Guest-state area

VMExit 時にゲストのレジスタを退避し、VMEntry 時に復帰するための領域です。次のレジスタが保存の対象：

- CR0
- CR3
- CR4
- DR7
- RSP
- RIP
- RFLAGS
- CS
- SS
- DS
- ES
- FS
- GS
- LDTR
- TR
- GDTR
- SMBASE

また、以下の MSR レジスタが保存の対象：

- IA32_DEBUGCTL

- IA32_SYSENTER_CS
- IA32_SYSENTER_ESP
- IA32_SYSENTER_EIP
- IA32_PERF_GLOBAL_CTRL
- IA32_PAT
- IA32_EFER

さらに、レジスタ以外のステートとして、仮想 CPU の状態・各セグメントの状態や属性・割り込みブロック状態の有無・VMX プリエンプションタイマのカウンタ・EPT の PTE アドレス等の情報も記録されている。

ここで保存の対象になっていないレジスタについてはハイパーバイザが退避と復帰を行う必要がある

### Host-state area

VMEntry 時にハイパーバイザのレジスタを退避し、VMExit 時に復帰するための領域。次のレジスタが 保存の対象になっている：

- CR0
- CR3
- CR4
- RSP
- RIP
- CS
- SS
- DS
- ES
- FS
- GS
- LDTR
- TR
- GDTR

また、以下の MSR レジスタが保存の対象になっている：

- IA32_SYSENTER_CS
- IA32_SYSENTER_ESP
- IA32_SYSENTER_EIP
- IA32_PERF_GLOBAL_CTRL
- IA32_PAT

- IA32_EFER

ここで保存の対象になっていないレジスタについては、ハイパーバイザが退避と復帰を行う必要がある

### VM-execution control fields

ゲストマシン実行時の CPU の挙動を設定するエリア。どのようなイベントで VMExit するかという情報はここに含まれている。各種の VMExit 要因のオン／オフフラグの他、EPT（メモリ仮想化支援）のオン／オフフラグ、EPT のアドレス、仮想 Local APIC の設定、VPID のアドレスなどが含まれる

### VM-exit control fields

VMExit 時の CPU の挙動を設定するエリア。外部割り込み要因で VMExit したときの CPU の挙動や、いくつかの MSR を退避・復帰機能のオン／オフフラグ、ハイパーバイザの 64 モードのオン／オフフラグなどが含まれる

### VM-entry control fields

VMEntry 時の CPU の挙動を設定する。ゲストマシンへの割り込み挿入のフィールドやいくつかの MSR の復帰機能のオン／オフフラグなど、ゲストの 64bit モードのオン／オフフラグなどが含まれる

### VM-exit information fields

VMExit 時の exit 要因がここに書き込まれる

## VMExit 要因

VT-x ではリスト 1 のような要因による VMExit を発生させることができます。これらの要因で VMExit を発生させるかどうかは VMCS 上の「VM-execution control fields」で設定し、発生した場合 VMCS 上の「VM-exit information fields」に書き込まれます。

### リスト１　VMExit 要因

- 例外か NMI 割り込みの発生
- 外部割り込みの発生
- トリプルフォールト
- INIT シグナルの受信
- SIPI の受信

- SMI の受信
- 内部割り込みの発生
- タスクスイッチ
- CPUID 命令の実行
- Intel SMX 関連の命令の実行
- HLT 命令の実行
- キャッシュ関連の命令の実行（INVD, WBINVD）
- TLB 関連の命令の実行（HNVLPG, INVPCID）
- I/O 関連の命令の実行（INB,OUTB など）
- パフォーマンスモニタリングカウンタ関連の命令の実行（RDPMC）
- タイムスタンプカウンタ関連の命令の実行（RDTSC）
- SMM 関連の命令の実行（RSM）
- VT-x 拡張命令セットの実行
- コントロールレジスタへのアクセス
- 拡張コントロールレジスタへのアクセス
- デバッグレジスタへのアクセス
- MSR へのアクセス
- MONITOR/MWAIT 命令の実行
- PAUSE 命令の実行
- APIC レジスタへのアクセス
- GDTR、IDTR、LDTR、TR レジスタへのアクセス
- VMX プリエンプションタイマのエクスパイヤ
- RDRAND 命令の実行

## VT-x 拡張命令セット

ハイパーバイザから VT-x の機能へアクセスするために、次のような命令が追加されています（ただし、VMCALL と VMFUNC についてはゲストマシンから呼ばれることが想定されています)。各命令のさらに詳しい説明については (“Intel(R) 64 and IA-32 Architectures Software Developer Manuals”) を参照してください。

### VMCS メンテンナンス命令

VMCS 内のデータは普通のメモリアクセス命令で読み書きするのではなく、VMREAD/VMWRITE を経由する必要があります。ゲストマシン起動時の手順は、VMPTRLD で VMCS のアドレスを設定、VMCLEAR で初期化、VMWRITE でフィールドへ初期値を書き込んでから VMEntry となります。

### VT-x 管理命令

ハイパーバイザ上で一連の VT-x 拡張命令セットを利用するには、まず VMXON を実行する必要があります。VMX non Root Mode へ VMEntry するには、初めて VMEntry するときだけ VMLAUNCH、以降 VMExit から復帰するときは VMRESUME となります。

### ハイパーコール命令

ゲストマシンからハイパーバイザを呼び出して何らかの処理を依頼するための命令です。準仮想化を実現するために使われることがあります。EPT 操作命令メモリアクセスの仮想化支援（EPT）が有効なハイパーバイザで使われる命令です。EPT については後述します。

### ページングとメモリの仮想化

マルチタスクをサポートする近代的な OS では、プロセスごとに独立したメモリ空間を持ち、あるプロセスは別のプロセスのメモリ領域へアクセスできないようになっています。

これを実現するために、CPU に「仮想メモリ」をサポートする機構（MMU）が搭載されています[*1]。

仮想メモリの実現方法として、現在では一般的に「ページング方式」がとられています。この方式では、プロセスごとの仮想メモリ空間を固定長（x86 アーキテクチャでは一般的に 4KB）に区切り、仮想ページと物理ページの割り当て情報を「ページテーブル」と呼ばれるメモリ上の表に記録します（図 2）。

CPU からメモリへのアクセスがあった時、MMU はページテーブルを用いて仮想アドレスを物理アドレスへ変換し、適切な物理メモリアドレスへのアクセスを可能にします。

ページテーブル上の各ページに対するエントリには対応する物理ページの情報以外にも、アクセス権（読み／書き／実行）のような属性情報が存在します。

これによって特定のメモリ範囲に対してアクセス制御を行うようなことができます。

また、アクセスの少ない物理ページをディスクへ書き出しページテーブル上の割り当て情報を解除（ページアウト）します。そして後にアクセスが起こったときにディスクから読み込んでページテーブル上の割り当て情報を再設定（ページイン）することで、物理メモリ容量よりも大きな仮想メモリを扱うことができるようになります。

仮想化されたシステムにおいて、CPU 上でそのままゲスト OS を実行する時、このページング機構を仮想化する必要が生じます。これは、ゲスト OS 上のプロセスに割り当てられた仮想ページの参照先である物理ページのアドレスが指しているのはハイパーバイザがゲスト環境へ割り付けたメモリ領域内のアドレスであり、ハイパーバイザが管理する物理メモリ空間上のアドレスではないからです。

---

[*1] 別の方式としてセグメント方式というものがあり、x86 アーキテクチャはこの方式からページング方式へ移行してきたという歴史的事情があるため、今でもセグメント方式をサポートしています。

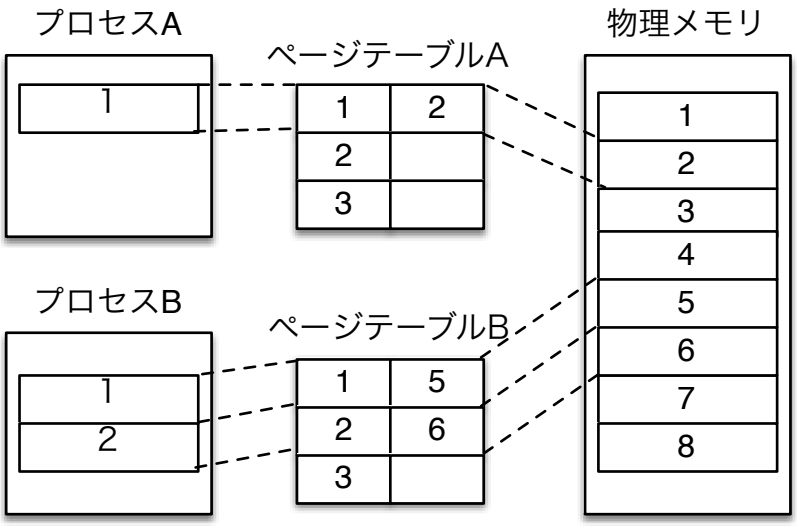


図 2　ページテーブル

たとえば、仮想マシン A へ 1 から 4 まで、仮想マシン B へ 5 から 8 までの物理ページを割り当てたとします。それぞれの仮想マシン上の物理ページ 1 へアクセスが行われる時、実際には仮想マシン A からであれば物理ページ 1 へ、仮想マシン B からであれば物理ページ 5 へアクセスが行われなければなりません（図 3）。

この問題を解決するメモリ仮想化の手法として、ソフトウェアによる「シャドーページング」、ハードウェアによる「EPT」の 2 つを紹介します。

## x86 アーキテクチャにおけるページング機構

まず、x86 アーキテクチャにおけるページング機構について、簡単におさらいしましょう。x86 アーキテクチャでは 1 ページのサイズは 4KB であるため、32bit モードでのページ総数は 1,048,576 ページとなります。このページテーブルを単純な 2 次元の表として表すと消費メモリ量が大きくなってしまいます。

そこで、x86 アーキテクチャではページディレクトリテーブルという概念を導入しました。これにより、少ないサイズでページテーブルを表現しています。1 つのページテーブルは 1024 エントリとし、1 つのページテーブルに 4MB の仮想メモリ空間の情報を持たせます。

ページディレクトリテーブルは、このページテーブルのアドレス情報を格納します。ページディレクトリテーブルは 1024 エントリあり、4GB の仮想メモリ空間全体を表現しています（図 4）。

64bit モードの場合は 8 バイトのページテーブルエントリがテーブル当たり 512 エントリとなり、さらにページディレクトリポインタテーブルとページマップレベル 4 テーブルの 2 段が追加されて 256TB の仮想メモリ空間をサポートしています[*2]（図 5）。

[*2] 32bit PAE モードの場合も 64bit モードのように段数が追加されて仮想メモリ空間が拡張されますが、ここでは解説を省略します。

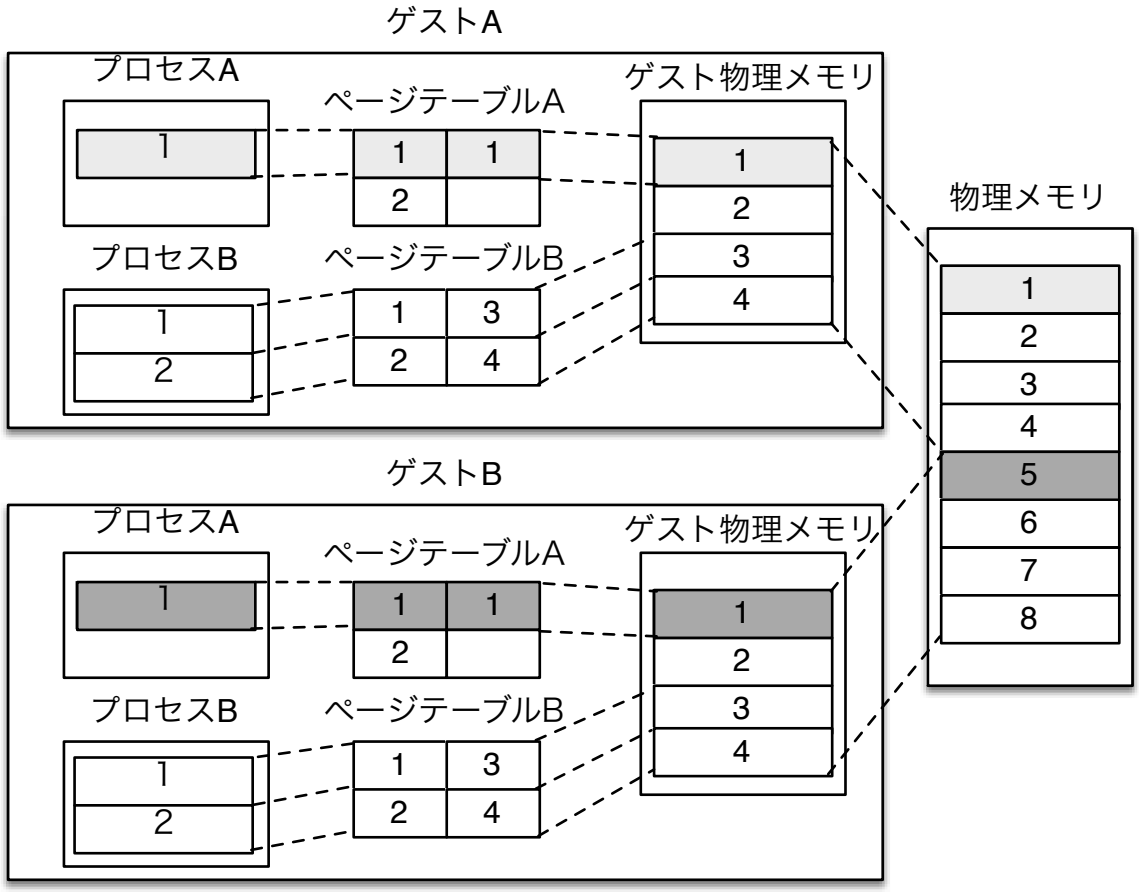


図 3　メモリの仮想化による問題

複数段にするだけではページテーブルの総エントリ数は変わりません。しかし、未使用の仮想メモリ空間も発生するため、必要な仮想メモリ空間に対してのみページテーブルを作成すれば、その分消費メモリを抑えることができます。

ページディレクトリテーブルのアドレスを CPU にセットするためのレジスタとして CR3 レジスタが用意されています。MMU は仮想メモリ空間へのアクセスに対して CR3 レジスタに書かれたページディレクトリテーブルのアドレスへアクセスして探索を行います。探索を行う時、仮想アドレスの先頭 10 ビットをページディレクトリテーブルへのオフセットとしてページテーブルを指すエントリを探索します。

ページテーブルでは仮想アドレスの 11～20 ビットをオフセットとして物理ページを指すエントリを探索します。見つかった物理ページのアドレスに対して仮想アドレスの残り 12 ビットの値を足したアドレスが、CPU がアクセスしようとしている物理アドレスとなります。

ページディレクトリテーブル／ページテーブルの各エントリにはページテーブル／物理ページのアドレスだけでなく、いくつかの属性情報が含まれます。

次にその詳細を示します。

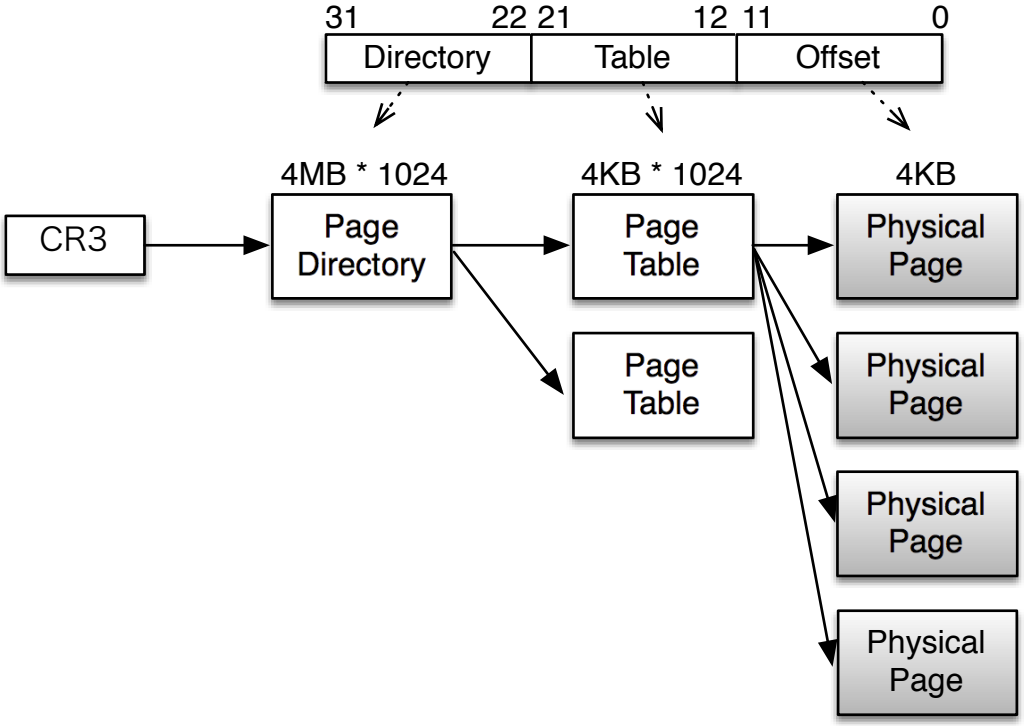


図 4　仮想メモリ空間のページング 2 段構成

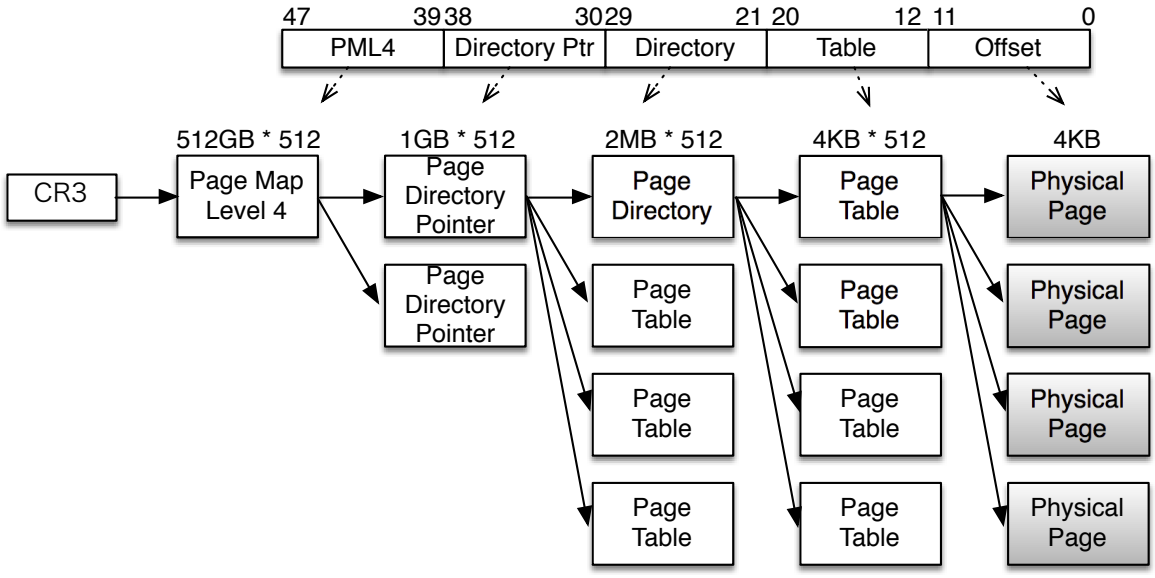


図 5　64bit モードの仮想メモリ空間

- プレゼントビット……1 なら物理メモリ上に存在、0 ならページフォルト例外
- リードライトビット……0 の時に書き込みが実行されたらページフォルト例外
- ユーザスーパーバイザービット……0 なら Ring3 からアクセス禁止
- ページライトスルービット……0 ならライトスルー、1 ならライトバック
- ページキャッシュディスエーブルビット……1 ならキャッシュ禁止
- アクセスビット……ページにアクセスした時に CPU が 1 にする
- ダーティービット……ページに書き込んだ時に CPU が 1 にする
- PAT ビット……ページで PAT を有効にする
- グローバルビット……1 ならグローバルなページ、TLB の挙動に影響する
- ベースアドレス（20 ビット）……ページディレクトリテーブルの場合はページテーブルのアドレス、ページテーブルの場合は物理ページのアドレス

64bit モードの場合は、ベースアドレス長を拡張するためにページテーブルエントリが 64bit へ拡張されます。たとえば、物理メモリの空き容量が低下した等の理由でページアウトを行う時は、物理ページをディスクへ書き出した後、プレゼントビットを 0 に設定します。ページアウトされた仮想ページへ CPU がアクセスした場合、ページフォルト例外が発生します。ページフォルト例外のハンドラは、ディスクからページアウトされたページをメモリへロードし、ページテーブルのエントリへ新しい物理ページのベースアドレスを書き込み、プレゼントビットを有効にします。

他には、カーネルモードとユーザモードでページテーブルを共有する場合、カーネルの領域をユーザスーパーバイザービット＝0 にしておき、ユーザの領域をユーザスーパーバイザービット＝1 にしておくという使い方をすることで、ユーザモードプログラムからカーネルのメモリ領域へアクセスされることを防ぎつつページテーブルの共有を可能にすることができます。ページングについてのより詳細な情報を得るには、(“Intel(R) 64 and IA-32 Architectures Software Developer Manuals”) の Chapter 4. Paging を参照してください。

## シャドーページング

ゲスト OS が各プロセスに対して作成したページテーブルとページディレクトリテーブルをそのまま実際の CPU の CR3 レジスタに設定すると、現実の物理メモリ空間が参照されてしまいます。これではハイパーバイザが意図しないメモリ領域を参照することになってしまいます。これを防ぐため、ハイパーバイザはゲストのページディレクトリテーブルとページテーブルを複製し、ハイパーバイザがゲストマシンに割り当てたメモリ範囲を反映した物理ページ番号をページテーブルの各エントリに書き込み、CPU の CR3 レジスタに複製したページディレクトリテーブルを書き込みます（図 6)。

これにより、ゲストマシンの実行時に仮想ページへアクセスが生じた時、ゲスト OS が設定したゲストマシン内の物理ページ番号ではなく、ハイパーバイザが設定した実際の物理ページ番号を参照するようになります。以上により適切なメモリ領域へ参照が行えるようになります。

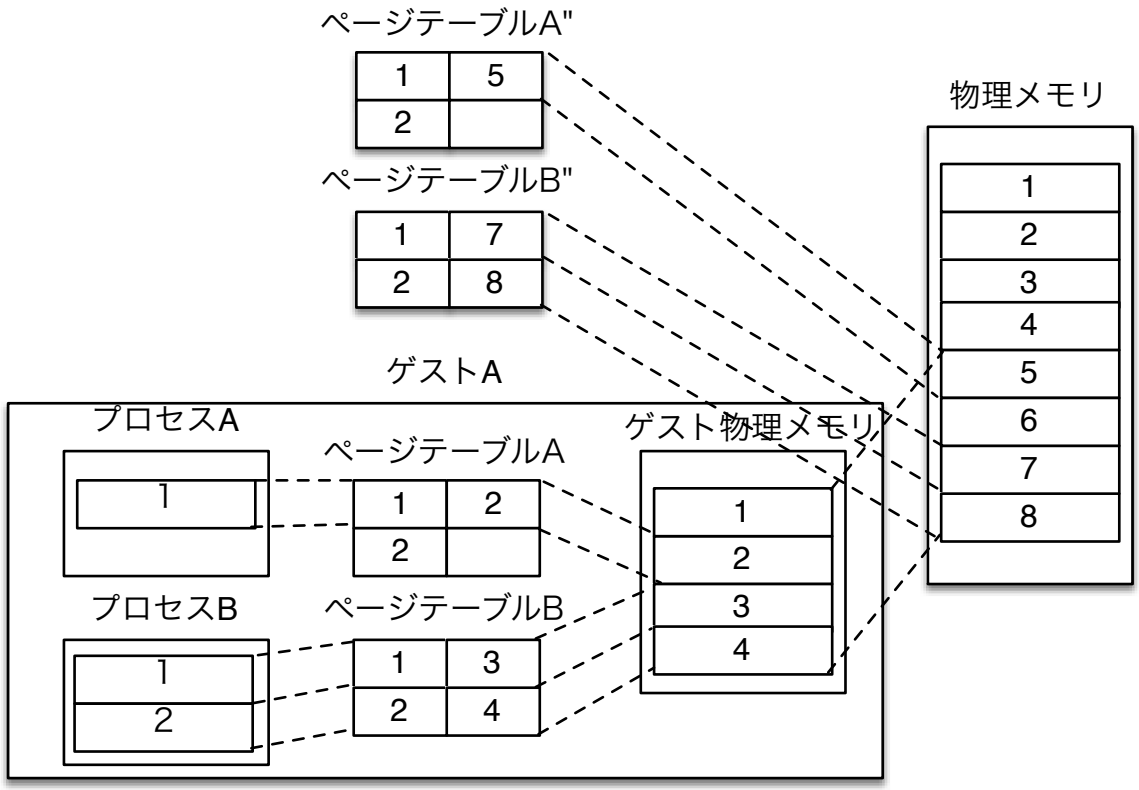


図 6　シャドーページング

## EPT

Nehalem 以降の Intel の CPU では、メモリ仮想化を CPU で支援する機能である「EPT」が追加されました。EPT をゲストマシンで有効にするには、64bit モードにおけるページングと同様に 4 段のページテーブルを作成し、ゲスト物理アドレスからホスト物理アドレスへのマッピング情報を書き込み、VMCS の VM Execution control field にある Extended Page Table Pointer（EPTP）にページマップレベル 4 テーブルのアドレスをセットします。

これにより、ゲストマシンから仮想メモリ空間への参照が行われた時に、CR3 にセットされたページテーブルでゲスト物理アドレスへの変換が行われます。さらに EPTP にセットされたページテーブルでゲスト物理アドレスからホスト物理アドレスへの変換が行われます（図 7)。

シャドーページングの場合は、CR3 レジスタやページテーブルエントリへの読み書きのたびにシャドーページテーブルを構築していました。このときに VMExit が発生し、オーバヘッドとなっていましたが、EPT ではこの処理が省かれることにより、パフォーマンスが向上します[*3]。

[*3] ハイパーバイザの実装によっては 30 %ほど高速化すると言われています。

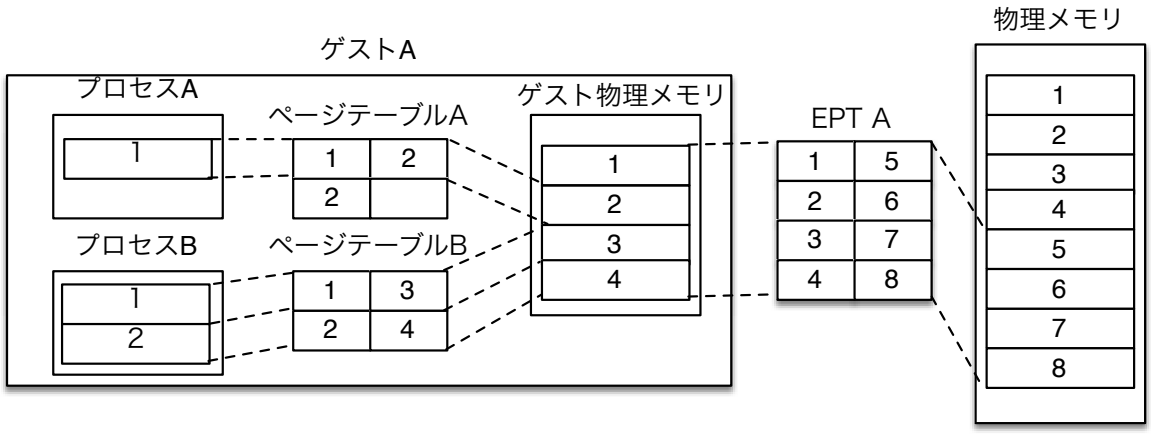


図 7　ゲスト物理アドレスからホスト物理アドレスへの変換

EPT のページテーブルエントリは、通常のページテーブルエントリとフィールドの割り当てが異なります。

## VPID

直接 EPT に関連する機能ではありませんが、メモリに関連するハードウェア仮想化支援機能で EPT とともに Nehalem 以降の Intel の CPU へ導入された「VPID（Virtual Processor Identifier)」を解説します。

アドレス変換を高速化するため、CPU にはページテーブルエントリをキャッシュする TLB という機構が搭載されています。これは、仮想アドレスと物理アドレスのペアを記憶する連想メモリ（CAM）であり、少数の頻繁にアクセスされる領域をキャッシュします。

また、ページテーブルエントリが書き換えられた時やページテーブルが切り替えられた時に「フラッシュ」操作を行い、エントリを削除してページテーブルとの整合性を保ちます。

ゲストマシンを VMExit してホスト OS でプロセスを実行したり、別のゲストマシンを実行する時、TLB をフラッシュしないと仮想アドレスのアドレス解決が誤動作を起こします。このため、初期の VT-x 対応 CPU では VMEntry 時／ VMExit 時に TLB をすべてフラッシュする必要がありました。

Nehalem 以降の Intel の CPU では、これを防いでパフォーマンスを向上させる機能として VPID が導入されました。VPID を有効にするには、VMCS の VM Execution control field にある VPID フィールドにそのゲストマシンのユニークな ID を設定します。

VPID 有効時には、VMEntry 時／ VMExit 時に INVVPID 命令により直前に実行していたゲストマシンのエントリのみ TLB からフラッシュできるようになります。

## まとめ

いかがでしたでしょうか。今回は、前回の解説を補足するために VMCS と VT-x 命令セットの解説を行い、さらにメモリ仮想化について見てきました。

次回は「I/O の仮想化」について解説します。

## 謝辞

校正に関して協力してくださった金子 直矢さん、水野 貴史さん、太居司さん、忠鉢 洋輔さんありがとうございました。

## ライセンス



## 参考文献

“Intel(R) 64 and IA-32 Architectures Software Developer Manuals.” http://www.intel.com/content/www/us/en/processor